UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

APUNTES DE ESTRUCTURAS DE DATOS

ING. JORGE I. EUÁN A.

APUNTES DE ESTRUCTURAS DE DATOS

Orden

APUNTES DE ESTRUCTURAS DE DATOS

ISBN 968 - 58 - 0517 - 2

Impreso en México Printed in Mexico

FACULTAD DE INGENIERIA

Director:

Javier Jiménez Espriú

Secretario General:

Roberto Ruíz Vilá

Jefe de la División de Ingeniería Mecánica y Eléctrica:

Odón de Buen Lozano

Secretario de la División:

Wilbert Arcila Rodríguez

Jefe de la Unidad de Apoyo Editorial:

Irma Hinojosa Félix

COORDINACION DEL SISTEMA DE UNIVERSIDAD ABIERTA

Coordinador:

Oscar Zorrilla Velázquez

Secretario Académico:

Jaime E. Cortés Arellano

Secretario Técnico:

Ma. Teresa Quintanilla Martínez

PRESENTACION

De acuerdo a las estadísticas de los últimos seis años, se inscriben a la carrera de Ingeniero Mecánico Electricista de la Facultad de Ingeniería de la U.N.A.M. un promedio anual de mil doscientos alumnos de nuevo ingreso. Sin embargo, en ese mismo período sólo aprueban su examen profesional y reciben su título, aproximadamente trescientos alumnos. Considerando que un cincuenta por ciento de los estudiantes, por diversos motivos abandona definitivamente la Facultad en los dos primeros años posteriores a su inscripción, se puede concluir que sólo la mitad de los seiscientos restantes se recibe.

De esta manera aumenta año con año, el número de alumnos que deben una o varias materias y que ya han sobrepasado el tiempo límite de siete años y medio que fija la Legislación Universitaria para ser considerados como alumnos regulares con derecho a reinscripción.

Esta misma Legislación otorga a los estudiantes en esta situación tres alternativas para terminar sus estudios. La primera, aplicable a cualquier alumno, es la de presentar seis exámenes extraordinarios por semestre; la segunda, válida solamente para los estudiantes que ya hayan acreditado el sesenta por ciento de los créditos totales de la carrera, es la de inscribirse en dos materias como oyentes y presentar su examen final en el tercer período de exámenes extraordinarios del semestre; y la tercera, aplicable exclusivamente a los alumnos que ya han pagado su servicio social y acreditado su seminario y no deban más de dos materias, es la de presentarlas en "examen especial", en cualquier fecha.

Resultado de la situación analizada es la existencia de un gran número de alumnos que tienen que presentar exámenes extraordinarios para la terminación de sus estudios.

El alumno que decide presentar un examen extraordinario comúnmente, no encuentra un apoyo institucional adecuado, ni sabe con seguridad qué es lo que tiene que estudiar, ya que el contenido de los exámenes depende de los profesores y no existe, un texto básico con el que el estudiante se puede preparar adecuadamente.

Con el propósito de analizar y ayudar a resolver este problema, la División de Ingeniería Mecánica y Eléctrica convocó, en marzo de 1981, a un seminario para buscar una metodología adecuada tendiente a facilitar la preparación de exámenes extraordinarios. En este seminario participaron los funcionarios y profesores de la División y especialistas del Centro de Servicios Educativos de la Facultad de Ingeniería y del Sistema de Universidad Abierta de la U.N.A.M.

En él, se llegó a la unánime conclusión de que la mejor solución para ayudar a los alumnos en la preparación de exámenes extraordinarios, sería llevarlos a cabo a través del Sistema de Enseñanza Abierta, con el apoyo de un profesor que sirva de tutor personal del estudiante, asimismo se concluyó, que el éxito de esta actividad, es la de producir los textos adecuados de autoaprendizaje.

Para comenzar los trabajos se seleccionaron de las materias a cargo de los diferentes Departamentos de la División de Ingeniería Mecánica y Eléctrica, aquéllas que habían presentado mayor dificultad de aprobación y se comisionó a diferentes profesores para la elaboración de los textos de autoaprendizaje correspondientes.

Algunos de estos materiales fueron elaborados en colaboración con otros profesores, y otros por profesores que dirigían personalmente un seminario de tesis en colaboración con sus alumnos.

La obra de Estructuras de Datos, representa el esfuerzo realizado en forma coordinada por profesores del Departamento de Computación, personal de la Unidad de Apoyo Edi-

torial de la Facultad de Ingenieria y la Coordinación del Sistema de Universidad Abierta de la U.N.A.M.

Esperamos que estas publicaciones cumplan con su propósito fundamental, ayudar a los alumnos en la preparación de exámenes extraordinarios, y que sirvan adicionalmente de base para el desarrollo de una adecuada metodología de autoaprendizaje que ayude a la formación de los alumnos regulares que cursan normalmente estas materias.

Toda crítica constructiva de profesores y alumnos en general, nos permitirá el constante mejoramiento de los contenidos y presentación de esta obra.

A t e n t a m e n t e

Ing. Odón de Buen Lozano
Jefe de la División de
Ingeniería Mecánica y
Eléctrica.

P R O L O G O

Recientemente las ciencias de la computación han cobrado gran importancia debido a los avances científicos y tecnológicos cuyos resultados han permitido su utilización en áreas tales como: la industria, la educación, la salud, la economía y otras actividades cotidianas del ser humano.

El reconocimiento de la computación como un área de ejercicio profesional bien definida con gran integración en otras disciplinas, llevó a la Facultad de Ingeniería de la U.N.A.M. a crear la carrera de Ingeniero en Computación para preparar los profesionales que el país requiere.

En estos apuntes, se pretende cubrir el programa vigente de la asignatura de Estructuras de Datos que forma parte del plan de estudios aprobado por el H. Consejo Técnico de esta Facultad.

Los apuntes se desarrollan siguiendo tres ideas principales: la organización de la información; su representación en la computadora y, las operaciones que se realizan sobre la información.

La unidad I presenta los elementos básicos necesarios para iniciar el estudio de las Estructuras de Datos.

La unidad II estudia las estructuras de datos elementales, las unidades III y IV las listas lineales y no lineales a las que se han denominado en este material Estructuras de Datos Compuestas.

La unidad V aborda un tipo de estructura de datos denominada archivo y que se trata por separado para su mejor comprensión.

Las unidades VI y VII, tratan sobre dos de las operaciones más utilizadas en el procesamiento de la información: el ordenamiento y la búsqueda.

Estos apuntes, por sí solos, constituyen una guía de estudio para la comprensión de la asignatura de Estructuras de Datos, no obstante es recomendable acudir a la bibliografía contenida al final de la obra con el fin de que el lector amplíe y profundice sobre aquellos tópicos de su interés.

El mejoramiento de esta obra podrá lograrse con la ayuda de las críticas y sugerencias que alumnos y profesores proporcionen al Departamento de Computación, mismas que serán valiosas para la elaboración de futuras ediciones.

Expresamos nuestro reconocimiento a los señores profesores:

JORGE IVAN EUAN AVILA

y

LUIS G. CORDERO BORBOA

por su valiosa intervención en la elaboración de estos apuntes, así como a:

IRMA HINOJOSA FELIX

por su colaboración en la adaptación pedagógica de los mismos y a los alumnos Vicente Cordova R. y José Luis Rodríguez, que en cumplimiento de su servicio social contribuyeron a la recopilación de la información.

FACULTAD DE INGENIERIA
DIVISION DE INGENIERIA MECANICA Y ELECTRICA
DEPARTAMENTO DE COMPUTACION

INSTRUCCIONES PARA EL MANEJO DEL TEXTO

Los presentes apuntes han sido elaborados tomando en cuenta los diferentes aspectos que caracterizan a los estudiantes de la División de Ingeniería Mecánica y Eléctrica.

El contenido temático, por su amplitud, se dividió en siete unidades, con lo que se pretende graduar el contenido para que el estudiante logre un mayor conocimiento y comprensión de la asignatura, pretendiendo garantizar el cumplimiento de las metas propuestas.

Con el fin de que esta obra se utilice adecuadamente, a continuación se presentan los elementos didácticos que contiene cada una de las unidades.

1. Objetivo general: Indica la conducta que deben lograr los alumnos al finalizar el estudio de la unidad.

2. Objetivos específicos: Tienden en su conjunto al logro del objetivo general propuesto al comienzo de la unidad.

3. Introducción: Es un comentario general, motivador de los contenidos de la unidad, en el que se resalta la importancia de estos contenidos.

4. Contenido: Es el desarrollo de los temas, incluyen ejemplos que representan aplicaciones concretas de los conceptos teóricos.

5. Ideas guía: Están ubicadas en los márgenes laterales; su finalidad es presentar la idea principal del párrafo al que se refieren, así como, localizar fácilmente una información.

6. Cuestionario de autoevaluación: Son actividades que el alumno debe realizar para que reafirme la comprensión y la aplicación del contenido. Adicionalmente le permiten medir el grado en que logró los objetivos de aprendizaje propuestos.

Al final de los apuntes el alumno podrá encontrar:

7. Examen de autoevaluación: Este examen es un instrumento que le permite al alumno verificar por sí mismo, si ha alcanzadc el mínimo necesario de los objetivos de aprendizaje propuestos en las siete unidades.

8. Solución a los cuestionarios de autoevaluación: En ella se agrupan las respuestas correctas a los reactivos y problemas.

9. Solución al examen de autoevaluación: Esta sección constituye la referencia a partir de la cual el alumno puede comprcbar o cotejar sus respuestas.

10. Bibliografía: La bibliografía tiene por finalidad informar al alumno acerca de las fuentes que puede consultar para prcfundizar sobre los temas que le interesen.

11. Apéndice: En este apartado se encuentran las estructuras de contrcl empleadas para la escritura de los algoritmos.

Se recomienda al alumno que se dedique con empeño al estudio de estos apuntes. Además, si desea ampliar algún concepto o disipar dudas específicas respecto a la asignatura de *Estructuras de Datos*, es conveniente que recurra a la *asesoría que el Departamento de Computación* tiene establecida en la División de Ingeniería Mecánica y Eléctrica.

INDICE GENERAL

UNIDAD I ELEMENTOS PARA EL ESTUDIO DE LAS ESTRUCTURAS DE DATOS

OBJETIVO GENERAL

El alumno comprenderá los aspectos básicos de la estructura de una computadora digital, que le permitirán obtener un marco de referencia para iniciar el estudio de las estructuras de datos.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Reconocerá qué es el hardware y las unidades funcionales en las que se divide.
2. Reconocerá qué es el software.
3. Identificará los elementos asociados a la organización física y lógica de la memoria primaria.
4. Identificará los elementos asociados a la organización física y lógica de la memoria secundaria.
5. Diferenciará la memoria primaria de la memoria secundaria.
6. Resolverá problemas de capacidad de almacenamiento de acuerdo con los conceptos estudiados.

INTRODUCCION

Para el estudio de las estructuras de datos se requiere del conocimiento de algunos aspectos relevantes de la organización y operación de la computadora digital que nos den un marco de referencia.

La unidad comprende la revisión de los componentes generales de una computadora, de ahí se parte hacia aquellos que son relevantes en el diseño y operación de las estructuras de datos, como son los medios de almacenamiento (memorias).

Estos se han dividido para su estudio, en medios de almacenamiento primario y medios de almacenamiento secundario, en ambos se hace una revisión de la organización física y lógica y se definen los conceptos asociados a las estructuras de datos.

I.1 GENERALIDADES

Para el estudio de la programación de una computadora digital que de aquí en adelante llamaremos computadora, máquina o equipo es necesario conocer los elementos físicos que la integran y sus características de operación asociadas al desarrollo de programas. Una buena programación generalmente está asociada al conocimiento de la operación del equipo.

Los elementos de esta herramienta tan poderosa lo constituyen básicamente las partes físicas llamadas el hardware y los programas y datos llamados el software. El hardware está constituido por las partes electrónicas y mecánicas que le permiten a la computadora realizar operaciones; y el software por conjuntos de instrucciones que permiten indicar y secuenciar las operaciones sobre los datos.

Hardware y software

I.1.1 COMPONENTES FISICOS DE UNA COMPUTADORA

Para su estudio, podemos dividir los componentes físicos

de la computadora en tres unidades funcionales: los dispositivos de entrada y de salida, la memoria y la Unidad Central de Proceso (U.C.P.). (Véase figura I.1).

Unidades físicas

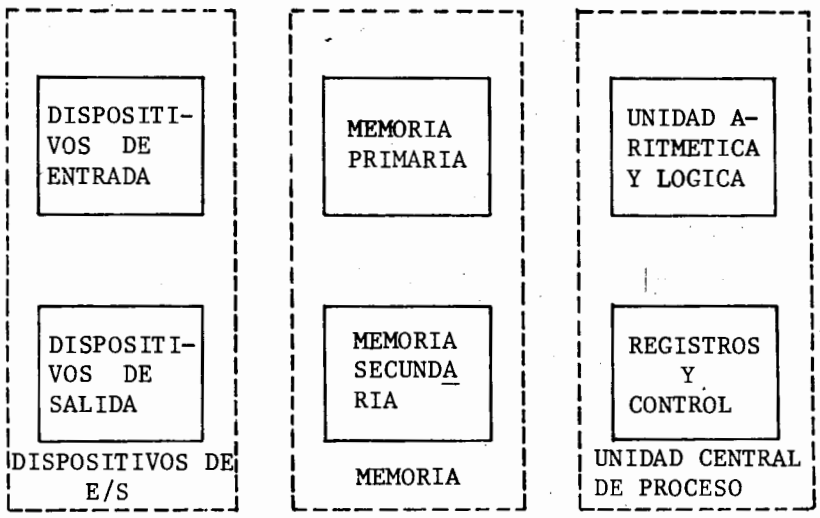


Figura I.1 Unidades funcionales de una computadora.

Los dispositivos de entrada y de salida son los que permiten a la computadora la comunicación con el exterior. Es a través de ellos que los usuarios proporcionan información a la computadora o reciben información de ella.

Dispositivos de entrada salida

Dentro del grupo de los dispositivos de entrada encontramos:

- el teclado de teletipo
- el teclado de terminales de video
- la lectora de tarjetas
- la lectora de cinta de papel
- la lectora de barras
- y otros

Dentro del grupo de los dispositivos de salida encontramos:

- el impresor del teletipo
- la terminal de video
- la impresora de líneas
- el graficador
- la perforadora de cinta de papel
- y otros

Definición de la memoria

La memoria es la unidad de almacenamiento de instrucciones y datos. Para lograr esta función, se utiliza una gran variedad de equipos entre los cuales podemos distinguir los dispositivos de almacenamiento primario y los dispositivos de almacenamiento secundario.

Almacenamiento primario

Los dispositivos de almacenamiento primario son capaces de operar a velocidades electrónicas, y en ellos se encuentran las instrucciones y datos al momento de ejecución. Típicamente consisten en circuitos semiconductores y núcleos magnéticos.

Almacenamiento secundario

Los dispositivos de almacenamiento secundario, operan a velocidades electromecánicas, esto es, a velocidades mucho menores que las anteriores; en ellos se encuentran instrucciones y datos no muy próximos a ser ejecutados. Su construcción está basada en sustancias como óxido de hierro, y se presentan en discos, cintas magnéticas, diskettes, cassettes y otros.

U. C. P.

La Unidad Central de Proceso es el dispositivo que ejecuta las instrucciones proporcionadas por los usuarios. La mayoría de las operaciones indicadas en las instrucciones se practican en la unidad aritmética y lógica, y el secuenciamiento para la ejecución de estas operaciones, es coordinado por la unidad de control.

I.1.2 PROGRAMAS DE COMPUTADORA

Se llama programa a las instrucciones que proporciona el usuario a una computadora para realizar una tarea específica. Dentro de los programas que generalmente se manejan en una computadora encontramos:

Programas del sistema

Programas que facilitan a los usuarios su interacción con la máquina, llamados programas del sistema. Estos son los que generalmente proporciona el fabricante cuando se adquiere un equipo y consisten en:

- sistemas operativos
- compiladores e intérpretes
- ensambladores y macroensambladores
- cargadores y ligadores
- editores
- programas de comunicación
- y otros

Programas de usuarios

Programas que escriben los usuarios para aplicaciones particulares, llamados programas del usuario. Algunos de estos programas son:

- solución de un sistema de ecuaciones
- control de asistencia
- nómina
- ruta crítica
- modelado de sistemas
- y otros

Para aclarar la interacción del software con el hardware, diremos que una computadora digital acepta información proveniente del exterior haciendo uso de los dispositivos de entrada. Esta información, que puede consistir en instrucciones o datos, es almacenada en la memoria para ser procesada por la Unidad Central de Proceso y los resultados obtenidos son enviados al exterior a través de los dispositivos de salida.

I.2 MEMORIA PRIMARIA

La memoria es, como se mencionó anteriormente, el almacén de instrucciones y datos, esto es, constituye el medio en

el que se conservarán los datos y las instrucciones. Por ello nos interesa conocerla mejor.

La memoria primaria se ha dividido para su estudio en dos partes: la organización física, que se refiere a la organización de los componentes físicos, y la organización lógica, que se refiere a las formas de utilización de la memoria.

Organización física y lógica

I.2.1 ORGANIZACION FISICA

La Unidad Central de Proceso toma instrucciones y datos de la memoria primaria en grupos de n bits, llamados palabras de computadora. (Véase figura I.2). Un bit es el nombre que recibe un dígito binario, el cual sólo puede tomar dos posibles estados asociados con el cero y el uno.

Palabra de computadora

La longitud de una palabra de computadora es el número de bits que la componen; en la mayoría de las máquinas esta longitud oscila entre 8 y 64 bits.

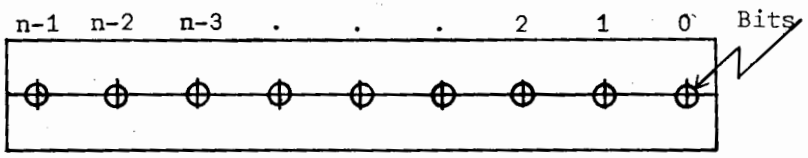


Figura I.2 Palabra de computadora en n bits.

Un bit puede representar dos objetos distintos, de aquí que una palabra de computadora de longitud n pueda representar a 2^n objetos distintos.

La memoria físicamente está constituida por un conjunto de m palabras de longitud n, en donde a cada palabra de computadora se le asocia una dirección, que es un número único, entre 0 y m-1, llamado la dirección de la palabra. (Véase figura I.3).

Dirección de memoria

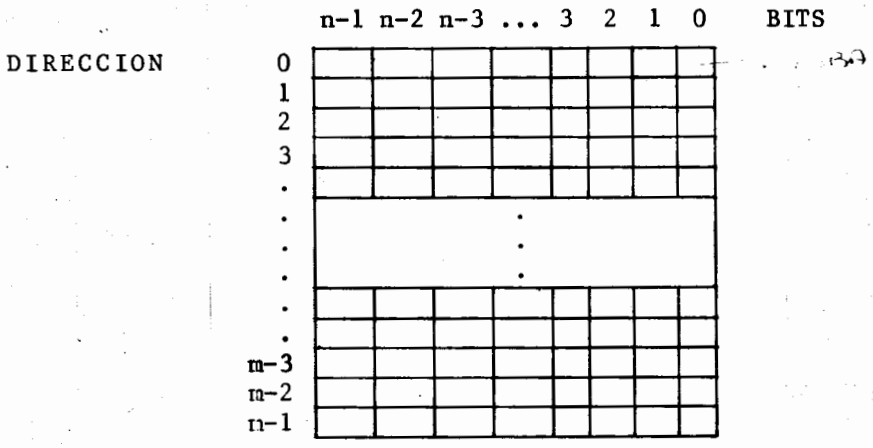


Figura I.3 Organización física de la memoria.

Registro de dirección, registro de datos y líneas de control

Para escribir (almacenar) o leer (recuperar) información de una palabra de computadora desde la unidad central de proceso, existen asociados a la memoria dispositivos como el registro de dirección, el registro de datos y líneas de control, mismos que permiten la realización de estas operaciones. (Véase figura I.4).

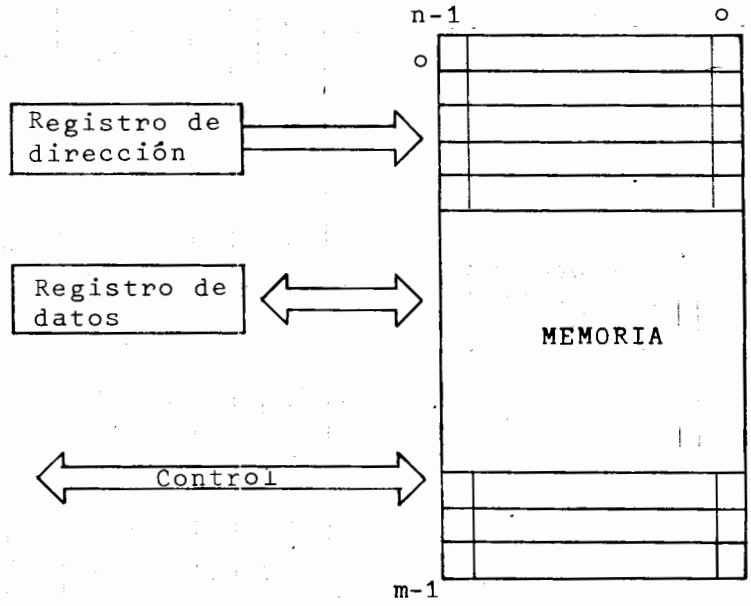


Figura I.4 Conexión entre la memoria primaria y la UCP.

El registro de dirección es un conjunto de k bits, en el cual es posible mantener cualquier dirección entre 0 y m - 1. Este registro es utilizado para seleccionar la dirección de la palabra con la que se desea operar.

Definición de registro de dirección

El registro de datos es un conjunto de n bits en los cuales se tiene el contenido de la palabra direccionada al finalizar una operación de lectura. En una operación de escritura, el registro mantiene la información que será depositada en la palabra direccionada.

Definición de registro de datos

El control permite indicar a la memoria qué operación se va a realizar (escritura o lectura). A través del control, la memoria indica el término de la operación.

Unidad de Control

Para leer o escribir en la memoria, la unidad central carga los registros de dirección y de datos con los valores apropiados y envía la señal para que se inicie la operación. El tiempo entre la señal de inicio de la operación y el término de ésta, es conocido como tiempo de acceso.

El tiempo para leer o escribir en cualquier localidad de la memoria primaria es constante; característica ésta de las memorias de acceso directo (Random Access Memory).

I.2.2 ORGANIZACION LOGICA

Para representar en la memoria la información, ésta requerirá de un cierto número b de bits, dependiendo del tipo de información de que se trate. El número b de bits podrá ser mayor, menor o igual que el número n de bits de la palabra, de aquí que deberá seguirse una estrategia para la representación de la información. Consideremos dos casos para ilustrar lo anterior.

Representación de la información

a. Sea b el número de bits que requiere cierta información para su representación y $b \leq n$. Esto significa que de una palabra de computadora pueden tomarse b bits con-

tinuos para representar la información. (Véase figura I.5).

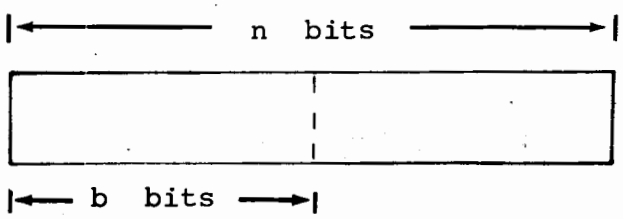


Figura I.5 Representación de la información cuando $b \leq n$.

b. Sea $b > n$; en este caso será necesario involucrar varias palabras de computadora para representar la información. (Véase figura I.6).

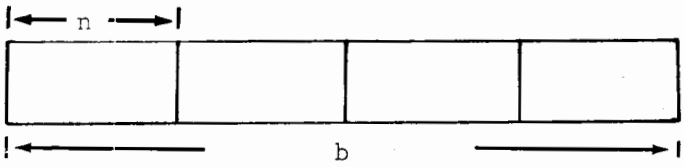


Figura I.6 Representación de la información cuando $b > n$.

Para el caso en que $b > n$, es importante la elección de las palabras de computadora que serán usadas para alcanzar la longitud b. Consideremos los siguientes dos casos:

i. Utilización de palabras que se encuentran contiguas. (Véase figura I.7); esto es, cuyas direcciones son:

$$a,\ a + 1,\ a + 2,\ \ldots$$

Representación de información en palabras contiguas

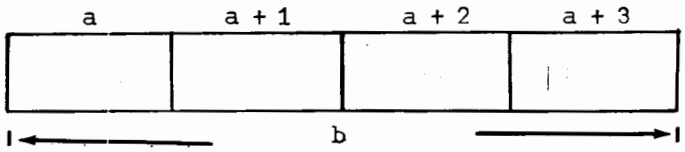


Figura I.7 Uso contiguo de localidades para alcanzar b.

ii. Utilización de cualesquiera localidades de la memoria.

Representación de información en palabras ligadas

La utilización de este esquema requiere de algún mecanismo que establezca cuáles son estas localidades y el orden de utilización de ellas. Para implementar este esquema, una posible alternativa es utilizar de cada palabra un cierto número de bits para indicar la dirección de la siguiente palabra. Este grupo de bits es llamado liga. De esta forma, es posible manipular un conjunto cualquiera de localidades de memoria para representar la información en forma ligada. (Véase figura I.8).

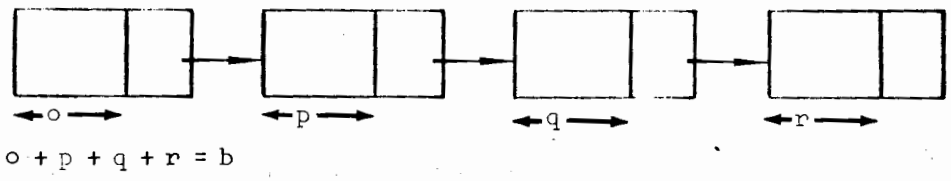


Figura I.8 Uso ligado de localidades para representar b bits de información.

Para que el programador pueda controlar las localidades disponibles para almacenar sus instrucciones y datos se requiere de un esquema de control, generalmente llamado mapa de memoria. De este mapa es posible obtener información tal como: la cantidad de memoria disponible, las áreas de memoria contigua, el número de palabras en cada área, etc. En el caso de las organizaciones ligadas es necesario manejar un fondo común (pool) de localidades disponibles del cual se podrá obtener una o más palabras, y a donde retornarán éstas cuando ya no se requieran. Los programas que retornan las localidades fuera de uso al fondo común son llamados recolectores de basura.

Control de localidades de memoria disponibles

I.3 MEMORIA SECUNDARIA

En la memoria primaria sólo están presentes los programas en ejecución con sus correspondientes datos, debido a que

la UCP los demanda de esta parte de la computadora por la alta velocidad de operación de la memoria. Por ello los programas y datos que no están siendo utilizados momentáneamente deberán estar almacenados en algún otro tipo de memoria.

Almacenamiento de información masiva

Si comparamos la cantidad de información (instrucciones y datos) normalmente en uso con la que no lo está, nos daremos cuenta de que esta última es mucho mayor. Extender la memoria primaria para almacenar esta información resulta inconveniente por lo siguiente:

El aumentar la memoria más allá de la capacidad de direccionamiento de la máquina no tendría ninguna utilidad, pues no podría hacerse referencia en forma directa a todas las localidades de memoria, a menos que se utilice un esquema de direccionamiento adecuado, se desperdiciaría un recurso, ya que no siempre se utiliza toda la memoria; no se tendría la facilidad de remover el medio de almacenamiento y sustituirlo por otro con diferente información, por lo que sería un sistema con muy poca flexibilidad.

Almacenamiento secundario

Estos factores han influido para que se utilicen medios de almacenamiento secundario, que guardan información por un tiempo indefinido, con un costo razonable. Además son bastante flexibles en su uso y, en la gran mayoría de los casos son removibles.

Estos medios de almacenamiento secundario necesitan de unidades especiales (dispositivos electromecánicos) para ser usados. El costo de estos dispositivos puede variar mucho, dependiendo de las características de la memoria secundaria escogida; pero es indudable que el beneficio obtenido recompensa el costo inicial.

Velocidades de almacenamiento en los dispositivos secundarios

Es importantes aclarar que el uso de la memoria secundaria es más lento que el de la memoria primaria, pues las velocidades de los dispositivos electromecánicos que manejan la memoria secundaria, no pueden compararse con las velocidades de los dispositivos electrónicos que manejan la memoria primaria.

I.3.1 ORGANIZACION FISICA

A continuación se estudian dos de los medios de almacenamiento secundario más usados en los sistemas de cómputo: la cinta magnética y el disco magnético.

CINTA MAGNETICA

La cinta magnética es una tira o cinta de material plástico recubierto de una película de material ferromagnético (óxido de hierro), lo que permite que sea grabada en forma continua sobre toda una superficie. Para indicar la longitud utilizable de la cinta se le colocan al inicio y al final marcas reflectantes que pueden ser detectadas por la unidad de cinta.

Descripción física de la cinta magnética

Comercialmente se encuentran cintas de 300, 600 y 2400 pies de longitud, siendo el ancho de éstas aproximadamente de media pulgada. Estas cintas se encuentran enrolladas en carretes, lo que permite su manejo y transportación. (Véase figura I.9).

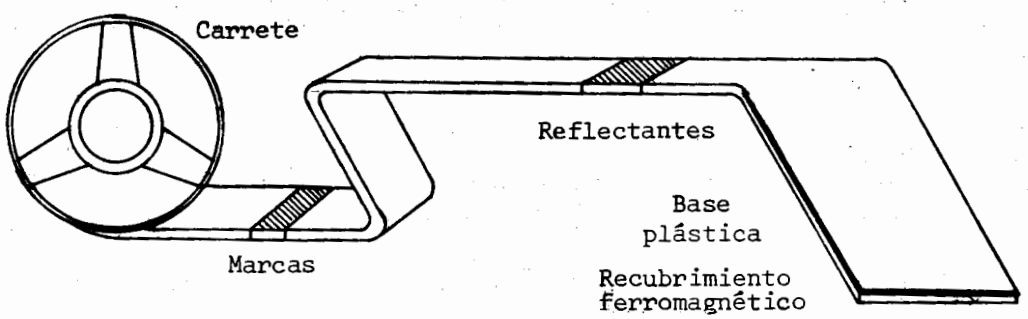


Figura I.9 Cinta magnética.

UNIDAD DE CINTA

Como se había mencionado anteriormente, en el caso de las memorias secundarias es importante diferenciar el medio de almacenamiento (cinta magnética), del dispositivo que permite operarlas (lectora-grabadora de cinta magnética), ya que de éste dependen las distintas modalidades de su utilización.

Específicamente, podemos hablar de cuatro características en la unidad de cinta:

- Densidad de grabación.- Se refiere a la cantidad de información que puede ser almacenada en la cinta por unidad de longitud y que va de 800 BPI (bytes por pulgada*) a 6250 BPI. Asumiendo que las características intrínsecas del material ferromagnético lo permitan, la densidad de grabación dependerá del dispositivo utilizado.
- Número de pistas.- Se refiere al número de bits que pueden grabarse en forma perpendicular a la longitud de la cinta, pudiendo ser de siete o nueve pistas.
- Velocidades de lectura o escritura.- Se refiere a la velocidad con la que pueden leerse o escribirse los datos en una cinta y se da en pulgadas sobre segundo (in/s).
- Velocidad de transferencia de datos.- Esta es una característica importante, ya que permite conocer la velocidad máxima a la que puede moverse la información en el dispositivo. Está relacionada con la velocidad de lectura o escritura y con la densidad de grabación usada. Por ejemplo, si la velocidad de lectura o escritura es de 45 in/s y la densidad de grabación de 1600 BPI, se tendrá una velocidad de transferencia de datos de 72,000 bytes/segundos, que resultan de multiplicar 45 y 1600.

Características de la unidad de cinta magnética

La lectura o escritura se lleva a cabo al pasar la cinta bajo una serie de bobinas (colocadas en forma perpendicular a la cinta) que detectan y magnetizan la película de óxido de hierro al deslizarse. (Véase figura I.10). Estas bobinas se agrupan en lo que se llama una cabeza de lectura o escritura, y precisamente del número de bobinas dependerá el número de pistas de grabación. Es bastante fre-

Lectura o escritura en cinta magnética

*byte = caracter = 8 bits.

cuente que se utilice una de las pistas para grabar un bit de paridad tanto en las grabaciones de siete como de nueve pistas. Este bit de paridad que forma parte de cada byte almacenado podrá tomar los valores 0 ó 1, de tal forma que el número total de bits prendidos en el caracter sea par (paridad par) o impar (paridad impar).

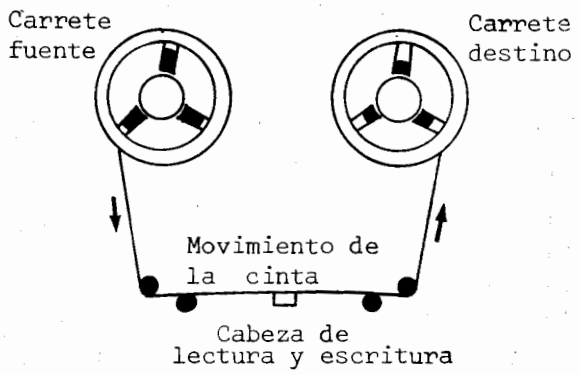


Figura I.10 Grabación en cinta magnética.

De esta manera, para la grabación de nueve pistas, la información queda representada mediante ocho bits de datos y uno de paridad colocados en forma perpendicular a la cinta. En el caso de la grabación de siete pistas, se utilizan seis bits para los datos y uno para la paridad.

Existen diferentes formas de controlar el movimiento de la cinta, pero en la mayoría de los casos es controlada por servomecanismos de columnas de vacío pudiéndose conocer dónde principia la cinta y dónde termina por las marcas reflectantes de comienzo de cinta (Begin Of Tape) y fin de cinta (End Of Tape).

El control de la unidad de cinta se lleva a cabo tanto manualmente como por programa, siendo las operaciones más comunes avanzar la cinta, retrocederla, leer y escribir. Para evitar errores y escribir accidentalmente en una cinta, el carrete en el que viene enrollada tiene un anillo plástico que debe quitarse manualmente para proteger la cinta.

Debemos considerar que, para leer o escribir en la cinta, ésta debe iniciar el movimiento y alcanzar su velocidad nominal, lo que requiere de un cierto tiempo. De la misma forma, cuando dejamos de leer o escribir en la cinta, se requiere un cierto tiempo para que ésta se detenga. Debido a ello, entre grupos de caracteres existe un espacio que no puede ser utilizado, llamado espacio entre registros (Inter Record Gap: IRG), el cual mide de media pulgada a tres cuartos de pulgada.

Al grupo de caracteres entre dos IRG, se le llama registro físico o bloque y su tamaño es un factor muy importante, ya que el número de caracteres almacenado en la cinta disminuye al aumentar el espacio utilizado por los IRG. (Véase figura I.11).

Registro físico o bloque

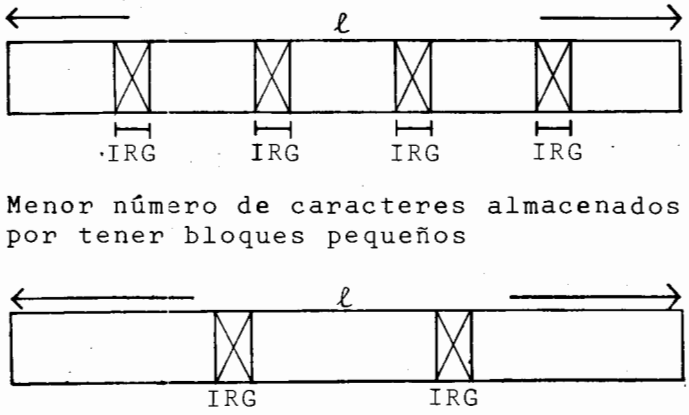


Figura I.11 Uso del espacio entre registros.

En cuanto a los registros físicos, puede ser que todos sean del mismo tamaño o que sean de longitud variable, dependiendo esto, de las características de las unidades de cinta utilizadas. Independientemente un registro físico es la mínima unidad de almacenamiento que puede ser leída o escrita en una cinta magnética.

Una vez que la cinta ha alcanzado su velocidad nominal, el mantener ésta constante es difícil y existen variaciones que impiden que un registro previamente grabado pueda ser

reescrito, ya que si la velocidad es menor, el registro quedará corto; y si la velocidad es mayor, quedará largo.

Ejemplo I.1

A la velocidad de 45 in/s y a una densidad de 1600 BPI, el tiempo para grabar 5120 caracteres (ch) es de (5120 ch) / ((45 in/s) x (1600 ch/in) = 7.111×10^{-2} segundos y la longitud de la cinta que pasó bajo la cabeza en ese tiempo fue de 45 in/s x 0.0711 s = 3.19999 pulgadas.

Ahora, si la velocidad de lectura escritura se incrementa en 10 % a 49.5 in/s y la transferencia se mantiene constante a 72 000 ch/s, para grabar un registro utilizaríamos una longitud de la cinta de:

49.5 in/s x 0.071111s = 3.51999 pulgadas

por lo que este registro resulta más largo y excede el tamaño con que había sido grabado originalmente. Si consideramos que la velocidad de lectura escritura disminuye un 10 % quedando en 40.5 in/s, la longitud de la cinta utilizada para grabar un registro sería de 40.5 in/s x 0.07111s = 2.8799 pulgadas, quedando este registro corto con respecto a la grabación original. (Véase figura I.12).

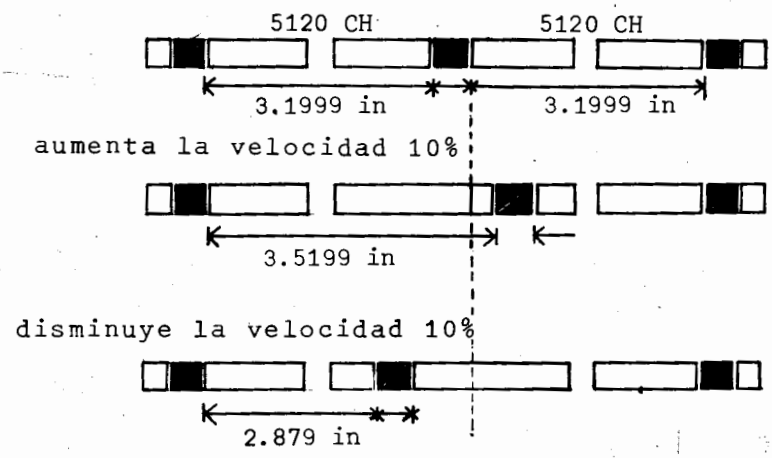


Figura I.12 Efecto producido por alterar la velocidad de grabación.

En general, es difícil lograr una velocidad siempre igual en los procesos de grabación sobre una cinta magnética, ésta es la razón por la cual, una vez que se ha grabado un registro o registros en una cinta, éstos no pueden ser modificados y reescritos sobre sí mismos, y es necesario volver a escribir toda la cinta en caso de modificar alguno de ellos.

En el caso de la lectura de la cinta, no existe ningún problema para leer los registros físicos las veces que se desee, pues sólo se está detectando la información.

Acceso secuencial

Deberá considerarse que debido a las características propias del medio de almacenamiento y de la unidad de cinta para leer o escribir el bloque n, deberán haberse leído o escrito los n - 1 bloques previos en forma secuencial.

Para concluir, diremos que la cinta magnética es un tipo de memoria secundaria organizada en registros físicos cuya dirección está dada por la posición que ocupan en la cinta y se tiene acceso a ellos en forma secuencial.

DISCO MAGNETICO

Descripción física del disco magnético

Este medio de almacenamiento está constituido por un plato o disco metálico en cuyas caras se deposita una capa de material ferromagnético, lo que permite que sea grabado en ambas superficies. (Véase figura I.13).

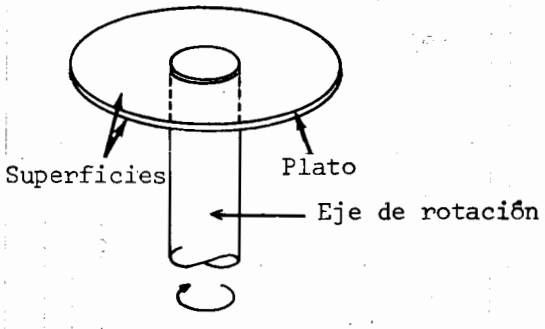


Figura I.13 Disco magnético.

Actualmente existen variantes sobre esta idea original como son:

El disco flexible (diskette o floppy disk) utiliza una base flexible no metálica y puede tener un diámetro de 5¼ pulgadas u 8 pulgadas; éste es un medio de almacenamiento de baja capacidad. (Véase figura I.14).

Disco flexible

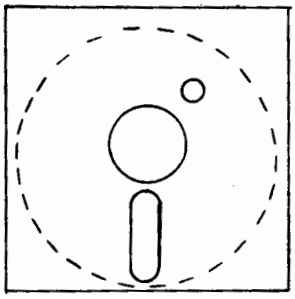


Figura I.14 Disco flexible.

El disco apilado (disk pack) utiliza varios platos del mismo diámetro montados sobre un eje. Este tipo de medio es de gran capacidad de almacenamiento. (Véase figura I.15).

Disco apilado

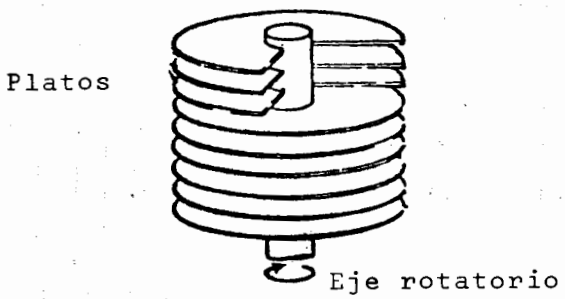


Figura I.15 Disco apilado.

UNIDAD DE DISCO

Al dispositivo electromecánico que se utiliza para leer o escribir en el disco se le llama unidad de disco y fundamentalmente consta de: un motor, que hace girar el eje sobre el que se encuentra colocado el disco o los discos en caso de estar apilados; el motor, que mueve las cabezas de lectura o escritura para colocarlas en cualquier

Lectura o escritura en el disco magnético

posición en la superficie del disco; y los circuitos electrónicos de control.

La unidad de disco lee o escribe información cuando la superficie magnética del disco girando a una velocidad constante pasa bajo la cabeza de lectura o escritura. En la mayoría de los discos, la cabeza no está en contacto con la superficie del disco por lo que cualquier partícula que se interponga entre ellas puede ocasionar serios daños. (Véase figura I.16).

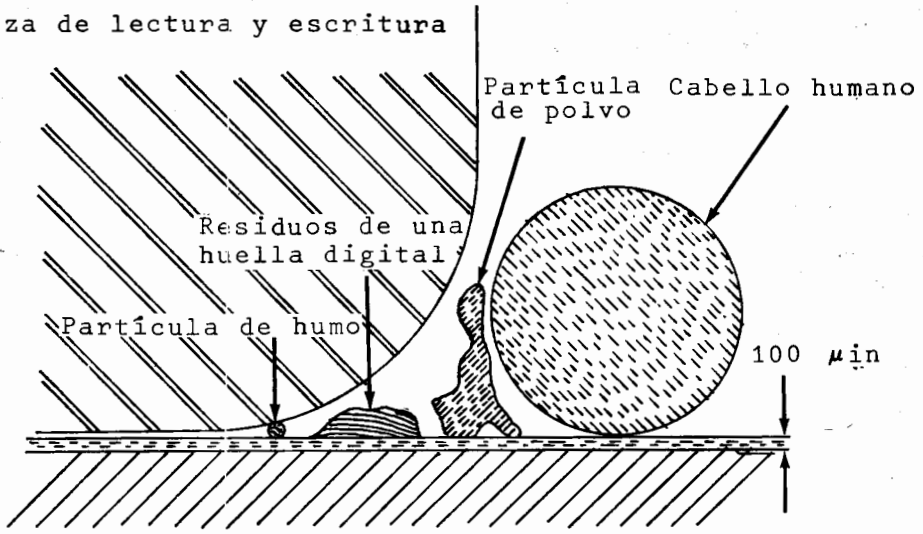


Figura I.16 Grabación de un disco magnético.

Al girar el disco bajo la cabeza y permanecer ésta en una posición fija, al cabo de un cierto tiempo se habrá descrito el lugar geométrico de un círculo formado por el conjunto de puntos que han pasado en una revolución del disco. A este lugar geométrico se le conoce con el nombre de pista (track) y debido a que la cabeza puede deslizarse en forma radial al disco se tienen varias pistas sobre su superficie, en las cuales se leen o escriben los bits en serie.

Pista

La definición de pista dada anteriormente, es válida para los sistemas en los que se utiliza más de un plato y ambas superficies, pudiéndose hablar en este caso de un ci-

Cilindro

lindro formado, hipotéticamente, por la proyección perpendicular de una pista sobre otra en el mismo plato y a su vez, sobre todos los platos en el disco apilado.

Cada pista está subdividida en unidades mínimas de almacenamiento y recuperación de información llamadas bloques o sectores, de tal manera que éstos constituyen el objeto final de un acceso a disco y pueden referenciarse en forma única mediante una dirección dada por los números de cilindro, superficie y sector. (Véase figura I.17).

Bloque o sector

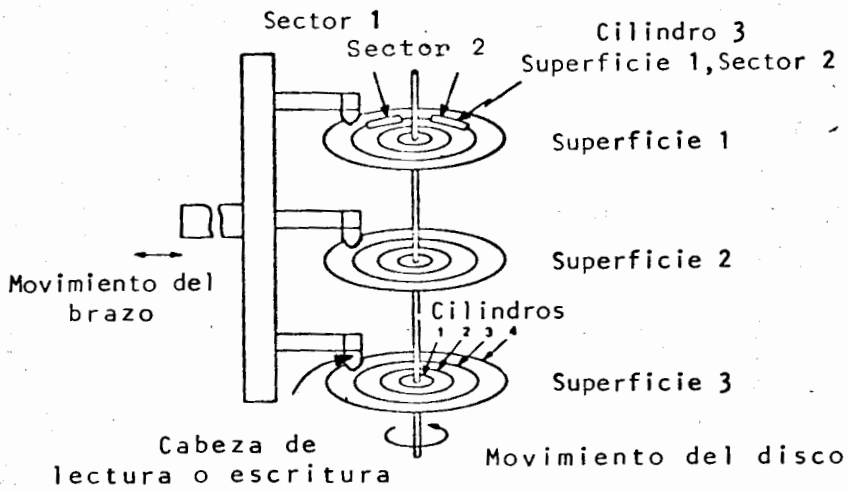


Figura I.17 Organización física de un disco magnético.

El hecho de que se pueda tener acceso a cualquier sector en el disco sin tener restricciones de acceso previo a otros sectores (recuérdese el caso de la cinta), hace que se le considere un dispositivo de acceso directo.

Acceso directo

El manejo de la información en un disco resulta considerablemente más lento que en una memoria primaria, ya que para leer o escribir un sector se requiere que la cabeza tome su posición sobre el cilindro (tiempo de búsqueda); después, debe esperar a que pase bajo ella el sector deseado (retraso rotacional o latencial) y, finalmente, se debe considerar el tiempo requerido para mover la información (tiempo de transmisión). Entonces, se obtiene el tiempo de acceso a un sector mediante la suma del tiempo de búsqueda y el rotacional, y el tiempo para realizar una ope-

Tiempo para realizar una operación de entrada o salida en disco

ración de entrada o salida mediante la suma del tiempo de acceso y el tiempo de transmisión.

Finalmente, existen discos que pueden removerse de la unidad de disco y otros que están fijos, así como unidades en las que las cabezas no se mueven, ya que se tiene una para cada pista, lo que disminuye el tiempo de acceso pero aumenta el costo de los mismos.

I.3.2 ORGANIZACION LOGICA

CINTA MAGNETICA

El usuario de alto nivel de un sistema de cómputo espera que el sistema le ofrezca flexibilidad para manipular la información. En particular, puede suceder que se agrupen conjuntos de palabras con significado especial para el usuario a las que llamaremos registros lógicos. Entonces pueden presentarse dos casos:

Registro lógico

a. Sea p el número de palabras que requieren sus entes lógicos para representar información y ℓ la longitud en palabras del registro físico; entonces si $p \leq \ell$ pueden almacenarse uno o más registros lógicos en un registro físico.

Factor de Bloqueo

Al número de registros lógicos en un registro físico se le conoce como factor de bloqueo y es muy importante, ya que determina el número de registros lógicos que pueden ser transferidos en una operación de entrada o salida.

b. Sea $p > \ell$ entonces será necesario utilizar más de un registro físico para almacenar un registro lógico. En la cinta magnética se utilizan exclusivamente registros físicos contiguos, ya que la forma de acceso a este medio de almacenamiento hace que pierda sentido la utilización de registros físicos ligados.

DISCO MAGNETICO

Cuando utilizamos un disco magnético para guardar información pueden ocurrir los siguientes casos:

a. Sea p el número de palabras que requiere un ente lógico para representar información y ℓ, la longitud en palabras de un sector, entonces, si $p \leq \ell$ pueden almacenarse uno o más registros lógicos en un sector. En ese caso se habla de un factor de bloqueo en forma análoga a como se definió para la cinta magnética.

b. Sea $p > \ell$, entonces será necesario utilizar más de un sector para almacenar un registro lógico. Una de las alternativas consiste en utilizar sectores contiguos en forma análoga a la cinta magnética. Otra alternativa se basa en el uso de apuntadores para lograr que los sectores sean secuenciales, independientemente de su posición física.

Para representar los apuntadores, se utilizan algunos bits dentro de los sectores para formar ligas; también es posible construir una tabla con apuntadores hacia los sectores que representan un ente lógico.

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

1.	Hardware	( ) Permite a los usuarios especificar las operaciones sobre los datos para la realización de una tarea en la computadora.
2.	Software	( ) En ella se realizan las operaciones aritméticas y lógicas.
3.	Unidad Central de Proceso	( ) Constituye las unidades funcionales de la computadora. (CPU - Memoria - E/S).
4.	Dispositivos de entrada y salida	( ) Paquetes y rutinas de utilería.
5.	Palabra de computadora	( ) Almacena gran cantidad de instrucciones y datos no muy próximos a ser ejecutados.
6.	Memoria primaria	( ) Permiten la comunicación con el exterior.
7.	Memoria secundaria	( ) Conjunto de bits que representan instrucciones o datos para la UCP.
		( ) Constituye el lugar donde la UCP toma instrucciones y datos al momento de ejecución.
		( ) Teletipos, impresoras, lectoras, otras.
		( ) Discos, cintas, diskettes, otros.

II. INDIQUE CON UNA X, SI LA AFIRMACION ES FALSA O VERDADERA.

		VERDADERO	FALSO
1.	Si n es el número de bits de una palabra se pueden representar 2^n objetos distintos.	( )	( )
2.	Con el registro de dirección se selecciona el bit n de la palabra m.	( )	( )

3. La dirección para un acceso a disco está totalmente definida si se especifican los números de cilindro, superficie y sector. ( ) ( )

4. Para una operación de escritura a la memoria primaria, el registro de datos almacena la dirección de la palabra. ( ) ( )

5. Si b=6 bits de información y n=13 bits de una palabra, necesitamos más de una palabra para almacenar la información. ( ) ( )

6. Si n=6 bits tiene una palabra y existe una liga=3 bits, el número de bits para almacenar información en ocho palabras es 24. ( ) ( )

7. Con el fondo común no se puede determinar el número de bits que están desocupados. ( ) ( )

8. Los programas y datos que no están en uso se encuentran almacenados en dispositivos llamados de almacenamiento secundario. ( ) ( )

9. Un número grande de IRG aumenta la capacidad de almacenamiento en una cinta magnética. ( ) ( )

10. Al número de registros lógicos contenidos en un registro físico se le llama factor de bloqueo. ( ) ( )

11. La lectura de registros en una cinta se puede practicar secuencialmente. ( ) ( )

12. La lectura o escritura de registros en un disco magnético puede practicarse en forma directa. ( ) ( )

13. El almacenamiento de la información en un disco puede hacerse utilizando sectores contiguos. ( ) ( )

14. La unidad de control indica las operaciones que se efectúan en la memoria y el término de las mismas. ( ) ( )

III. RESUELVA LOS SIGUIENTES PROBLEMAS.

1. ¿Cuántos bits se requieren para representar las placas de un automóvil, si consideramos que son seis símbolos los que aparecen en cada placa? Cada símbolo puede ser cualquiera de las 29 letras del alfabeto o cualquiera de los dígitos del 0 al 9.

2. Considere una memoria de acceso directo de 1024 palabras de 20 bits de longitud. Represente en forma contigua y ligada el siguiente texto utilizando las direcciones comprendidas entre la 900 y 1024. El texto puede contener cualquiera de las 29 letras del alfabeto.
   Texto: "POR MI RAZA HABLARA EL ESPIRITU"
   Elabore los diagramas correspondientes:

3. ¿Cuántos caracteres de 6 bits pueden almacenarse en una cinta de 7 pistas en una longitud de 20 cm a una densidad de 1600 BPI?

4. Si la velocidad de lectura o escritura es de 37.5 m/s y la velocidad de transferencia de datos es de 243 750 bits/s.

   ¿Cuál es la densidad de grabación?

   ¿Cuál es la longitud de la cinta para grabar un caracter?

UNIDAD II ESTRUCTURAS DE DATOS ELEMENTALES

OBJETIVO GENERAL

El alumno conocerá las formas de representar, almacenar y manipular las estructuras de datos elementales.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Reconocerá qué es una estructura de datos elementales.
2. Reconocerá las formas de representar en la computadora caracteres, números enteros y números reales.
3. Reconocerá qué es un arreglo y los modos de almacenar un arreglo en la memoria.
4. Derivará funciones de mapeo y vectores de acceso para calcular la dirección de los elementos de un arreglo almacenado en forma contigua.
5. Escribirá algoritmos para calcular la dirección de los elementos de un arreglo almacenado en forma ligada.

INTRODUCCION

El material de las estructuras de datos se ha organizado para su estudio en: *Estructuras de datos elementales*, aquellos datos (números enteros, números reales, caracteres, arreglos, otros) cuya manipulación y representación se ha estandarizado en los lenguajes de programación, y *Estructuras de datos compuestos* (pilas, colas, gráficas, árboles, otros) cuya manipulación y representación requieren del ingenio de los usuarios.

En esta unidad, se estudiarán las estructuras de datos elementales.

II.1 GENERALIDADES

Una estructura de datos en su forma más general consiste de una colección de nodos o registros que mantienen importantes relaciones entre sí. (Véase figura II.1)

Estructuras de datos

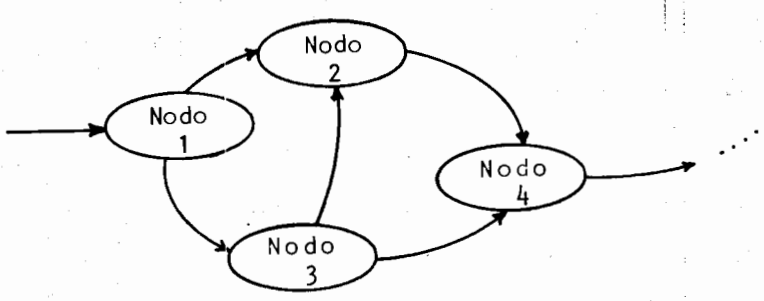


Figura II.1 Representación de una estructura de datos.

El nodo es el elemento básico para mantener la información en una estructura de datos. Para representar la información contenida en un nodo se pueden usar una o más palabras de computadora dependiendo de las características de los datos. Un nodo puede subdividirse en campos de tal manera que sea fácil la manipulación de la información. (Véase figura II.2).

Nodos o registros

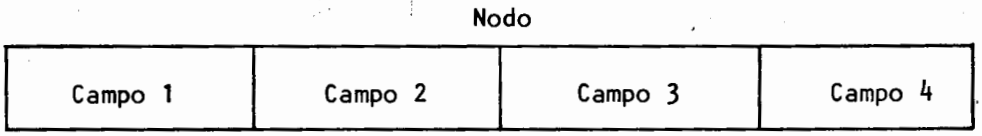


Figura II.2 Subdivisión de un nodo en campos.

Es común encontrar que el contenido de un campo es el de una estructura de datos elemental, esto es: un número entero, un número real, una cadena de caracteres, etc.

II.2 REPRESENTACION DE NUMEROS ENTEROS

Un número entero es cualquier número que pertenece al conjunto definido como:

Número entero

$$\text{Número entero} = \ldots\ldots -(n+1), -n \ldots\ldots -2, -1, 0, 1, 2, \ldots\ldots n, (n+1), \ldots\ldots$$

Cualquier número entero sin signo puede ser representado por una secuencia de dígitos de la forma:

$$dn\ dn\text{-}1 \ldots\ldots d_2 d_1 d_0$$

cuyo valor o magnitud es determinado por la suma de:

$$dnb^n + d_{n-1}b^{n-1} + \ldots\ldots + d_2b^2 + d_1b^1 + d_0b^0$$

donde b es la base o raíz del sistema.

Ejemplo II.1
La magnitud del número 124_{10} es la suma de los términos:

$$1 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$$

Para representar un número entero en la computadora se utiliza el sistema de numeración binaria o de base 2 debido a la tecnología que se utiliza para la memoria, la cual sólo permite el uso de dos dígitos: el cero y el uno.

Numeración binaria

Ejemplo II.2

La magnitud del número 10111_2 es la suma de los términos:

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Los números enteros de acuerdo a la definición pueden ser positivos o negativos, esto hace que para representarlos en la computadora se establezcan convenciones como pueden ser el método de signo y magnitud o el del complemento.

SIGNO Y MAGNITUD

Para representar un número con este método, se escribe el signo (+, -) precediendo a la magnitud del número.

Método de signo y magnitud

Para representarlo en la computadora generalmente se utiliza el dígito binario de la extrema izquierda para denotar el signo, y los restantes se utilizan para la magnitud. Consideremos el caso de utilizar cuatro dígitos binarios para la representación. En este caso tomamos un dígito para el signo y tres para la magnitud. (Véase la figura II.3).

Número positivo	Signo y magnitud	Número negativo	Signo y magnitud
+0	0 000	-0	1 000
+1	0 001	-1	1 001
+2	0 010	-2	1 010
+3	0 011	-3	1 011
+4	0 100	-4	1 100
+5	0 101	-5	1 101
+6	0 110	-6	1 110
+7	0 111	-7	1 111

Figura II.3 Representación con 4 dígitos binarios en signo y magnitud.

COMPLEMENTO

Para representar números enteros positivos y negativos en la computadora, se utiliza el método del complemento a la base $C = b^n - |\text{Número}|$, en donde C es la representación del número en n dígitos de la base b y |Número| es la magnitud del número negativo. Si usamos tres dígitos decimales, se pueden representar 1000 números sin signo entre 0 y 999. La idea consiste en asignar la mitad de este rango para los positivos y la otra mitad para los negativos, pudiendo entonces manejarse 1000 números con signo entre - 500 y + 499 considerando al cero positivo.

Método del complemento

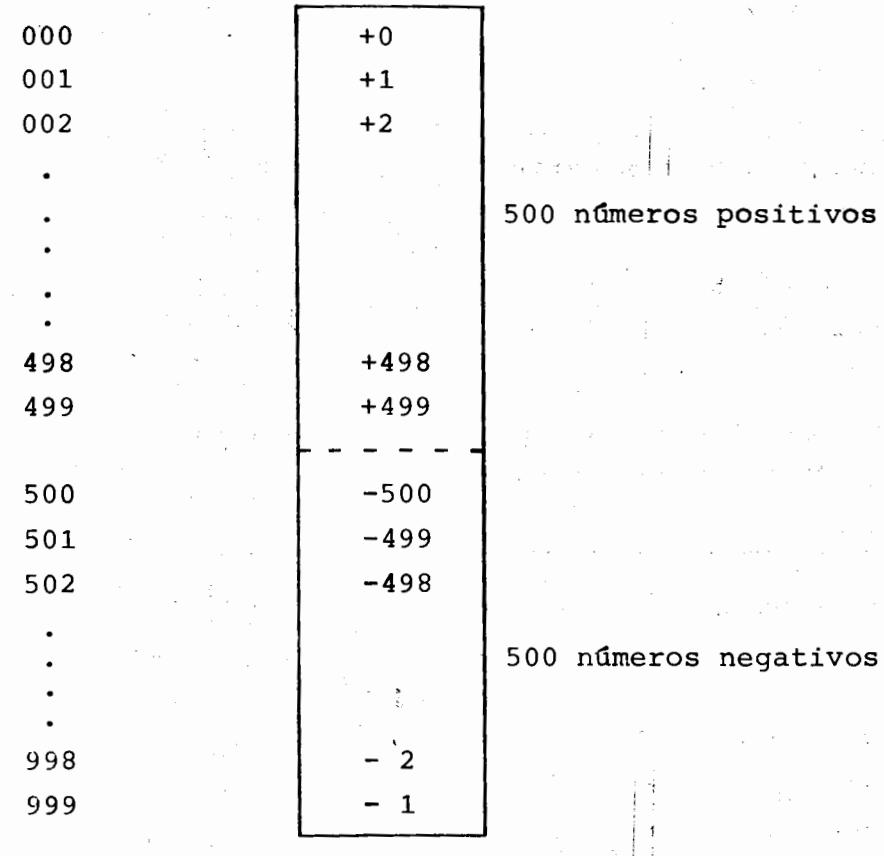


Figura II.4 Representación con 3 dígitos decimales de números positivos y negativos en complemento a 10.

COMPLEMENTO A DOS

Debido a que los números que se manejan en la computadora utilizan una base binaria, el complemento a 2 de este tipo de números nos interesa para representaciones de los números negativos.

Complemento a dos

Complemento a 2 = 2^n - |número| donde número está en base 2 y n es la cantidad de dígitos binarios para la representación.

Ejemplo II.3
Representar el número 5 como negativo en complemento a 2 utilizando 4 dígitos binarios:

$$5_{10} = 0101_2$$
$$-5_{10} = 10000_2 - 0101_2 = 1011_2$$

También el complemento a 2 puede ser calculado cambiando los ceros por unos y los unos por ceros y sumando a esto la unidad.

Ejemplo II.4
Representar el número 5 como negativo en complemento a 2 utilizando 4 dígitos binarios:

$$5 = 0101_2$$
Invirtiendo $= 1010_2$
Sumando 1 $= 1011_2$

Es posible conocer el alcance de representación con cuatro bits, utilizando la fórmula anterior. (Véase la tabla II.1).

Número positivo	Complemento a 2	Número negativo	Complemento a 2
+ 0	0 000		
+ 1	0 001	- 1	1 111
+ 2	0 010	- 2	1 110
+ 3	0 011	- 3	1 101
+ 4	0 100	- 4	1 100
+ 5	0 101	- 5	1 011
+ 6	0 110	- 6	1 010
+ 7	0 111	- 7	1 001
		- 8	1 000

Tabla II.1 Representación con 4 dígitos binarios de números positivos y negativos enteros en complemento a 2.

El complemento a 2 elimina la duplicidad en la representación del cero, lo cual no sucede en la representación de signo y magnitud.

La ventaja del complemento a 2 es la facilidad para realizar operaciones aritméticas.

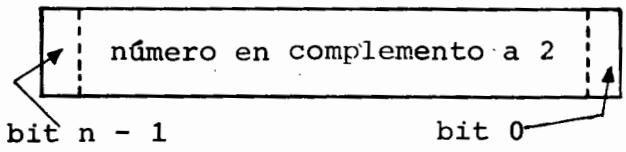
Ejemplo II.5
Calcular 3 + 4 y 3 - 4:

a) $3 + 4 = (0011)_2 + (0100)_2 = (0111)_2 = 7$

b) $3 - 4 = (0011)_2 +$ complemento a 2 de $(0100)_2$

$= (0011)_2 + (1100)_2 = (1111)_2 = -1$

El formato para un número entero en complemento a 2, usando una palabra de n bits para su almacenamiento es el indicado en la siguiente figura.



Formato para un número entero

Figura II.5

En el caso del complemento a 2 todos los bits forman el número, pero adicionalmente el bit de la extrema izquierda proporciona información respecto al signo del número representado, cero igual a positivo y uno igual a negativo.

II.3 REPRESENTACION DE NUMEROS REALES

Un número real, en cualquier base (R), puede ser representado como:

$$A_R = \pm (A_n\ A_{n-1}\ A_{n-2} \ldots A_1 A_0 \cdot A_{-1}\ A_{-2} \ldots A_{-(m+1)} A_{-m})_R$$ Número real

Esta notación llamada de punto fijo es suficiente para representar cualquier número real dentro de la computadora, pero existen algunas aplicaciones que involucran el manejo de cantidades como: .000000000082 ó 18 000 000 000 000, las cuales requerirían de una gran cantidad de bits para su representación. Por ejemplo, para obtener una precisión de 27 dígitos decimales, se requiere de aproximadamente 90 dígitos binarios.

Punto fijo

Un método alterno que facilita la manipulación de los números reales es la notación científica o de punto flotante.

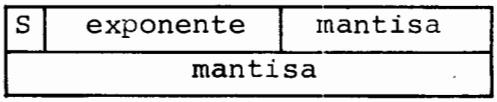
La forma general de un número real de base R en notación flotante es:

$$\pm \cdot f_{-1}\ f_{-2}\ f_{-3} \ldots f_{-m} \times R^{\pm E}$$

donde $f_{-1}\ f_{-2}\ f_{-3} \ldots f_{-m}$ es llamada la parte fraccionaria o mantisa y E es un número entero llamado el exponente y R la base.

Notación científica o de punto flotante

Para esta representación, el formato más usual en los lenguajes de programación es el que se muestra en la figura II.6.



S = Signo de la mantisa

Exponente = exponente (como número entero en complemento a 2).

Mantisa = mantisa (como número entero en magnitud y signo).

Formato para un número real

Figura II.6 Formato para el almacenamiento de números reales.

El número de bits para el exponente y la mantisa depende de la magnitud y de la precisión respectivamente de los números a representar. En el software que provee el fabricante, el número de bits ya está definido y se especifica claramente la magnitud de los números que pueden ser representados y la precisión que se logra.

Debido a esta limitante la diferencia que se tiene entre el valor aproximado de representación y el valor original es llamado error de truncamiento o de redondeo, dependiendo de la estrategia que se siga.

Error de truncamiento o de redondeo

Para aumentar la precisión de un número, es posible asignar más palabras de computadora para la mantisa, llamado en este caso número de doble precisión. Un formato para este tipo de representación es el mostrado en la figura II.7.

Números reales de doble precisión

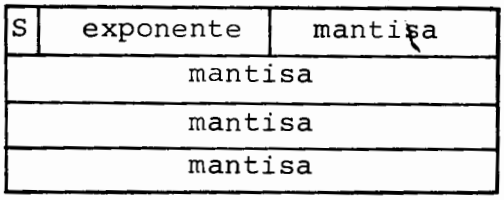


Figura II.7 Formato para el almacenamiento de números de doble precisión.

Otros tipos de datos como los números complejos y los valores lógicos son también manipulados en la computadora. Un número complejo es de la forma a + bj, donde a y b son números reales y j es la raíz de - 1 . Para representarlo en la computadora un formato posible es el siguiente:

Números complejos

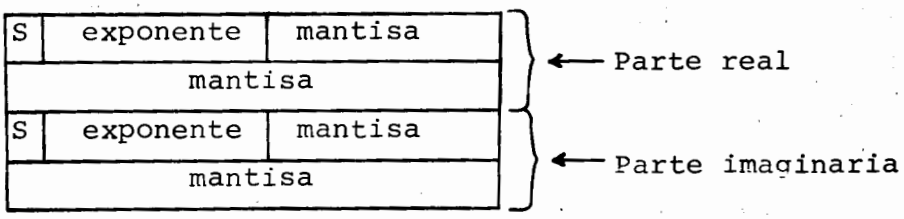


Figura II.8 Formato para representar números complejos.

El valor lógico falso o verdadero que toma una variable lógica, puede ser representado en la computadora utilizando los siguientes formatos:

Valores lógicos

todos los bits apagados	falso
todos los bits prendidos	verdadero

Figura II.9 Formato para representar datos lógicos.

II.4 REPRESENTACION DE CARACTERES

Para facilitar la escritura y lectura de los programas y datos de una computadora, se desarrollaron códigos simbólicos para representar la información con caracteres en lugar de números.

Cada caracter es representado por un patrón de bits diferente que lo identifica; el número de bits para representar un caracter está determinado por el código utilizado. Por ejemplo, los códigos de 64, 128 y 256 caracteres pueden representarse con 6, 7 y 8 bits respectivamente.

Caracteres

Si 8 bits (un byte) son usados para representar cada caracter y la longitud de la palabra es mayor, es ineficiente almacenar un caracter por palabra. En este caso, varios caracteres son empacados en una palabra. En el caso de algunas máquinas con longitud de palabra de 16 bits existen dos posibilidades: Almacenar un solo caracter y desperdiciar 8 bits llamado formato A1 ó almacenar dos caracteres por palabra llamado formato A2. (Ver figura II.10).

Byte

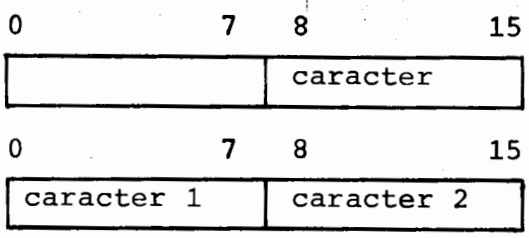


Figura II.10 Almacenamiento de caracteres con formato A1 y A2

Algunos de los códigos más importantes que actualmente se emplean para la representación de letras, números y símbolos especiales son el de la American Standard Code for Information Interchange (ASCII), el cual requiere 8 bits para representar cualquier caracter; el Binary Coded Decimal (BCD), que requiere de 6 bits para representar cualquiera de sus 64 caracteres y el Extended BCD Interchange Code (EBCDIC) el cual también es otro importante código de 8 bits.

La tabla II.2, muestra la representación de letras, números y caracteres especiales para los códigos BCD, EBCDIC y ASCII.

II.5 REPRESENTACION DE ARREGLOS

Una de las estructuras de datos y posiblemente la más conocida y utilizada por los programadores es el arreglo.

Arreglos

Los lenguajes de programación de alto nivel facilitan el definir y operar los arreglos utilizando instrucciones como: dimensión, integer, real en FORTRAN, o Array en ALGOL que permiten definir el número de nodos y el formato para representar los datos.

Caracter	BCD	EBCDIC	ASCII
blanco	110 000	0100 0000	0010 0000
(	111 100	0100 1101	0010 1000
+	010 000	0100 1110	0010 1011
$	101 011	0101 1011	0010 0100
*	101 100	0101 1100	0010 1010
)	011 100	0101 1101	0010 1001
-	100 000	0110 0000	0010 1101
/	110 001	0110 0001	0010 1100
.	111 011	0110 1011	0010 1111
=	001 011	0111 1110	0011 1101
A	010 001	1100 0001	0100 0001
B	010 010	1100 0010	0100 0010
C	010 011	1100 0011	0100 0011
D	010 100	1100 0100	0100 0100
E	010 101	1100 0101	0100 0101
F	010 110	1100 0110	0100 0110
G	010 111	1100 0111	0100 0111
H	011 000	1100 1000	0100 1000
I	011 001	1100 1001	0100 1001
J	100 001	1101 0001	0100 1010
K	100 010	1101 0010	0100 1011
L	100 011	1101 0011	0100 1100
M	100 100	1101 0100	0100 1101
N	100 101	1101 0101	0100 1110
O	100 110	1101 0110	0100 1111
P	100 111	1101 0111	0101 0000
Q	101 000	1101 1000	0101 0001
R	101 001	1101 1001	0101 0010
S	110 010	1110 0010	0101 0011
T	110 011	1110 0011	0101 0100
U	110 100	1110 0100	0101 0101
V	110 101	1110 0101	0101 0110
W	110 110	1110 0110	0101 0111
X	110 111	1110 0111	0101 1000
Y	111 000	1110 1000	0101 1001
Z	111 001	1110 1001	0101 1010
0	000 000	1111 0000	0011 0000
1	000 001	1111 0001	0011 0001
2	000 010	1111 0010	0011 0010
3	000 011	1111 0011	0011 0011
4	000 100	1111 0100	0011 0100
5	000 101	1111 0101	0011 0101
6	000 110	1111 0110	0011 0110
7	000 111	1111 0111	0011 0111
8	001 000	1111 1000	0011 1000
9	001 001	1111 1001	0011 1001

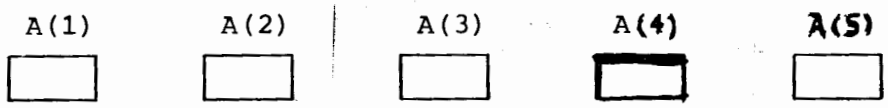
Tabla II.2 Códigos

II.5.1 DEFINICION Y OPERACIONES

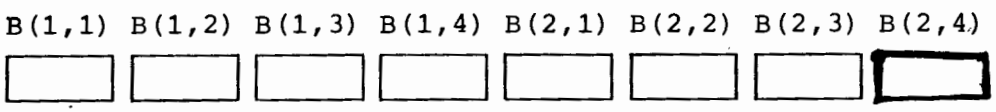
Un arreglo es una estructura con un número fijo de nodos. Al conjunto de nodos se le identifica con un nombre y a los nodos con un índice. Este concepto de arreglo es tomado de lo que en matemáticas se entiende como vector o matriz.

Ejemplo II.6
Para manipular un arreglo en lenguaje fortran es muy común el uso de la declaración DIMENSION, seguida del nombre del arreglo y, entre paréntesis, el número de nodos del arreglo.

a. DIMENSION A(5) se refiere a un conjunto de nodos formado por:



b. DIMENSION B(2,4) se refiere a un conjunto de nodos, formado por:



En un arreglo, se pueden practicar todas las operaciones relativas a la información contenida en los nodos (leer o escribir), pero no aquellas operaciones relativas a alterar el número de los nodos (suprimir o agregar). Hay que hacer notar, sin embargo, que algunos lenguajes como ALGOL permiten redefinir el número de nodos de los arreglos.

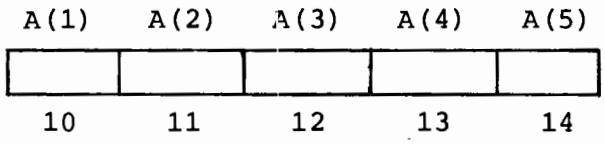
II.5.2 ALMACENAMIENTO Y RECUPERACION

Ya sabemos que las localidades de memoria básicamente pueden organizarse de forma contigua y ligada, un arreglo, también puede organizarse de cualquiera de estas dos formas: organización contigua y organización ligada.

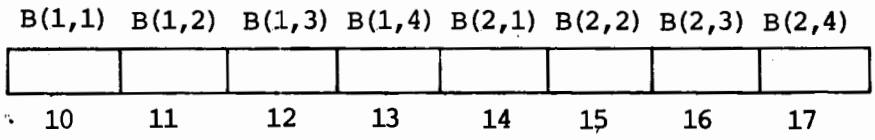
ORGANIZACION CONTIGUA

Los nodos del arreglo, ocupan localidades contiguas de la memoria.

a) A(i = 1,5)



b) B(i = 1,2; j = 1,4)



Formas de recuperación de los elementos de un arreglo

Cuando los nodos de los arreglos se organizan en la memoria de forma contigua, la recuperación de cualquiera de los nodos se hace básicamente sobre dos esquemas:

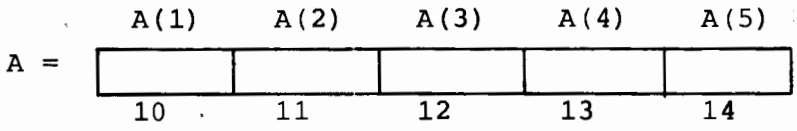
- Funciones de mapeo de relación 1 a 1
- Tablas de recuperación.

FUNCIONES DE MAPEO DE RELACION 1 A 1

Una función de mapeo de relación 1 a 1, es una expresión que permite calcular la dirección de un nodo en función de sus índices.

Ejemplo II.7

a) Consideremos que el arreglo A es almacenado en forma contigua.



Una expresión que nos permite llegar a cualquier nodo A(i) es:

$$DIR\ A(i) = 9 + i$$

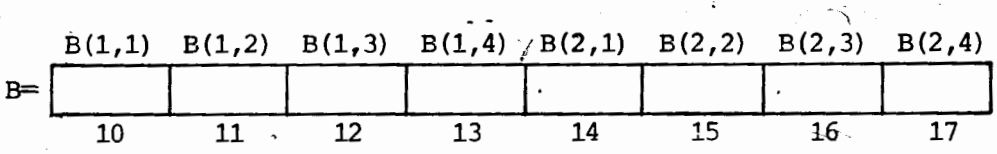
para: A(3) = 9 + 3 = 12

de forma general:

DIR A(i) = BA + i

donde **BA** es una dirección base.

b) Consideremos que B está organizado en forma contigua.



Si observamos el orden en que están los nodos de A podemos decir que el almacenamiento de los elementos es por renglones; el primer índice se mantiene constante y el segundo es el que varía; en algunos casos el orden de almacenamiento puede ser por columnas.

Almacenamiento de un arreglo por renglones

Una expresión que permite llegar a cualquier elemento B(i,j) del arreglo es:

DIR B(i,j) = 9 + 4(i - 1) + j

para:

DIR B(2,3) = 9 + 4(2 - 1) + 3 = 16

de forma general:

DIR B(i,j) = BA + r (i - 1) + j

donde BA es una dirección base y r es el número de columnas.

Existen arreglos que por las características de sus elementos reciben diversos nombres. Por ejemplo, los arreglos llamados simétricos en los cuales A(i,j) = A(j,i), los arreglos triangulares en los que los elementos arriba o abajo de la diagonal principal son cero.

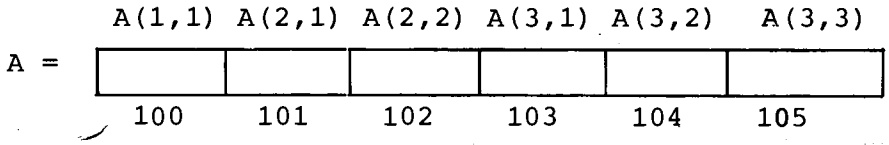
Arreglo simétrico

Una función de mapeo asociada a este tipo de arreglos ofrece un ahorro considerable de las localidades de memoria.

Ejemplo II.8

Sea A una matriz triangular inferior y que ésta se almacena por renglones:

$$A = \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



si BA es la localidad base BA = 99, una función de mapeo para cualquier A(i, j) es:

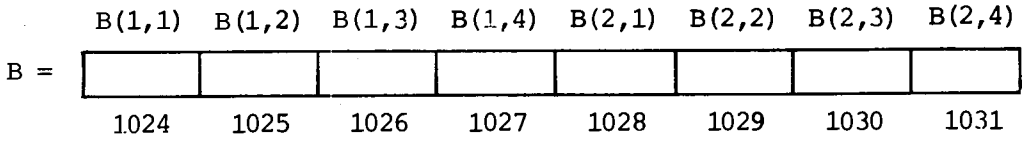
$$\text{DIR } A(i,j) = BA + \frac{i(i-1)}{2} + j$$

TABLAS O VECTORES DE ACCESO

Algunas funciones de mapeo, por las operaciones que involucran, ocasionan una sobrecarga durante el cálculo de la dirección, un método alterno para la recuperación de los nodos de un arreglo y que reduce esta sobrecarga, consiste en asociar al arreglo un vector que contiene las localidades de inicio de cada renglón o columna.

Ejemplo II.9

Consideremos a B organizada en forma contigua:



el vector asociado a B sería:

T = 1023, 1027

de esta forma, la expresión para el cálculo de la dirección de cualquier localidad B(i,j) sería:

$$DIR\ B(i,j) = T(i) + j$$

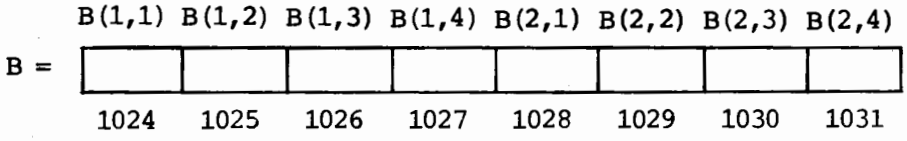
para:

$$B(2,3) = T(2) + 3 = 1027 + 3 = 1030$$

Algunos algoritmos para resolver un conjunto de ecuaciones lineales simultáneas, requieren intercambiar las columnas para prevenir pivotes pequeños que ocasionen errores de redondeo. El uso de las tablas de acceso para intercambiar renglones es muy eficiente, ya que solamente bastará intercambiar los elementos del vector.

Intercambio de columnas o de renglones

Ejemplo II.10

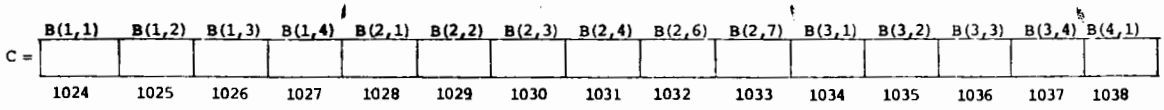


El vector asociado a B es T = 1023, 1027 si deseamos intercambiar los renglones bastará hacer T = 1027, 1023.

Un arreglo dentado, es aquel que tiene un número variable de elementos en cada renglón. El uso de una tabla de acceso, permite una aplicación sencilla de este concepto, así como de un ahorro considerable de memoria.

Arreglo dentado

Ejemplo II.11



la tabla asociada será T = 1023, 1027, 1033, 1037

ORGANIZACION LIGADA

En este caso, los elementos del arreglo, ocupan cualquier

localidad de la memoria y la relación entre ellas está definida por el campo de liga. Para recuperar cualquier elemento del arreglo se requiere de algún procedimiento, que puede ser iterativo o que utilice alguna función de mapeo, esto depende de la forma de almacenamiento la cual puede ser parcial o totalmente ligada. Una organización parcialmente ligada para B(m,n) es la de la figura II.11.

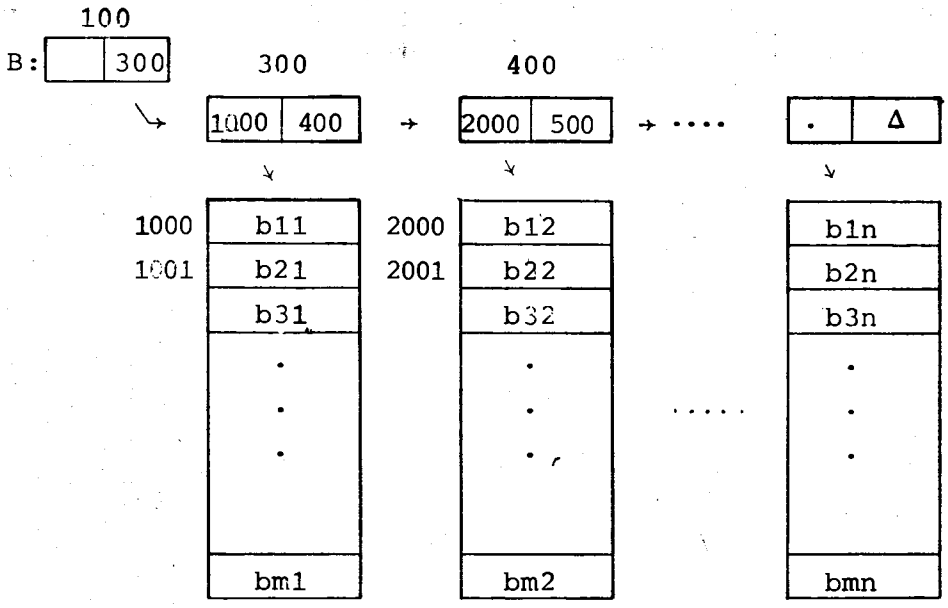


Figura II.11 Almacenamiento parcialmente ligado para un arreglo.

Un algoritmo* para llegar a cualquier B(i,j) de una estructura parcialmente ligada sería:

```
AB = DIR(B)
DESDE M=1 HASTA J
      AB = LIGA DERECHA (AB)
FIN
DIR (B(I,J)) = LIGA IZQUIERDA (AB) + I - 1
```

* Todos los algoritmos utilizaron las estructuras de control que se especifican en el apéndice de esta obra.

Ejemplo II.12 Recuperar B(2,2)J = 2 e i = 2

AB = 100
AB = LIGA DERECHA (AB) = 300
AB = LIGA DERECHA (AB) = 400
DIR = B(2,2) = LIGA IZQUIERDA (400) + 2 - 1 = 2001

Una organización totalmente ligada para el arreglo B(2,4) sería:

$$B = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

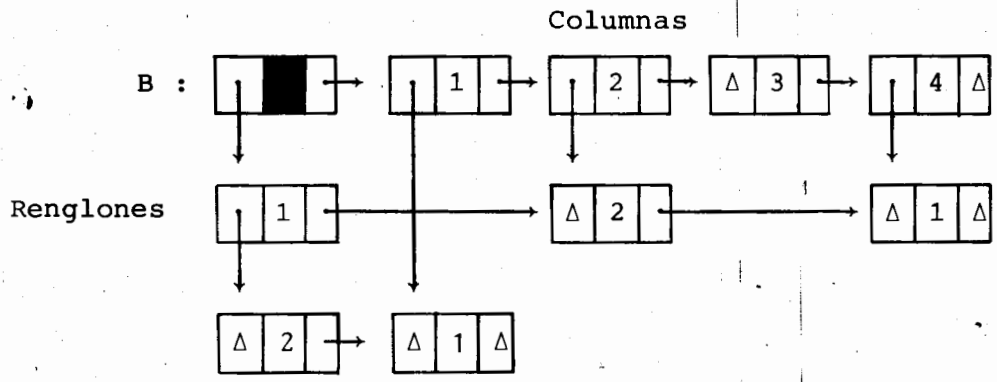


Figura II.12 Almacenamiento totalmente ligado para un arreglo de dos dimensiones.

Una representación de este tipo resulta conveniente para un arreglo esparcido, el cual tiene la mayoría de sus elementos iguales a cero. Para decir que un arreglo es esparcido, puede considerarse que el 90 % o más de los elementos del arreglo son cero.

Arreglo esparcido

Existen muchos problemas en los que se involucra el uso de matrices esparcidas. Solución de ecuaciones diferenciales, representación de gráficas, etc.

El uso de una representación directa, resultaría muy ineficiente, ya que estaríamos almacenando y operando los elementos iguales a cero. El uso de una representación ligada es una alternativa para lograr una manipulación eficiente de la información.

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

1.	Complemento a 2	( )	Consiste en asignar un dígito para representar el signo y los demás para la magnitud.
2.	Número entero	( )	Colección de nodos o registros que mantienen importantes relaciones entre sí.
3.	Estructuras de datos	( )	Conjunto de números definido como: $\ldots -(n+1), -n, \ldots -2, -1, 0, 1, 2, \ldots, n, (n+1), \ldots$
4.	Método de signo y magnitud	( )	Se calcula como $C=2^n - \lvert \text{número} \rvert$.
5.	Notación científica	( )	Con ellos, se facilita la escritura y lectura de programas y datos.
6.	Código ASCII	( )	Código de 8 bits para representar caracteres.
7.	Caracteres	( )	La representación consiste de una parte fraccionaria llamada mantisa y de un exponente.
8.	Arreglo	( )	Se utilizan para calcular la dirección de almacenamiento de un elemento de un arreglo.
9.	Organización contigua	( )	Conjunto de nodos que no puede ser aumentado, ni disminuido.
10.	Funciones de mapeo	( )	Tipo de organización de la memoria donde se utilizan números de direcciones de la forma k, k+1, k+2, k+3, ...

II. RESOLVER LOS SIGUIENTES PROBLEMAS

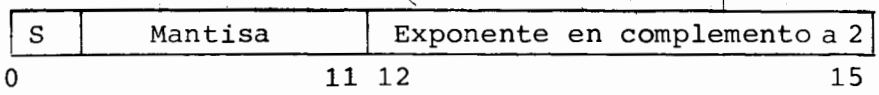
1. Calcular la magnitud en base 10 de los siguientes números.

a. 1023_{10} b. 327_8 c. 111001_2

2.- Usando el método de complemento a 2, represente todos los números enteros positivos y negativos que pueda manejar con 5 bits.

3. Calcular usando 8 bits y complemento a 2 las operaciones (-7 + 2) y (3 - 5).

4. Un número real tiene el siguiente formato:



a. ¿Cuál es el mayor número que se puede representar en base 10?

b. ¿Cuál es el número positivo más pequeño distinto de cero?

c. ¿Cuál es el menor número?

d. ¿Cuál es la precisión de la mantisa en dígitos decimales?

5. ¿Cuál es el código ASCII y EBCDIC para los símbolos 1, 9, A, B y C?

6. Derivar una función de mapeo para una matriz simétrica almacenada por renglones.

7. Derivar la función de mapeo para una matriz triangular superior almacenada por renglones.

8. Construya para una matriz dentada una función que involucre una tabla de acceso para recuperar cualquiera de los elementos.

9. Para una organización totalmente ligada como la de la figura II.12, escriba un algoritmo para recuperar cualquier elemento B(i, j).

UNIDAD III ESTRUCTURAS DE DATOS COMPUESTAS: LISTAS LINEALES

OBJETIVO GENERAL

El alumno aplicará las formas de representar y operar en la computadora las principales listas lineales.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Identificará los conceptos de lista lineal, cola, cola doble, lista circular y lista doblemente ligada.
2. Realizará las operaciones que se practican sobre una pila, cola y cola doble.
3. Escribirá algoritmos para las operaciones en una pila, en una cola y en una cola doble almacenadas en forma contigua y ligada.
4. Escribirá algoritmos para las operaciones de una cola circular contigua, almacenada en forma ligada.
5. Escribirá algoritmos para las operaciones de una lista doblemente ligada y circular doblemente ligada.
6. Resolverá problemas en los que se involucran la utilización de las listas lineales.

INTRODUCCION

Un arreglo, como se ha visto anteriormente, está constituido por un conjunto fijo de nodos y, en algunos lenguajes de programación, puede dárseles un manejo dinámico que permita aumentar o disminuir el número de nodos.

Las listas, nombre que recibe un conjunto de nodos que puede ser aumentado o disminuido, vienen a darle mayor flexibilidad de programación a los usuarios, con un ahorro considerable de memoria por las operaciones que pueden practicarse sobre ellas.

La unidad comprende el estudio de un tipo de lista: las listas lineales, cuya representación y utilización serán ejemplificadas con la pila, la cola y la cola doble.

Las listas no lineales se dejan para su estudio en la siguiente unidad.

III.1 GENERALIDADES

Una lista es una estructura de datos que tiene un número variable de nodos. Una lista lineal, es una lista cuyos nodos están ordenados por un solo criterio, en donde el último y el primer nodo no tienen sucesor y antecesor respectivamente. Una lista lineal formalmente puede ser definida como un conjunto de nodos X(1), X(2), ... X(n), cuyas propiedades estructurales esencialmente involucran relaciones en una sola dimensión entre sus nodos. Por ejemplo, X(1) es el primer nodo, X(n) es el último nodo y el nodo X(k) es precedido por X(k - 1) y le sigue el X(k + 1). Véase figura III.1.

Lista lineal

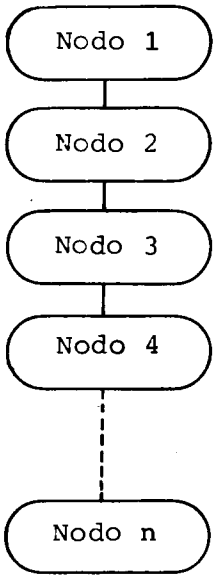


Figura III.1 Representación gráfica de una lista lineal.

III.2 PILA

III.2.1 DEFINICIONES Y OPERACIONES

Una pila o stack es una estructura de datos lineal, en la cual las operaciones se realizan por uno de los extremos de la lista.

Modelo de una pila

El modelo propuesto para su representación es una implementación ferroviaria como la de la figura III.2 que en el argot ferrocarrilero se llama i griega.

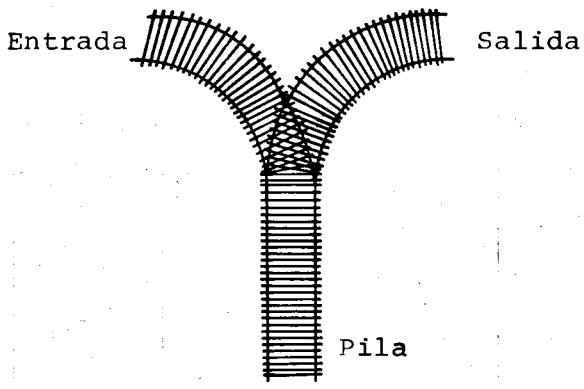
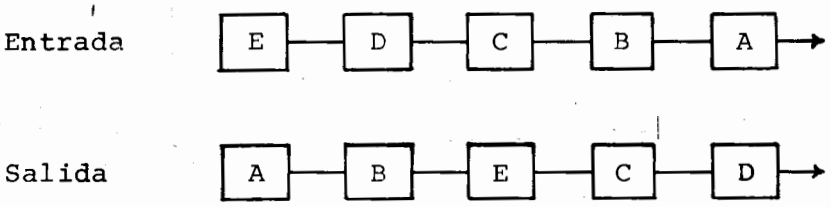


Figura III.2 Implementación ferroviaria en forma de i griega, como modelo de una pila.

Ejemplo III.1

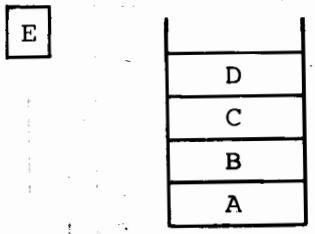
Para ilustrar la forma como una pila trabaja consideremos los cinco carros de ferrocarril A, B, C, D y E parados en la entrada de la i griega y supongamos que el convoy requiere de un nuevo orden de carros D, C, E, B y A.



Usando la i griega de la siguiente forma podremos obtener la salida deseada, de acuerdo a los siguientes pasos:

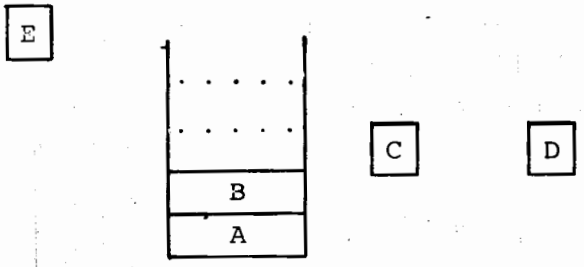
Paso 1

Meter los carros A, B, C, D.
En forma gráfica quedaría:



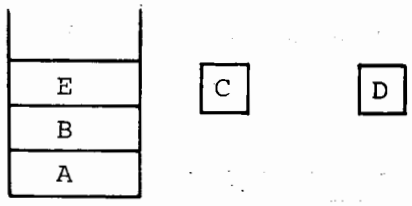
Paso 2

Sacar los carros D y C, de esta forma la i griega queda:



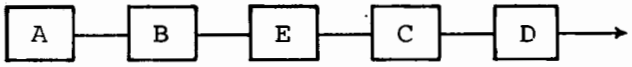
Paso 3

Meter el carro E a la i griega:

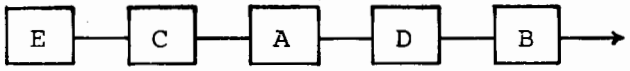


Paso 4

Sacar los carros E, B y A con lo cual obtenemos la secuencia deseada.



Hay que hacer notar que no todas las secuencias son posibles de obtener en una sola pasada; por ejemplo, la secuencia:



ya que no es posible sacar "A" antes que "C".

Por la forma en que se agregan y retiran elementos de la pila, el método ha sido llamado UEPS (últimas entradas primeras salidas) en inglés LIFO (last input first output). Esto significa que solamente puede ser retirado de la pila el último elemento agregado. En el ejemplo III.1, en el paso 2, el elemento D puede salir ya que es el último agregado.

Ultimas entradas primeras salidas

En esta estructura las operaciones de agregar y retirar reciben en el campo de la computación los nombres de PUSH y POP respectivamente.

Push y Pop

III.2.2 REPRESENTACION Y ALGORITMOS DE LAS OPERACIONES

Una pila puede ser representada en la computadora eligiendo localidades contiguas de memoria de acuerdo a como se muestra en la figura III.3

Almacenamiento contiguo de una pila

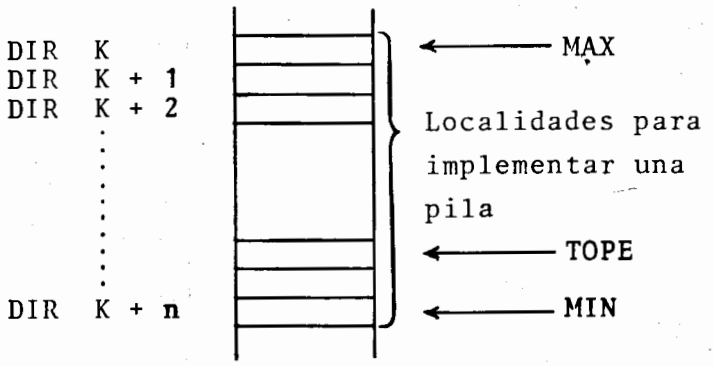


Figura III.3 Pila almacenada en localidades contiguas.

Para indicar la localidad donde se agregará un nuevo elemento a la pila o de donde se retirará el último elemento insertado, existe asociado a la estructura un apuntador T para implementar estas dos operaciones.

Debido a que la pila no puede ser infinita, esto es, que el número de localidades para implementarla es fijo, deberá verificarse el no exceder estos límites. Por esto, antes de agregar un nuevo elemento, hay que verificar si existe lugar para él, si no existe, habrá un error en la operación de la pila, conocido como pila llena (overflow) así como verificar en la operación retiro si existe un elemento por retirar, si no, habrá un error de pila vacía (underflow).

Pila llena (Overflow)

Pila vacía (Underflow)

ALGORITMOS DE LAS OPERACIONES DE AGREGAR Y RETIRAR

Para agregar un nuevo elemento a la pila, un algoritmo que realiza la operación es el siguiente:

```
SI T=MAX
   ESCRIBE ('PILA LLENA (OVERFLOW)')
OSI T=Δ (NULO O VACIO)
   T=MIN
   PILA(T)=DATO
OBIEN
   T=T - 1
   PILA(T) = DATO
FIN
```

Para retirar un elemento de la pila, un algoritmo que representa a la operación es el siguiente:

```
SI T=Δ
   ESCRIBE ('PILA VACIA (UNDERFLOW)')
OSI T=MIN
   DATO=PILA(T)
   T=Δ
OBIEN
   DATO=PILA(T)
   T=T + 1
FIN
```

MAX es el límite superior y MIN es el límite inferior, estos apuntadores delimitan el área de localización de la pila.

Almacenamiento ligado de una pila

Una pila también puede ser representada usando localidades ligadas de memoria de acuerdo a como se muestra en la figura III.4.

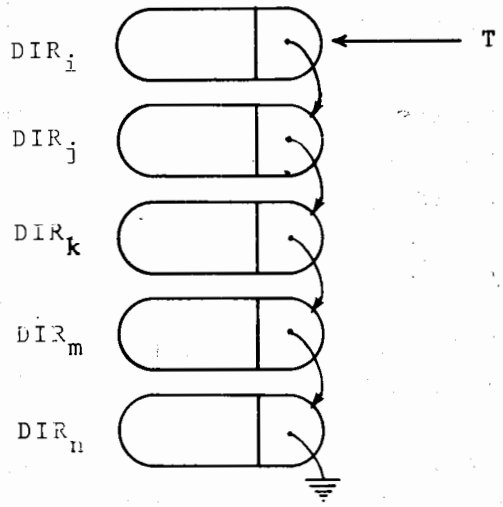


Figura III.4 Pila almacenada en localidades ligadas.

Las localidades para cada uno de los elementos de la pila son cualesquiera, pero el orden de la pila está definido por T y las ligas de la estructura. El apuntador T se usa para indicar la posición del último elemento agregado.

ALGORITMOS DE LAS OPERACIONES DE AGREGAR Y RETIRAR

Para agregar un nuevo elemento a una pila se requiere en forma general:

1) Obtener la dirección del nuevo elemento N y
2) agregarlo al Stack

En forma gráfica el proceso puede ser ejemplificado de la siguiente manera:

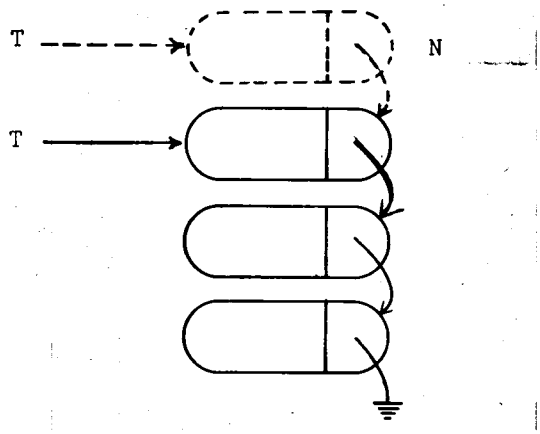


Figura III.5 Representación gráfica de la operación de agregar.

Un algoritmo para representar la operación es el siguiente:

```
OBTENER N
SI T=Δ
   LIGA(N)=Δ
   DATOP(N)=DATO
   T=N
OBIEN
   LIGA(N)=T
   DATOP(N)=DATO
   T=N
FIN
```

En caso de no poder obtener una localidad N disponible, - se dice que la pila está llena y se ha alcanzado una condición o señal de overflow.

Para retirar un elemento de una pila se requiere en forma general de:

1.- Retirar el nodo apuntado por T y
2.- Retornarlo como disponible.

En forma gráfica el proceso puede ser ejemplificado de la siguiente forma:

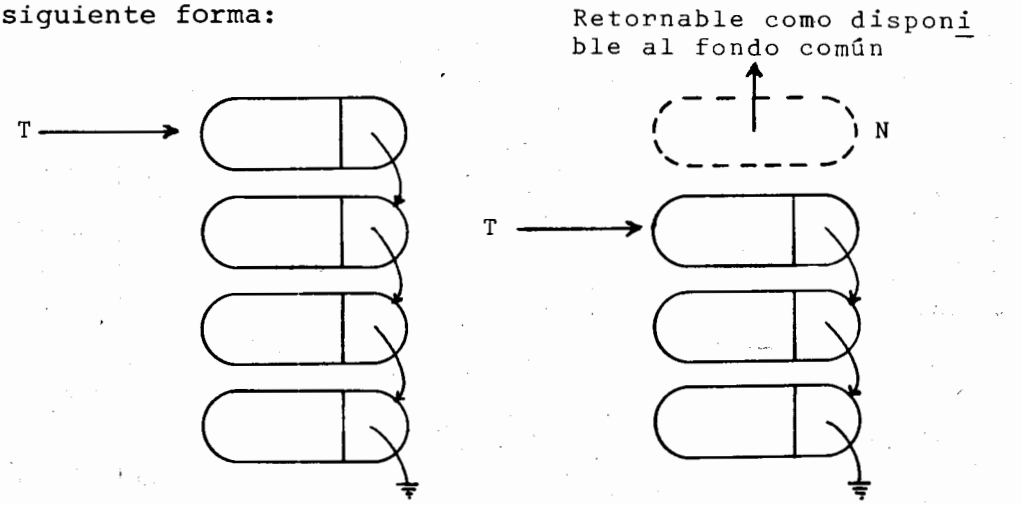


Figura III.6 Representación gráfica de la operación de retirar.

Un algoritmo que retira un elemento de una pila es el siguiente:

```
SI T=Δ
   ESCRIBIR ('PILA VACIA')
OBIEN
   DATO=DATOP(T)
   N=T
   T=LIGA(T)
FIN

RETORNAR  N
```

Ejemplos:

a) Recursividad.- La recursividad, es una formulación que permite definir un concepto en términos propios.

Existen muchos problemas cuya descripción algorítmica resulta mejor si se hace de manera recursiva en lugar de iterativa.

Consideremos la función factorial definida recursivamente como:

$$\text{FACTORIAL}(N) = \begin{cases} 1, & \text{si } N = 0 \\ N * \text{FACTORIAL}(N-1) & \text{si } N \neq 0 \end{cases}$$

Una facilidad con la que cuentan los programadores en un lenguaje de programación, son los procedimientos (Funciones o subrutinas). Un procedimiento que contiene una llamada a sí mismo, o una llamada a un procedimiento que llama a otro procedimiento que eventualmente llama al primero es conocido como un procedimiento recursivo.

Un procedimiento recursivo en su forma más general contiene los siguientes pasos:

1) Se guardan los parámetros, las variables locales y la dirección de retorno.
2) Si el criterio base ha sido alcanzado, se practica el cálculo final y se va al punto (3), si no se practica un cálculo intermedio y se regresa al punto (1) (se inicia la llamada recursiva).
3) Restaura los más recientes parámetros guardando las variables locales y la dirección de retorno. Va a la dirección de retorno.

Procedimiento recursivo

La característica UEPS de un procedimiento recursivo sugiere que una pila es la estructura más obvia a usarse para implementar los pasos (1) y (3) del algoritmo. Para cada llamada del algoritmo se agregan (PUSH) los valores actuales de las variables del procedimiento y para cada retorno se retiran (POP) los valores guardados en la llamada anterior.

Para ilustrar el mecanismo consideremos un algoritmo para el cálculo del factorial.

ALGORITMO FACTORIAL

Dado un entero N, el algoritmo evalúa N!. Para la implementación del algoritmo se utiliza una pila llamada A, para guardar en cada llamada el valor actual de N y la dirección de retorno DIR en el algoritmo. TOPE es el apuntador a la pila A.

1.- Se salvan N y la dirección de retorno en A.-
PUSH(A, TOPE, N, DIR)

2.- Se verifica el criterio BASE.

Si n = 1
entonces FACTORIAL = 1 e ir al paso (4)
si no hacer N = N-1 y DIR = dirección e
ir al paso (1).

3.- Cálculo del Factorial.- FACTORIAL = FACTORIAL* N

4.- Restaura el valor anterior de N y la dirección.-
POP(A, TOPE, N, DIR)

Ejemplo III.2

Rastreo del algoritmo factorial para n = 3:

Paso 1: Se salva N y dirección de retorno

3	P3

Paso 2: es n=1 no entonces n=n-1 y DIR=P3 ir al paso 1.

Paso 1: Se salva N y dirección de retorno

2	P3
3	P3

Paso 2: es N=1 no entonces n=N-1 y DIR= P3 ir al paso 1.

Paso 1: se salva N y dirección de retorno

1	P3
2	P3
3	P3

Paso 2: es N=1 si entonces FACTORIAL=1 ir al paso 4.

Paso 4: hace N=1 y DIR=P3 ir al paso 3

Paso 3: FACTORIAL = 1x1=1

Paso 4: hace N=2 y DIR=P3 ir al paso 3

Paso 3: FACTORIAL = 1x2=2

Paso 4: hace N=3 y DIR=P3 ir al paso 3

Paso 3: FACTORIAL= 2x3=6

Paso 4: pila vacía.

Para la programación de un algoritmo recursivo se debe considerar si el software del equipo permite que un procedimiento pueda llamarse a sí mismo, si éste es el caso, un programa para calcular el factorial sería:

```
C    PROGRAMA PARA EL CALCULO DE N!. EL PROGRAMA
C    ES RECURSIVO Y SE APOYA EN EL SOFTWARE DEL EQUIPO
C
     COMPILER DOUBLE PRECISION
C
5    CONTINUE
     ACCEPT "PARA QUE NUMERO QUIERES EL FACTORIAL?", N
     IF(N.LE.Ø) GO TO 1Ø
C
     FACT=N
     CALL FACT2(N,FACT)
C
     TYPE "EL FACTORIAL ES IGUAL A", FACT
     GO TO 5
1Ø   CONTINUE
     STOP
     END
R
C    SUBRUTINA PARA EL CALCULO DEL FACTORIAL
C
     COMPILER DOUBLE PRECISION
     SUBROUTINE FACT2(N,FACT)
     N=N - 1
     FACT=FACT*N
     IF(N.GT.1) CALL FACT2(N,FACT)
     RETURN
     END
```

Si el software no permite llamadas recursivas, será el programador quien tendrá que manejar la pila para implementar el procedimiento de acuerdo a la forma general planteada anteriormente. Un ejemplo de esto sería:

```
TYPE FACTORIAL
C    PROGRAMA PARA EL CALCULO DE N!.  EL PROGRAMA
C    ES RECURSIVO CON APOYO EN UNA PILA
C
     COMPILER DOUBLE PRECISION
     INTEGER PILA (2,1ØØØ)
     INTEGER TOPE,DIR
```

```
C
C   INICIALIZACION DE VARIABLES
C
      TOPE=Ø
      DIR=Ø
C
C     SE ACEPTA EL VALOR DE N
C
5     CONTINUE
      ACCEPT "PARA QUE NUMERO QUIERES EL FACTORIAL? ",N
      IF(N.LE.Ø) GO TO 6Ø
C
C     SALVA PARAMETROS
C
1Ø    CONTINUE
      TOPE=TOPE+1
      PILA(1,TOPE)=N
      PILA(2,TOPE)=DIR
C
C     CRITERIO BASE
C
      IF(N.NE.Ø) GO TO 2Ø
      FACTORIAL=1
      GO TO 4Ø
C
C     SI EL CRITERIO BASE NO SE ALCANZA
C     SE PROCEDE RECURSIVAMENTE
C
2Ø    CONTINUE
      N=N-1
      DIR=1
      GO TO 1Ø
C
C     CALCULO DEL FACTORIAL N*(N-1)
C
3Ø    CONTINUE
      FACTORIAL=FACTORIAL*N
      IF(DIR.EQ.Ø) GO TO 5Ø
C

C     RESTAURA PARAMETROS DE LA LLAMADA ANTERIOR
C
4Ø    CONTINUE
      TOPE=TOPE-1
      N=PILA(1,TOPE)
      DIR=PILA(2,TOPE)
      GO TO 3Ø
C
C     TERMINA EL CALCULO DE N!
C
5Ø    CONTINUE
      TYPE "EL FACTORIAL DE ",N,"ES IGUAL A ",FACTORIAL
      GO TO 5
6Ø    CONTINUE
      STOP
      END
```

Es claro que el programa resulta ser más extenso, ya que requiere de más líneas de código, básicamente por la implementación de las operaciones de PUSH y POP.

b) Manipulación de expresiones aritméticas.- En la notación convencional o notación infija, se define una jerarquía para los operadores aritméticos (+, -, ↑, *, ÷) de tal forma que la operación A + B * C debe efectuarse en el siguiente orden B * C y después más A; si lo que deseamos es primero sumar A con B para multiplicarlo por C será necesario introducir a la notación los paréntesis (A+B)*C.

Evaluación de expresiones

En la notación polaca, todos los operandos poseen la misma jerarquía y el orden de evaluación está dado por la posición de los operandos.

Ejemplo III.3 Representación prefija y sufija de las operaciones.

Notación infija	Not. Pol. Prefija	Not. Pol. Sufija
A↑B	↑AB	AB↑
A*B	*AB	AB*
A/B	/AB	AB/
A+B	+AB	AB+
A-B	-AB	AB-
A∿B	∿AB	AB∿

En la notación polaca prefija, encontramos que el operador antecede a los operandos y, si el operador es binario, es de esperarse dos operandos. En la notación polaca sufija encontramos que los operandos anteceden al operador.

Ejemplo III.4

Representación prefija y sufija de las expresiones.

Infija	Prefija	Sufija
A/C + B/D	+/AC /BD	AC/BD/+
A * (B-C) + D/E	+ * A-BC/DE	ABC - * DE/ +

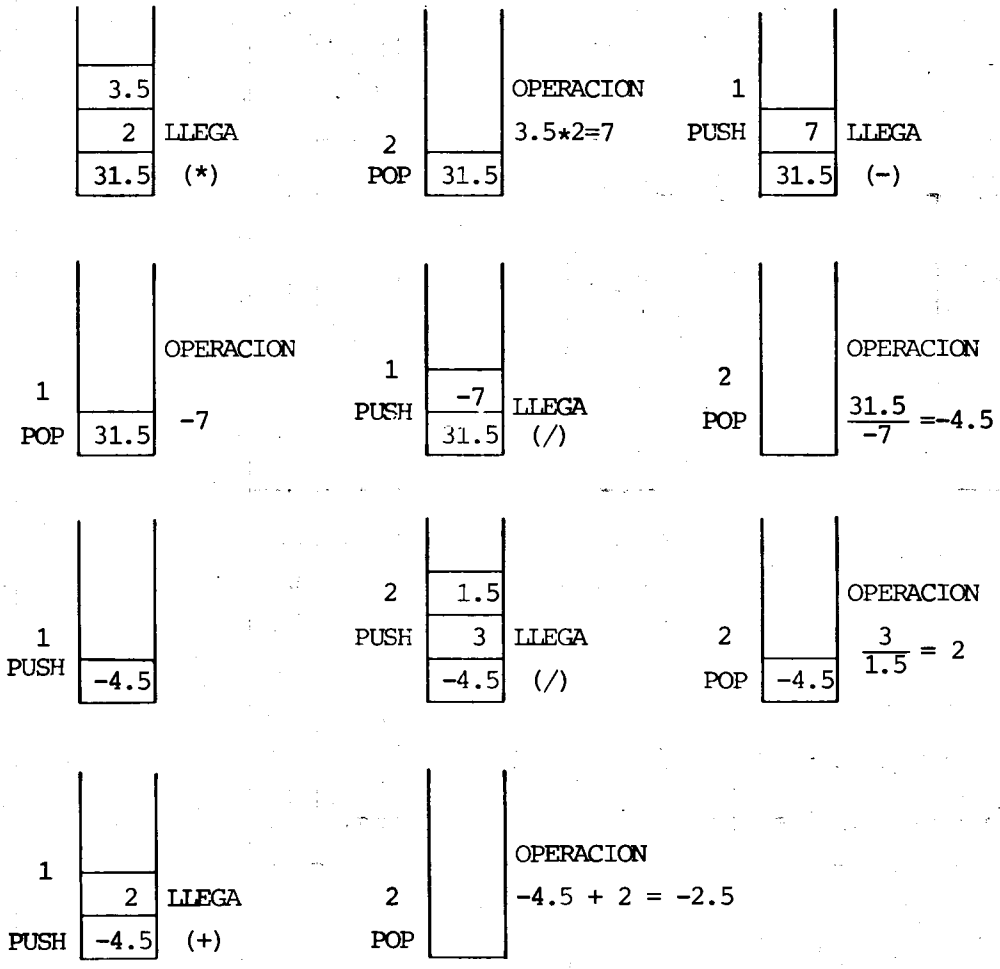
Para evaluar una expresión escrita en notación polaca sufija, el uso de una pila facilita la evaluación. Consideremos la siguiente expresión:

31.5/(- (2*3.5)) + 3/1.5

la cual al pasarse a notación polaca sufija quedaría:

31.5 2 3.5 *-/ 3 1.5 / +

Vamos a leer la expresión de izquierda a derecha e iremos introduciendo (PUSH) a la pila cada uno de los elementos, hasta encontrar un operador, en ese momento se practicará la operación entre los últimos operandos metidos a la pila haciendo POPS, se practicará la operación y se depositará el resultado nuevamente en la pila haciendo un PUSH.



III.3 COLA

III.3.1 DEFINICION Y OPERACIONES

Una cola es una estructura de datos lineal en la cual las operaciones se realizan por ambos extremos. Se agrega por uno de los extremos y se retira por el otro. Son ejemplos de cola, una línea de espera para comprar boletos, para pagar o para cobrar en un banco, para comprar gasolina, para pasar un semáforo, etc. Un modelo para una cola es el de la figura III.7.

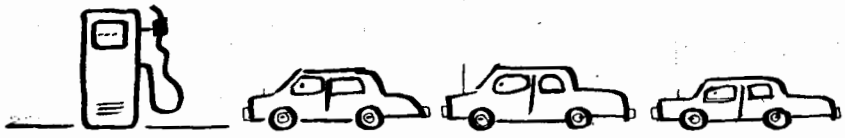


Figura III.7 Formación en una bomba de gasolina como modelo de una cola.

Ejemplo III.5

Para ilustrar la forma de operación de una cola consideramos que en la figura III.7 un nuevo elemento desea agregarse a la cola, éste solamente podría hacerlo después del último.

Nuevos elementos que se agregan a la cola

primero ..... último

y el elemento que puede retirarse de la cola es aquel que se encuentra al frente de la taquilla o sea el primero de la cola.

Salen de la cola

primero ..... último

Por la forma en que se agregan los elementos a una cola el método ha sido llamado PEPS (primeras entradas, primeras salidas) en inglés FIFO (first input, first output).

Primeras entradas. Primeras salidas

En algunas aplicaciones también el método es conocido como PAPS (primeros en arribar, primeros en ser servidos) o en inglés FCFS (first come, firts served).

Con esta política se entiende que solamente puede ser retirado de la cola el primer elemento agregado, pero sabemos que en algunas aplicaciones algunos elementos abandonan la cola sin ser servidos, y que otros, que no están al frente de la cola pueden ser servidos, creándose en este caso una cola con prioridades. La prioridad se logra ya sea por la finalidad, por algún parentesco, por una propina, etc.

III.3.2 REPRESENTACION Y ALGORITMOS DE LAS OPERACIONES

Una cola puede ser representada en la computadora eligiendo localidades contiguas de la memoria de acuerdo a como se muestra en la figura III.8.

Almacenamiento contiguo de una cola

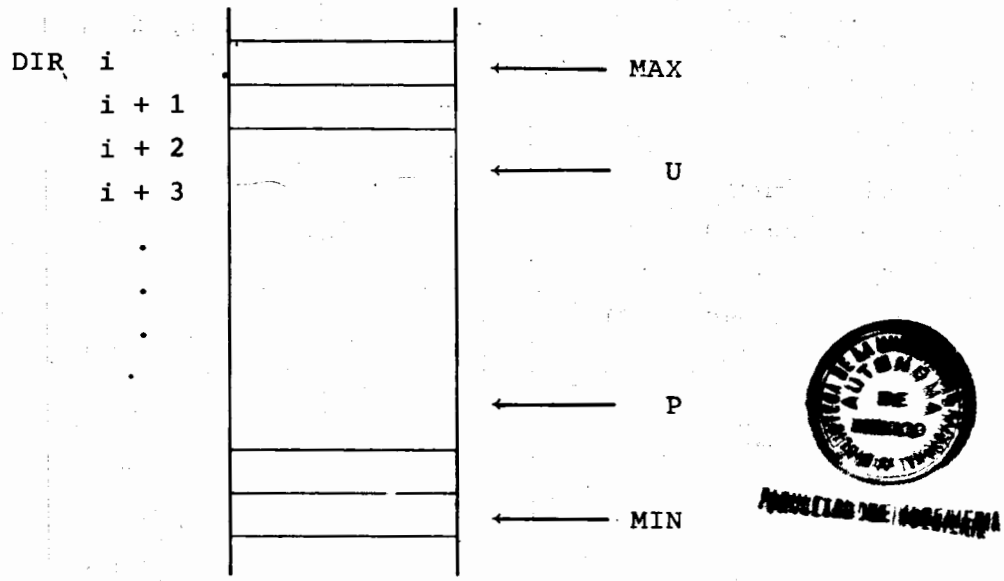


Figura III.8 Cola almacenada en localidades contiguas.

Para indicar la localidad donde se agregará un nuevo elemento se tiene un apuntador U (último de la cola) y para indicar la localidad de donde se retirará un elemento se tiene al apuntador P (primero de la cola).

Cola llena y Cola vacía

De forma similar a la pila, la cola requiere que se delimite el número de localidades sobre las que se implementará para considerar el caso de cola llena (overflow). La cola estará vacía cuando U = P = vacío.

ALGORITMOS DE LAS OPERACIONES DE AGREGAR Y RETIRAR

Para agregar un nuevo elemento a una cola de acuerdo a la figura III.8 se requiere de un algoritmo que realice lo siguiente:

```
SI    U=MAX
      ESCRIBE ('COLA LLENA (OVERFLOW)')
OSI   U=Δ
      U=P=MIN
      COLA(U)=DATO
OBIEN
      U=U-1
      COLA(U)=DATO
FIN
```

Para retirar un elemento de la cola se requiere de un algoritmo que realice lo siguiente:

```
SI    U=P=Δ
      ESCRIBE ('COLA VACIA (UNDERFLOW)')
OSI   U=P
      DATO=COLA(P)
      P=U=Δ
OBIEN
      DATO=COLA(P)
      P=P-1
FIN
```

Almacenamiento ligado de una cola

Una cola también puede ser representada usando localidades ligadas de memoria. (Véase figura III.9).

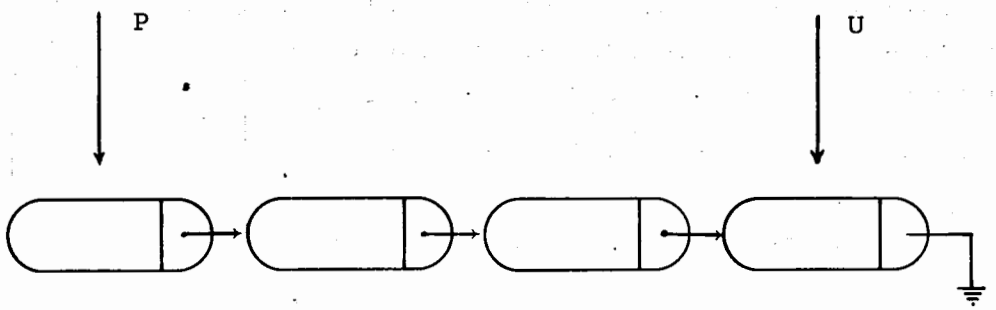


Figura III.9 Cola almacenada en localidades ligadas.

Algoritmo de las operaciones de agregar y retirar.

Para agregar un elemento a una cola almacenada en localidades ligadas se requiere:

1.- Obtener un nuevo elemento (N).

2.- Agregarlo a la cola.

En forma gráfica el proceso puede ser ilustrado por la figura III.10.

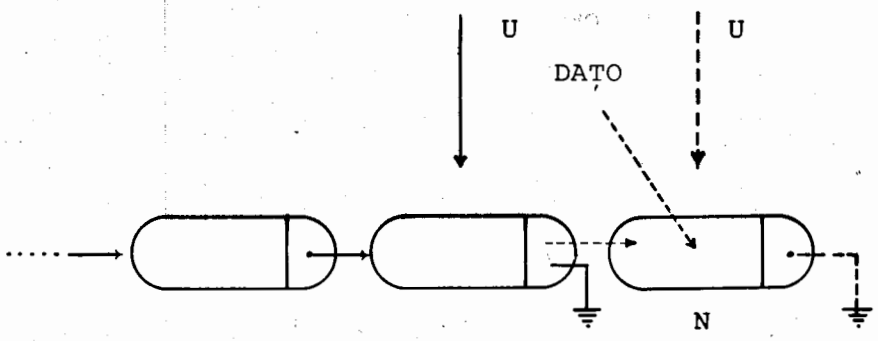


Figura III.10 Operación de agregar en una cola almacenada en localidades ligadas.

Un algoritmo que realiza la operación de agregar es el siguiente:

```
SI N SE OBTIENE
        LIGA(N)=Δ
        DATOC(N)=DATO
        SI U=P=Δ
           U=P=N
        OBIEN
           LIGA(U)=N
           U=N
        FIN
OBIEN
        ESCRIBE ('COLA LLENA (OVERFLOW)')
FIN
```

Para retirar un elemento de la cola se requiere de un algoritmo que realice lo siguiente:

```
SI  U=P=Δ
    ESCRIBE ('COLA VACIA (UNDERFLOW)')
OBIEN
    DATO=DATOC(P)
    N=P
    P=LIGA(P)
    RETORNAR N
    SI P=Δ
       U=Δ
    FIN
FIN
```

III.4 COLA DOBLE

III.4.1 DEFINICION Y OPERACIONES

Una cola doble es una estructura de datos en la cual las operaciones de agregar y retirar se practican por ambos extremos.

Una cola doble es una estructura en la que, por la forma en que se realizan las operaciones, puede comportarse como una pila o como una cola.

Un modelo que ha sido sugerido para la cola doble es la implementación ferroviaria de la figura III.11.

Modelo para una cola doble

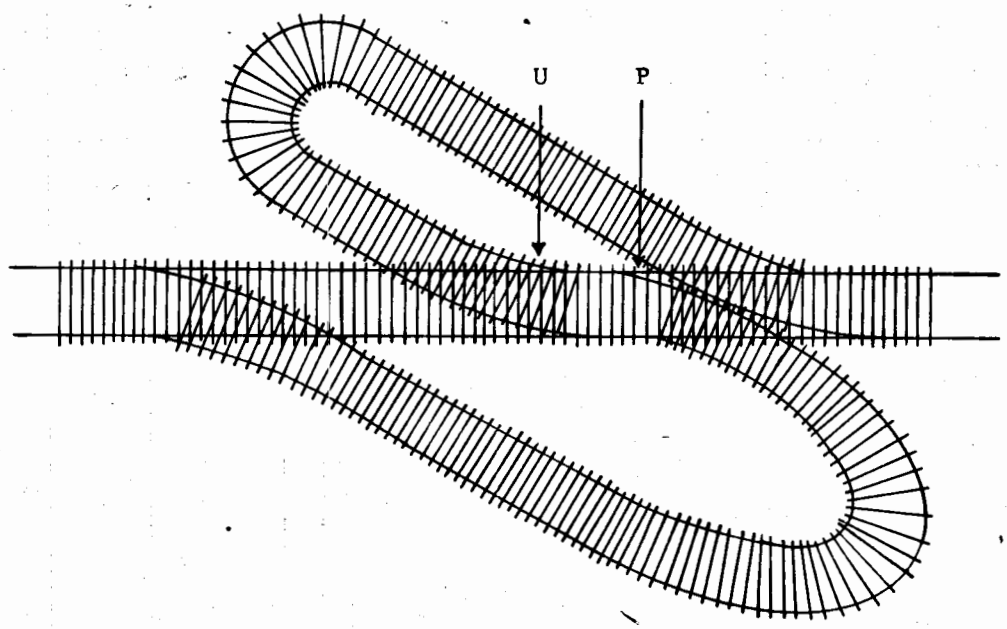


Figura III.11 Implementación ferroviaria como un modelo para una cola doble.

Ejemplo III.6

Para ilustrar la forma de operar de una cola doble, consideremos que en la figura III.11 un nuevo elemento desea agregarse a la cola, éste podría hacerlo de tal forma que ocupe la primera posición o la última, los elementos que se encuentran al principio y al final de la cola pueden retirarse.

Por la forma como se agregan y retiran elementos, no existe un método para la cola doble, aunque es posible practicar los métodos PEPS y UEPS ó una combinación de ellos.

III.4.2 REPRESENTACION Y ALGORITMOS DE LAS OPERACIONES

Almacenamiento contiguo para una cola doble

Una cola doble puede ser representada eligiendo localidades contiguas de memoria de acuerdo a la figura III.12.

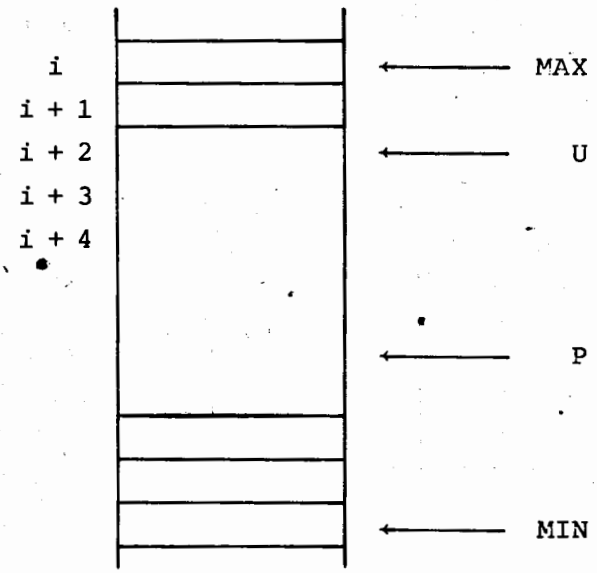


Figura III.12 Cola doble almacenada en localidades contiguas.

Cola doble llena y Cola doble vacía

Para indicar uno de los extremos de la cola doble por donde se agregarán o retirarán elementos, usaremos un apuntador (U) y para el otro extremo usaremos un apuntador (P). Para delimitar el área de implementación usaremos dos apuntadores MAX y MIN, de idéntica forma como en la pila y la cola para detectar la cola doble llena. La cola doble estará vacía cuando P = Vacío.

ALGORITMOS DE LAS OPERACIONES DE AGREGAR Y RETIRAR

En una cola doble la operación de agregar se puede realizar por U o por P al igual que la operación de retirar. De esta forma, son cuatro los algoritmos que se requieren para el manejo de una cola doble:

- agregar por U
- retirar por U
- agregar por P
- retirar por P

Para el almacenamiento secuencial sólo se presentarán los algoritmos para agregar por U y para agregar por P.

Un algoritmo que permite agregar un elemento por U es el siguiente:

```
SI  U=MAX
     ESCRIBE ('COLA DOBLE LLENA (OVERFLOW)')
OSI P=U=Δ
    U=P=(ALGUN VALOR ENTRE MAX Y MIN)
    COLA DOBLE (U)=DATO
OBIEN
    U=U-1
    COLA DOBLE (U)=DATO
FIN
```

Para agregar por P el algoritmo debe realizar lo siguiente:

```
SI  P=MIN
    ESCRIBE ('COLA DOBLE LLENA (OVERFLOW)')
OSI P=U=Δ
    U=P=(ALGUN VALOR ENTRE MAX Y MIN)
    COLA DOBLE (P)=DATO
OBIEN
    P=P+1
    COLA DOBLE (P)=DATO
FIN
```

Almacenamiento ligado para una cola doble

Es posible almacenar una cola doble en localidades ligadas de memoria de acuerdo a la figura III.13.

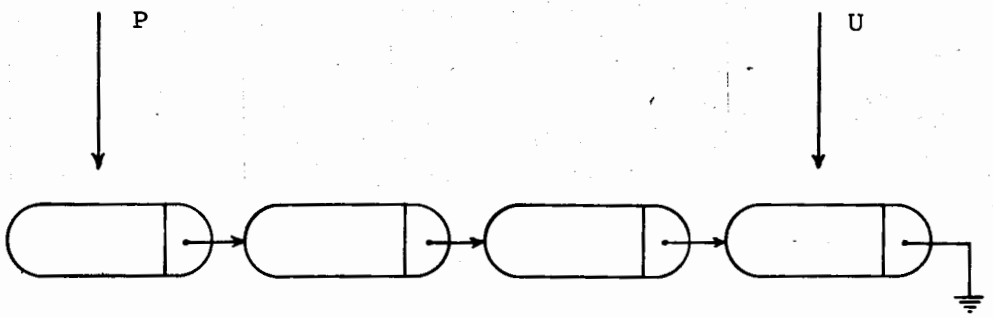


Figura III.13 Cola doble almacenada en localidades ligadas.

ALGORITMO DE LAS OPERACIONES DE AGREGAR Y RETIRAR

Consideremos que la cola doble está vacía cuando P=U=Δ y que estará llena cuando no sea posible obtener N.

Un algoritmo para agregar por U es el siguiente:

```
SI N SE OBTIENE
     DATOC(N)=DATO
     LIGA (N)=Δ
     SI U=Δ
        U=P=N
     OBIEN
        LIGA(U)=N
        U=N
     FIN
OBIEN
     ESCRIBE ('COLA DOBLE LLENA (OVERFLOW)')
FIN
```

Un algoritmo para retirar por P es el siguiente:

```
        SI    P=U=Δ
              ESCRIBE ('COLA DOBLE VACIA (UNDERFLOW)')
        OSI   P=U
        DATO=DATOC(P)
        N=P
        SE RETORNA N
        P=U=Δ
    OBIEN
        DATO=DATOC(P)
        N=P
        P=LIGA(P)
        SE RETORNA N
    FIN
```

La operación de retirar por U no es directa, ya que no es posible conocer desde U cuál es el elemento que se encuentra antes de U, el que vendría a ser el nuevo elemento a la cabeza por U.

III.5 LISTA CIRCULAR

III.5.1 DEFINICION Y OPERACIONES

Una lista circular es una estructura de datos que tiene como característica fundamental un orden en el que, a la última localidad de almacenamiento, le sigue la primera, o que al último nodo le sigue el primero.

Este tipo de lista viene a solucionar el problema de desplazamiento de los nodos sobre la memoria como ocurre en el caso de la cola, o el problema de mover los nodos a una posición después de un borrado.

Los algoritmos para una cola dados anteriormente, ocasionan que la cola se mueva entre los límites de implementación. (Véase figura III.14).

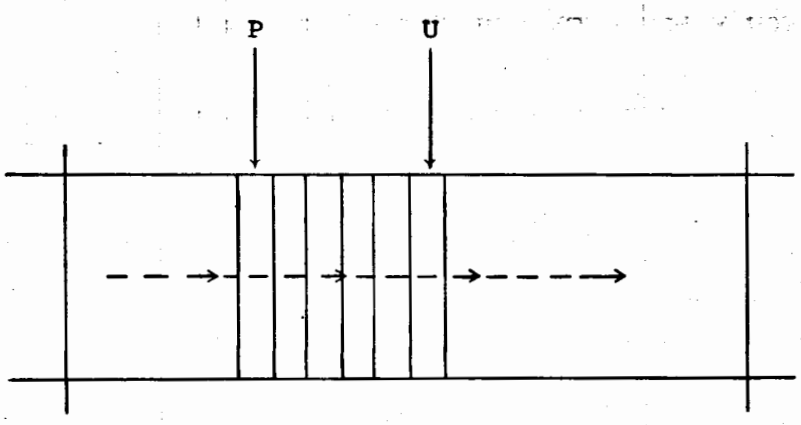


Figura III.14 Desplazamiento de una cola en la memoria.

Una alternativa para evitar que la cola se desplace es mantener fija su cabeza, esto es que P no se mueva y que, cuando se retire un elemento de la cola, se muevan todos los elementos una localidad hacia la izquierda. (Véase figura III.15).

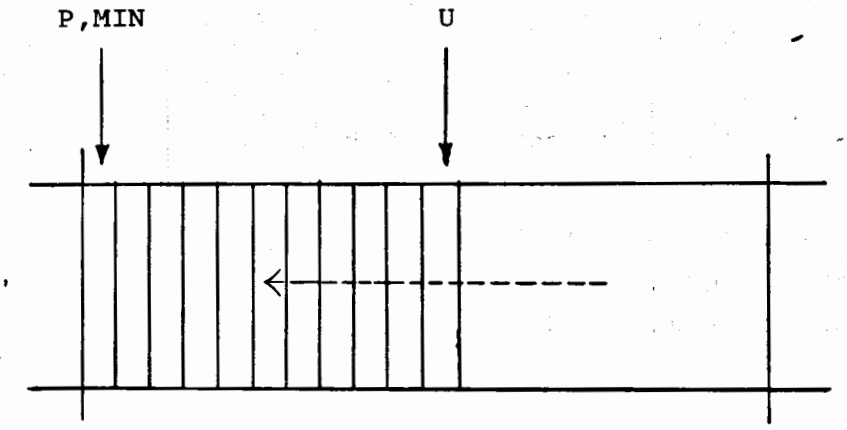


Figura III.15 Cola en la que el primer elemento por retirar siempre está en MIN.

Mover los elementos una localidad puede ser muy costoso si la cola es muy larga, pero solucionaría el problema del desplazamiento. En estos casos, la mejor alternativa es el uso de una lista circular.

III.5.2 REPRESENTACION Y ALGORITMOS DE LAS OPERACIONES

Almacenamiento contiguo de una lista circular

Una lista circular puede ser representada en la computadora usando localidades contiguas de memoria de acuerdo a la figura III.16

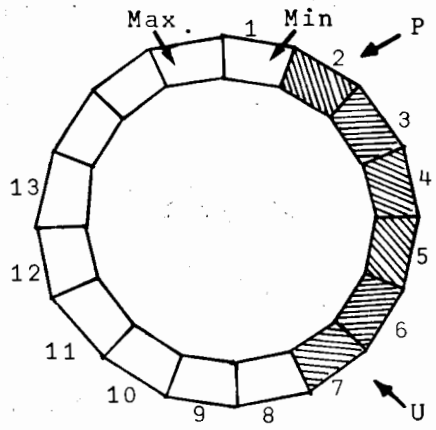


Figura III.16 Lista circular para el manejo de una cola.

ALGORITMOS DE LAS OPERACIONES DE AGREGAR Y RETIRAR

La lista circular de la figura III.16 es usada para el manejo de una cola con el auxilio de los apuntadores P y U que indican el principio y el fin de la cola.

La idea de lista circular permite que la cola se desplace sobre la memoria, pero al llegar a una cierta localidad límite por la cual normalmente diríamos que la cola no puede seguir creciendo, tomaremos las localidades disponibles por donde se inició la cola. (Véase figura III.16).

De acuerdo a lo anterior un algoritmo que realiza la operación de agregar es el siguiente:

```
SI      (P=U+1) o (U=MAX Y P=MIN)
        ESCRIBE ('COLA LLENA (OVERFLOW)')
OSI     P=U=Δ
        U=P=MIN
        COLA CIRCULAR(U)=DATO
OSI     U=MAX
        U=MIN
        COLA CIRCULAR(U)=DATO
OBIEN
        U=U+1
        COLA CIRCULAR(U)=DATO
FIN
```

Para retirar de la cola un elemento el algoritmo debe realizar las siguiente operaciones:

```
SI      P=U=Δ
        ESCRIBE ('COLA VACIA (UNDERFLOW)')
OSI     P=U
        DATO=COLA CIRCULAR(P)
        P=U=Δ
OSI     P=MAX
        DATO=COLA CIRCULAR(P)
        P=MIN
OBIEN
        DATO=COLA CIRCULAR(P)
        P=P+1
FIN
```

Una lista circular también puede almacenarse en localidades de memoria ligadas. (Véase figura III.17).

Almacenamiento de una cola circular ligada

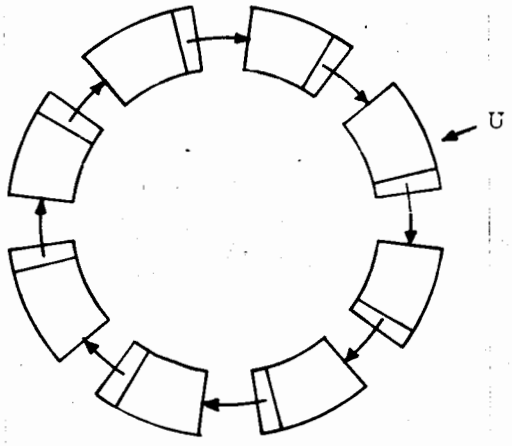


Figura III.17 Lista circular almacenada en localidades de memoria ligadas.

En el manejo de una cola almacenada en forma circular los apuntadores P y U pueden reducirse a un solo apuntador U ya que P es la liga de U. De esta forma, el apuntador U

será el único elemento que participe en las operaciones de agregar y de retirar.

ALGORITMO DE LAS OPERACIONES DE AGREGAR Y RETIRAR

En la figura III.17 el algoritmo para agregar un elemento, si consideramos la representación como la de una cola es:

```
SI N SE OBTIENE
     SI U=Δ
          DATOC(N)=DATO
          LIGA (N)=N
          U=N
     OBIEN
          DATOC(N)=DATO
          LIGA (N)=LIGA(U)
          LIGA (U)=N
          U=N
     FIN
OBIEN
     ESCRIBE ('COLA LLENA (OVERFLOW)')
FIN
```

Para retirar un elemento de la cola, se requiere de un algoritmo que realice las operaciones siguientes:

```
SI   U=Δ
     ESCRIBE ('COLA VACIA (UNDERFLOW)')
OSI   U=LIGA(U)
      DATO=DATOC(U)
      N=LIGA(U)
      U=Δ
      SE RETORNA N
OBIEN
      DATO=DATOC(LIGA(U))
      N=LIGA(U)
      LIGA(U)=LIGA(LIGA(U))
      SE RETORNA N
FIN
```

III.6 LISTAS DOBLEMENTE LIGADAS

III.6.1 DEFINICION Y OPERACIONES

En una lista doblemente ligada se han incluido dos campos liga; uno que señala al nodo sucesor, llamado liga derecha (LD), y el otro que señala al nodo antecesor, llamado liga izquierda (LI). Al incluir estos dos campos, se logra un manejo más eficiente de las listas, como podría ser, conocer desde cualquier nodo cuál es el nodo sucesor y cuál es el antecesor, cosa que nos es imposible lograr en forma sencilla, en una lista con una sola liga. La figura III.18 representa una lista doblemente ligada con dos apuntadores I y D que denotan al nodo más hacia la izquierda y al nodo más hacia la derecha en la lista.

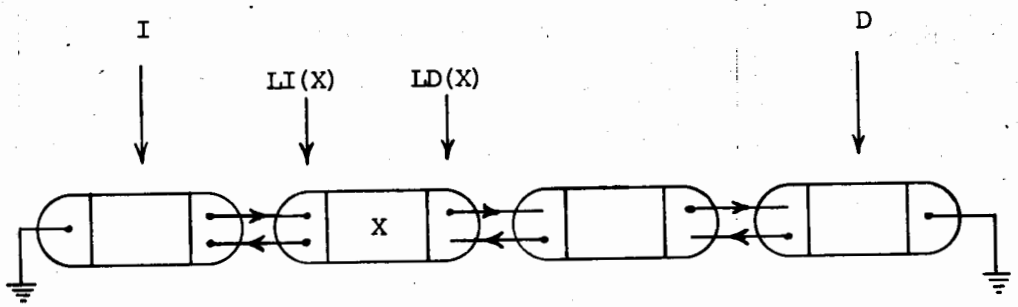


Figura III.18 Lista doblemente ligada.

III.6.2 REPRESENTACION Y ALGORITMOS DE LAS OPERACIONES

Por la estructura de la lista, es posible agregar o retirar un nodo conociendo cualquier nodo de la lista. Consideremos el caso de agregar un nodo (N) en una lista no vacía del lado izquierdo del nodo (M). Gráficamente la operación estaría representada por la figura III.19.

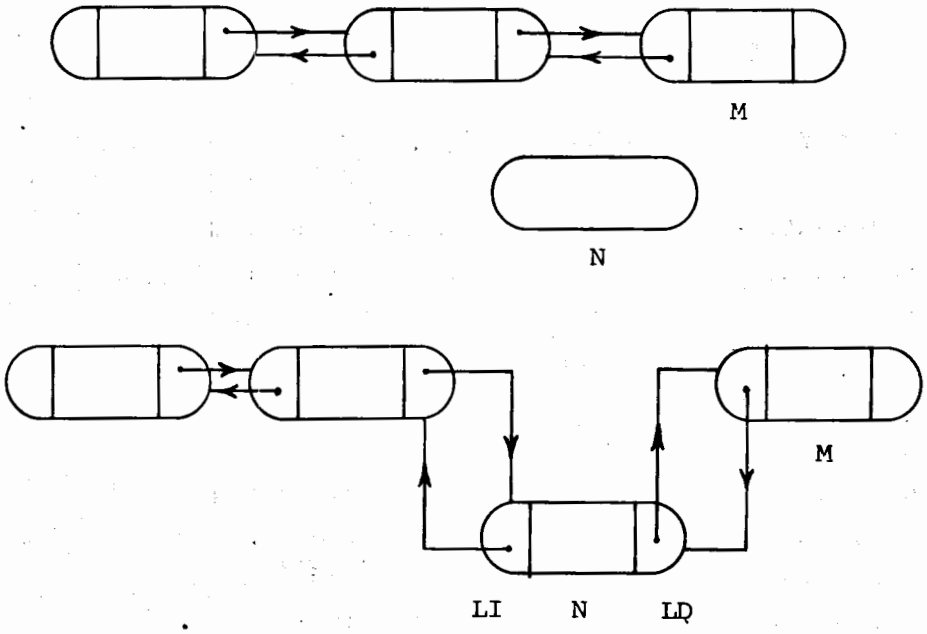


Figura III.19 Representación de la operación de agregar.

Un algoritmo que realiza las operaciones para agregar el nodo (N) a la izquierda del nodo (M) es:

```
DATOD(N) = DATO
LI (N) = LI (M)
LD (N) = M
LD(LI(M)) = N
LI (M) = N
```

Otras operaciones serían las de agregar un nodo a la derecha del nodo (M), agregar un nodo cuando la lista está vacía, etc. En los algoritmos que se han presentado muchos contemplan el caso de lista vacía lo que trae como consecuencia la aplicación de una parte especial del algoritmo. Una forma de simplificar los algoritmos evitando el manejo de lista vacía es no permitir que la lista quede vacía, esto puede lograrse incorporando a la lista un nodo especial, llamado cabeza de lista, el que siempre permanece en ella. Para evitar los casos particulares en los extremos de una lista doblemente ligada, como serían agregar un elemento a la izquierda del nodo más a la izquierda y el de agregar a la derecha del nodo más a la derecha, se sugiere hacer la lista circular de acuerdo a como se muestra en la figura III.20.

Lista circular doblemente ligada

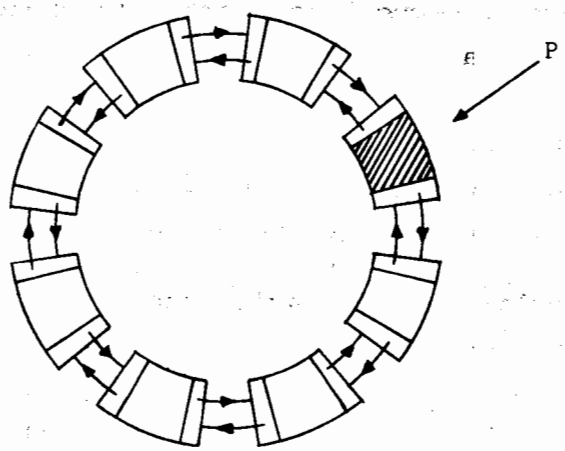


Figura III.20 Lista circular doblemente ligada con cabeza de lista.

De acuerdo con la figura III.20 podemos decir que para cualquier nodo (M) se cumple que:

$$LD\ (LI\ (M)) = LI\ (LD\ (M)) = M$$

De esta forma en los algoritmos de agregar y retirar se han simplificado de tal manera de que, para agregar un elemento a la derecha de cualquier nodo, se realizarían las operaciones siguientes:

```
DATOD(N)=DATO
LD(N)=LD(M)
LI(N)=M
LI(LD(M))=N
LD(M)=N
```

III.7 CONSIDERACIONES SOBRE EL ALMACENAMIENTO CONTIGUO Y LIGADO

El almacenamiento contiguo aparenta hacer un uso eficiente de memoria por la cantidad de localidades que utiliza, pero hay que tener en cuenta, que en muchos casos, siempre se delimita el área de implementación y, aunque la estructura requiera de pocas localidades, las ya reservadas no pueden ser utilizadas. El almacenamiento ligado requiere

un poco más de memoria por el campo liga, pero esto puede ser aparente, ya que el nodo a veces no puede ser de la longitud exacta de la información y en este caso el espacio restante se puede aprovechar para la liga. Por otro lado, la información en un nodo puede ser común a varias listas evitando que ésta se multiplique en ellas.

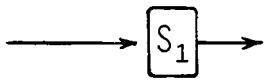
Los algoritmos para manipular la información que está almacenada en forma contigua resultan aparentemente más fáciles de realizar, aunque para retirar un elemento casi siempre hay que recorrer una gran cantidad de elementos, mientras que, en un almacenamiento ligado, bastará con alterar las ligas.

Para la recuperación resulta ser más eficiente, desde el punto de vista de comparaciones, el almacenamiento contiguo, ya que puede practicarse en forma directa; mientras que en el almacenamiento ligado, se requiere de un mayor número de comparaciones por el recorrido que hay que hacer; indicado éste, por los campos liga.

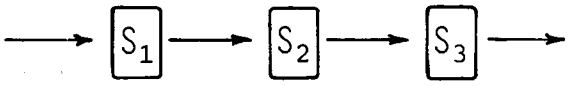
Ejemplos:

La formación de una línea de espera es un mecanismo para regular el orden en que cada elemento de la cola será servido. Existen diferentes tipos de colas de acuerdo a la forma como los servicios se asignan, por ejemplo:

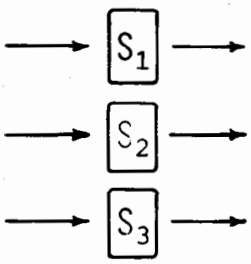
a) un solo servidor y una sola cola



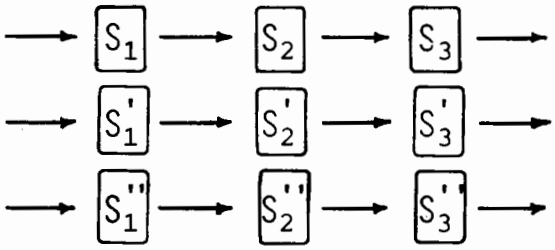
b) varios servidores diferentes y una sola cola



c) varios servidores iguales y varias colas



d) varios servidores diferentes y varias colas.



La formación de una cola con el objeto de esperar un servicio, es un mecanismo que encontramos en diversidad de situaciones, por ejemplo: una cola para comprar boletos en un cine, una cola para pagar la luz en un banco, una cola en un restaurante de autoservicio para tomar los alimentos, una cola de programas para ser procesados por la computadora, etc.

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

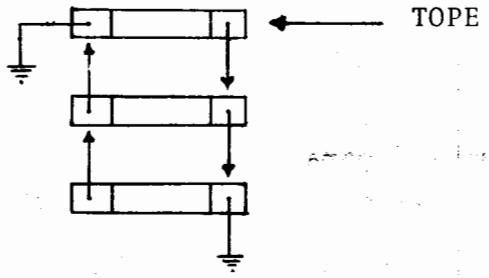
1.	Lista	( ) En esta lista, con el campo de liga sólo es posible conocer a su sucesor.
2.	Lista lineal	( ) Método que se emplea en una cola para agregar y retirar elementos.
3.	Pila	( ) En esta lista, el último elemento agregado es el primero en ser retirado.
4.	Cola	( ) Conjunto de nodos que puede ser aumentado o disminuido.
5.	Cola doble	( ) En esta lista, el primer elemento agregado es el primero en ser retirado.
6.	Peps	( ) En esta lista, el último nodo tiene un apuntador al primero.
7.	Ueps	( ) Método que se emplea en una pila para agregar y retirar elementos.
8.	Lista circular	( ) En esta lista, sólo el último y el primer nodo no tienen sucesor y antecesor respectivamente.
9.	Lista doblemente ligada	( ) Es una lista donde las operaciones de agregar y retirar se practica en ambos extremos.
10.	Lista simple ligada	( ) En este tipo de lista, desde cualquier nodo es posible conocer a su antecesor y a su sucesor.
		( ) Es un ejemplo la línea de espera en la parada de camiones.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

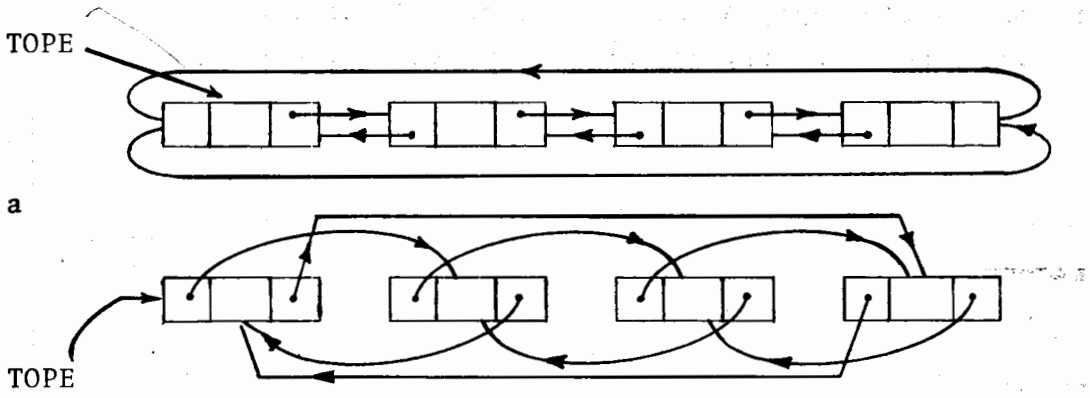
1. Considere los elementos ABCDEF. Diga con qué listas (pila, cola, cola doble), se obtuvieron las siguientes secuencias. Los elementos fueron tomados de izquierda a derecha.

		pila	cola	cola doble
a.	ABCDEF	(X)	( )	( )
b.	ABDCEF	( )	( )	( )
c.	FEABCD	( )	( )	( )
d.	DBECFA	( )	( )	( )
e.	DEFABC	( )	( )	( )
f.	AFDECB	( )	( )	( )
g.	FEDCBA	( )	( )	( )

2. Escriba los algoritmos de agregar y retirar para una PILA almacenada de acuerdo a la figura siguiente:



3. Escriba un algoritmo para transformar la lista siguiente:



4. Transforme la siguiente expresión a su equivalente en notación POLACA SUFIJA y calcule la expresión, operación por operación, con el uso de una pila.

$$|A > F(B * .5) \quad o \quad (B * C) <> (D - 12.3)|$$

y

$$|(D+C) <= (A-B)|$$

donde A = 1, B = 2, C = 3, D = 4

5. Escriba una función recursiva para calcular la raíz cuadrada de un número utilizando la definición siguiente:

$$RAIZ\ (n, a, e) = \begin{cases} a; & \text{si } |a\uparrow 2 - n| < e \\ RAIZ\ (n, (a\uparrow 2+n)/2a, e); & \text{en cualquier otro caso} \end{cases}$$

donde n = número del que se desea la raíz
a = raíz del número
e = error permitido

6. Se desea instalar una gasolinera con una sola bomba en la carretera que va a la ciudad X. Después de estudiar los parámetros para la demanda de servicio se vio que:

   los automóviles arribarán en el intervalo $30 \leq T \leq 10$ segundos y pueden demandar los servicios siguientes:

   - gasolina a razón de 1 litro por 3 segundos (30 lts $\leq$ tanque $\leq$ 100)
   - agua al radiador 300 segundos
   - aire 30 segundos por llanta
   - limpiar el vidrio delantero 60 segundos
   - verificar aceite y completar 240 segundos
   - pagar exacto 10 segundos
   - pagar y hay que dar cambio 60 segundos

   Simule el proceso y diga si será o no eficiente la gasolinera con una sola bomba.

UNIDAD IV ESTRUCTURAS DE DATOS COMPUESTAS: LISTAS NO LINEALES

OBJETIVO GENERAL

El alumno aplicará las formas de representar y operar en la computadora las principales listas no lineales.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Reconocerá los conceptos de lista no lineal, gráfica y árbol.

2. Identificará los elementos de una gráfica y de un árbol.

3. Representará en la computadora gráficas, usando arreglos y listas ligadas.

4. Realizará operaciones de recorrido sobre los árboles binarios.

5. Escribirá algoritmos para las operaciones sobre los árboles binarios.

6. Resolverá problemas en los que se utilicen listas no lineales.

INTRODUCCION

En la Unidad III se precisó que una lista lineal es una estructura de datos que expresa las relaciones entre los nodos por un solo criterio o en una sola dimensión; el propósito de esta unidad, es el estudio de estructuras más complejas, aquellas cuyas relaciones entre sus nodos son en más de una dimensión. Iniciaremos la unidad, estudiando las gráficas y los árboles como estructuras de datos no lineales y los conceptos asociados a ellas, posteriormente se verá su representación en la computadora y algunas de las operaciones más importantes que se practican sobre ellas.

IV.1 GENERALIDADES

IV.1.1 CONCEPTOS Y DEFINICIONES DE GRAFICAS

Nodos y Arcos

Una gráfica G denotada como G=(A,R), es una relación de R sobre un conjunto A. Los elementos de A, son llamados nodos, puntos o vértices y los elementos de R son llamados arcos o líneas.

Una relación R de un conjunto S a T es cualquier subconjunto del producto cartesiano SxT, esto es:

$$R \subseteq \{<s, t> \mid s \in S \quad y \quad t \in T\}$$

Producto cartesiano

El producto cartesiano SxT de dos conjuntos, es el conjunto de pares ordenados, de tal forma que la primera coordenada de cada par es un miembro de S y la segunda es un miembro de T, esto es:

$$SxT = \{<s, t> \mid s \in S \quad y \quad t \in T\}$$

Ejemplo IV.1

Sea A=(1, 2) y B=(a, b, c) el producto cartesiano AxB es:

$AxB = \{<1, a><1, b><1, c><2, a><2, b><2, c>\}$

Una relación R de A a B, es cualquier subconjunto AxB, esto es:

$R1 = \{<1, a><1, b>\}$

$R2 = \{<1, a><2, b><2, c>\}$

De acuerdo a la definición de gráfica dada, consideremos que A es el conjunto de los elementos $(n_1, n_2, n_3, n_4, \ldots,)$, el producto cartesiano AxA es $\{<n_1, n_1><n_1, n_2><n_1, n_3> \ldots$ $\ldots\ldots<n_2, n_1><n_2, n_2><n_2, n_3> \ldots < n_3, n_1><n_3, n_2><n_3, n_3>$. $\ldots\ldots\ldots\}$. Una gráfica G es cualquier subconjunto AxA.

Ejemplo IV.2

Sea $A=\{a, b, c\}$ y $R=\{<a, a><a, b><b, c><a, c><c, b><b, a><b, b>\}$.

La figura IV.1 representa a la gráfica $G=(A,R)$.

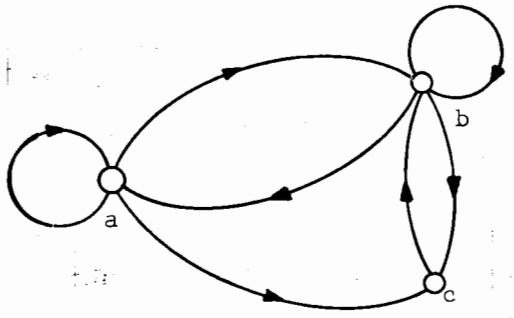


Figura IV.1 Representación para la gráfica del ejemplo IV.2.

ARCO DIRIGIDO

Si en un arco $e = <u, v>$, es importante considerar que u es el nodo inicial, u=inic(e) o sea, el nodo de donde parte el arco, y v el nodo final, v=fin(e), el nodo a donde llega el arco, estaremos hablando de un arco dirigido. (Véase la figura IV.2).

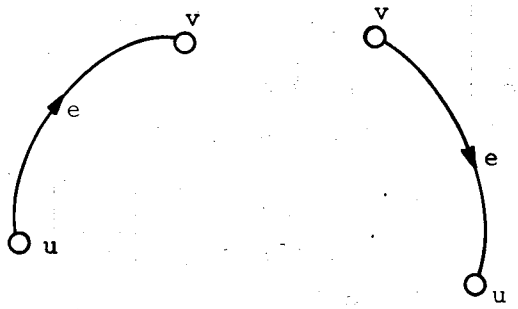


Figura IV.2 Arcos dirigidos.

GRAFICA DIRIGIDA

Una gráfica cuyos arcos son todos dirigidos es llamada gráfica dirigida. (Véase figura IV.3).

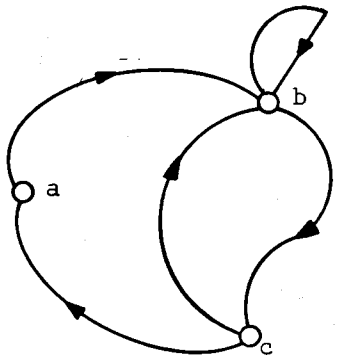


Figura IV.3 Gráfica dirigida.

GRADO EXTERNO DE UN NODO

El grado externo de un nodo u es el número de arcos que salen de él. Esto es, el número de arcos e, tales que inic(e) = u.

GRADO INTERNO DE UN NODO

El grado interno de un nodo u, es el número de arcos que llegan a él. Esto es, el número de arcos e, tales que fin(e) =u.

Si en una gráfica dirigida ciertos miembros de R pueden ser colocados en una secuencia de la forma

$$\langle a_1, a_2\rangle\langle a_2, a_3\rangle\langle a_3, a_4\rangle\ldots\ldots\langle a_{n-1}, a_n\rangle,$$

Trayectoria

el conjunto de arcos es llamado una trayectoria desde a_1 hasta a_n. Si $a_n = a_1$, la trayectoria es un ciclo. Una gráfica que contiene al menos un ciclo es llamada gráfica cíclica de otra forma es llamada acíclica. (Véase figura IV.4).

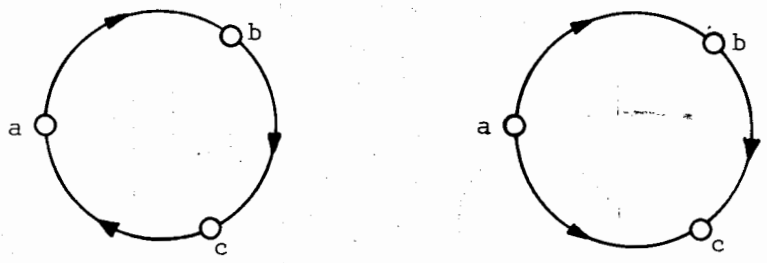


Figura IV.4 Gráficas cíclicas y acíclicas.

Cuando los arcos de la secuencia son distintos, la trayectoria es simple. Si los arcos son distintos y contienen a todos los nodos de A, la trayectoria es hamiltoniana. La longitud de una trayectoria es el número de arcos que la componen.

Trayectoria simple y Hamiltoniana

ARCO NO DIRIGIDO

Si en un arco $e = \langle u, v\rangle$ no es importante considerar cuál es el nodo inicial ni cuál es el nodo final, estaremos hablando de un arco no dirigido $\langle u, v\rangle = \langle v, u\rangle$. (Véase figura IV.5).

Longitud de una trayectoria

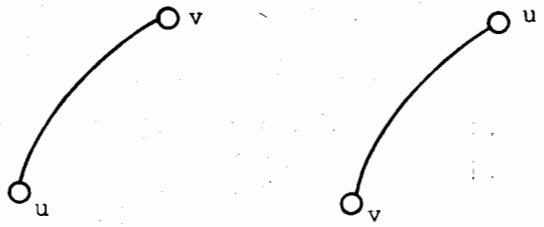


Figura IV.5 Arcos no dirigidos.

LAZO O LOOP

Un arco que une un vértice consigo mismo se llama lazo, (véase figura IV.6). La dirección de un lazo no tiene ningún significado y puede ser considerada como arco dirigido o no dirigido.

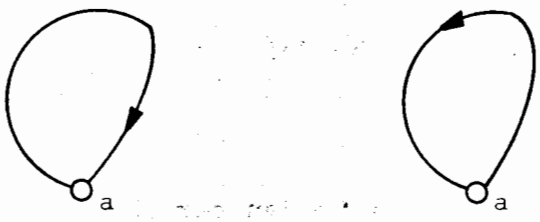


Figura IV.6 Arcos llamados lazo.

GRAFICA NO DIRIGIDA

Una gráfica es no dirigida cuando todos los arcos son no dirigidos. (Véase figura IV.7).

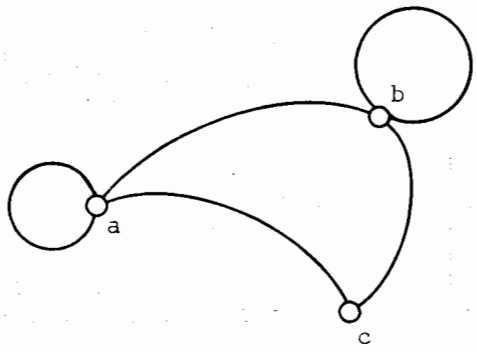


Figura IV.7 Gráfica no dirigida.

GRAFICA MIXTA

Una gráfica es mixta cuando contiene arcos dirigidos y no dirigidos. (Véase figura IV.8).

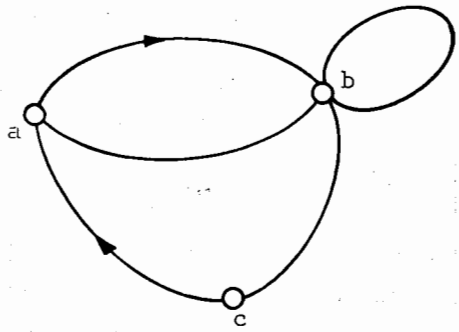


Figura IV.8 Gráfica mixta.

Con la definición de gráfica que se ha dado, no es posible conectar entre dos nodos dos arcos con el mismo sentido, ni conectar más de dos arcos, entre dos nodos.

Estas restricciones en algunos casos pueden ser eliminadas, ya que es posible encontrar que ciertos pares de nodos en algunas aplicaciones están unidos por más de dos arcos, inclusive, con el mismo sentido; tales arcos, son llamados arcos paralelos. (Véase figura IV.9).

Arcos paralelos

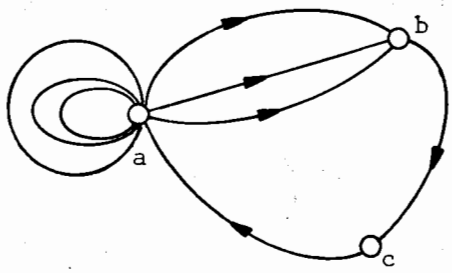


Figura IV.9 Gráfica con arcos paralelos (multigráfica).

Una gráfica que contiene arcos paralelos se llama multigráfica. En caso de contener solamente un arco entre cualquier par de nodos se llama gráfica simple.

Multigráfica

Gráfica simple

En muchas aplicaciones también encontramos que a los arcos de una gráfica se les asignan valores, sea ésta una gráfica dirigida o no. Estos valores, llamados peso del arco, dan origen a una gráfica llamada pesada. (Véase figura IV.10).

Gráfica pesada

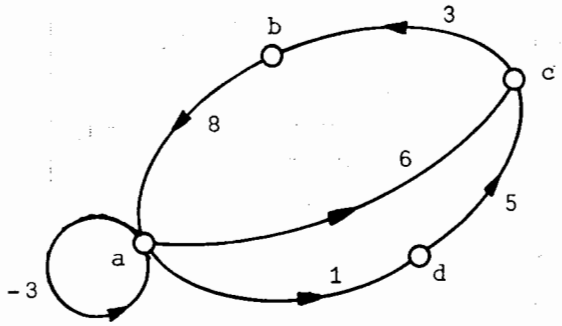


Figura IV.10 Gráficas pesadas.

El peso en muchas situaciones corresponde al costo de algún aspecto del arco, en otras a la capacidad del arco o a algunas características propiedad del arco, etc.

Ejemplo IV.3

Consideremos la gráfica pesada de la figura IV.10.

1. Una trayectoria del nodo a al b es: <a, c><c, b>.
2. Otra trayectoria del nodo a al b es: <a, d><d, c><c, b>.
3. Otra trayectoria del nodo a al b es:

   <a, d><d, c><c, b><b, a><a, c><c, b>.
4. Las trayectorias 1 y 2 son simples.
5. La trayectoria 2 es hamiltoniana.
6. La longitud de las trayectorias 1, 2 y 3 son 2, 3 y 6 respectivamente.
7. La trayectoria <a, d><d, c><c, b><b, a> es un ciclo de longitud 4.
8. La gráfica es cíclica.
9. Los nodos c, b y d son adyacentes a a.

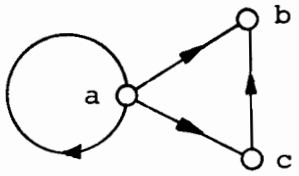
10. El grado externo de a es 3.

11. El grado interno de a es 2.

IV.1.2 REPRESENTACION DE GRAFICAS EN LA COMPUTADORA

Una gráfica G=(A, R) dirigida con $A=\{a_1, a_2, a_3, a_4, \ldots a_n\}$, puede ser completamente especificada con una matriz de orden n, llamada matriz de adyacencia. Para definir la matriz debe considerarse que cualquier elemento X_{ij} de la matriz es igual a 1, si el arco $\langle a_i, a_j \rangle$ existe en R, X_{ij} es igual a cero, si el arco no existe. (Véase figura IV. 11).

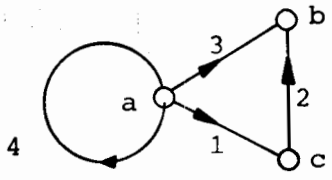
Matriz de adyacencia



	a	b	c
a	1	1	1
b	0	0	0
c	0	1	0

Figura IV.11 Representación matricial de una gráfica dirigida.

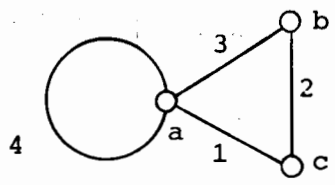
En una gráfica dirigida y pesada el elemento X_{ij} de la matriz de adyacencia es igual al peso del arco, si el arco existe en la gráfica y X_{ij} es igual a cero, si el arco no existe. (Véase figura IV.12).



	a	b	c
a	4	3	1
b	0	0	0
c	0	2	0

Figura IV.12 Representación matricial de una gráfica dirigida y pesada.

Una gráfica no dirigida, pesada o no, queda representada por una matriz triangular, ya que los arcos $<a_i, a_j>$ y $<a_j, a_i>$ quedan representados en un solo elemento X_{ij} de la matriz de adyacencia. (Véase figura IV.13).



	a	b	c
a	4	3	1
b	0	0	2
c	0	0	0

Figura IV.13 Representación matricial de una gráfica no dirigida.

Muchas representaciones matriciales de gráficas resultan ser matrices esparcidas y triangulares. Con el objeto de ahorrar memoria, es conveniente la utilización de las técnicas de representación de arreglos, vistas en la unidad II.

Representación ligada

Otras alternativas para representar gráficas en la computadora son la utilización de listas ligadas. Un posible formato es la utilización de una lista ligada con cabeza de lista para cada uno de los nodos de la gráfica. (Véase figura IV.14).

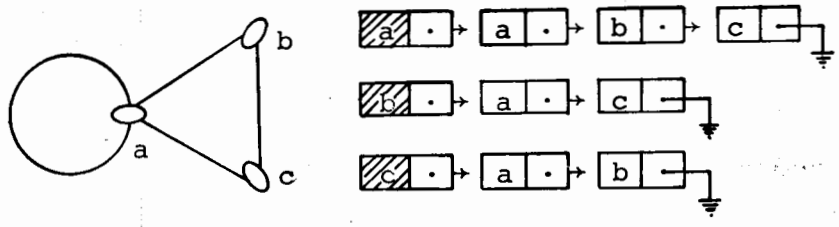


Figura IV.14 Representación ligada para una gráfica no dirigida.

Cada nodo fuera de la cabeza de lista, representa un arco de la gráfica, en particular el nodo c→ si se encuentra en la lista de a representa al arco ac, si se encuentra en b representa al arco bc. Cada arco está representado dos

veces en la estructura, una en la lista de uno de los nodos y la otra en la lista del otro nodo; ambos forman el arco. De esta forma, se requieren 2m nodos y n cabezas de lista, donde m es el número de arcos y n el número de nodos. (Para gráficas sin lazos).

En una gráfica dirigida sólo estarán en la lista de cualquier nodo aquellos nodos adyacentes a los que se tiene acceso desde él. Si la gráfica también es pesada, será necesario agregar un campo adicional al nodo para guardar el peso del arco. (Véase figura IV.15).

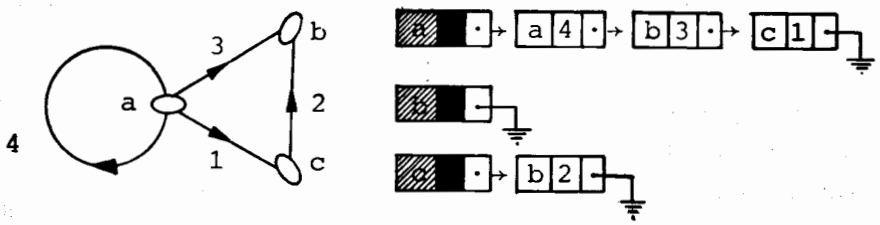


Figura IV.15 Representación ligada para una gráfica dirigida y pesada.

IV.2 ARBOLES

IV.2.1 CONCEPTOS Y DEFINICIONES

Un árbol es una gráfica G=(A,R) en la que:

1. El número de nodos es igual al número de arcos más uno $|A|=|R|+1$.
2. Todos los nodos son de grado interno uno, excepto un nodo llamado la raíz, de grado cero. — Definición de Arbol
3. No hay ciclos.
4. Cualquier trayectoria es simple.
5. Entre cualquier par de nodos sólo hay una trayectoria.
6. Cualquier arco, es un arco de desconección.

La definición proporciona las características que una gráfica debe seguir para ser un árbol. Otra forma de uso frecuente para definir un árbol es la siguiente definición recursiva:

Un árbol es un conjunto de uno o más nodos en el que hay un nodo especial, llamado la raíz del árbol, y los demás nodos son particiones en subconjuntos disjuntos T_1, T_2, T_3, ..., T_n ($n \geq 0$), cada uno de los cuales es un árbol. Cada T_i ($1 \leq i \leq n$) es llamado un subárbol de la raíz.

Consideremos la siguiente gráfica G = (A,R), donde

$$A = \{a, b, c, d, e, f\} \quad \text{y}$$

$$R = \{<a,b><a,c><a,d><c,e><c,f>\}$$

representada en la figura IV.16 y la definición de árbol dada inicialmente.

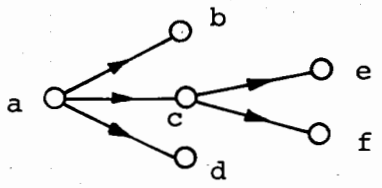


Figura IV.16 Gráfica dirigida R=(A,R).

Para verificar si la gráfica es un árbol, examinemos las particularidades de la definición.

1. El número de nodos, es efectivamente el número de arcos más uno: $6 = 5 + 1$.
2. Cualquier nodo es de grado interno 1 excepto el nodo (a) que es la raíz.
3. No existe ningún ciclo en la gráfica.
4. Las trayectorias son todas simples.
5. Entre dos nodos cualesquiera sólo hay una trayectoria.
6. Cualquier arco que se retire de la gráfica desconecta una parte de ella.

Al ser aplicables estos conceptos a la gráfica, podemos decir que G es un árbol.

La segunda definición aplicada sobre A = {a, b, c, d, e, f } de forma que:

1. si llamamos al nodo (a) la raíz del árbol y
2. los nodos restantes los particionamos en T_1 = { b }, T_2 = {c, e, f} y T_3 = { d }, obtenemos la gráfica intermedia mostrada en la figura IV.17.

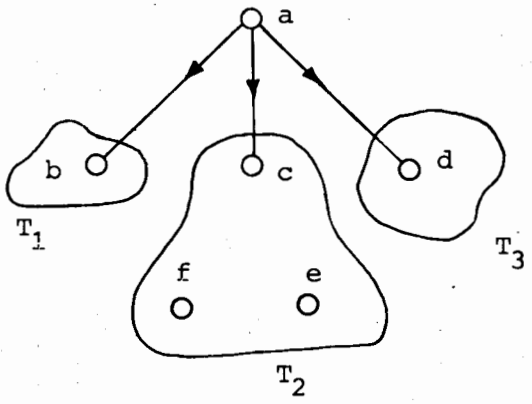


Figura IV.17 Estado del árbol al aplicarse por primera vez la definición.

3. La partición T_2 todavía no es un árbol. Si llamamos en esta partición al nodo c como la raíz y,
4. los nodos restantes los particionamos en T_4 = { e } y T_5 = { f }, las particiones resultantes son todas árboles. (Véase figura IV.18).

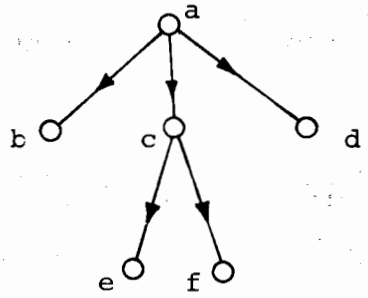


Figura IV.18 Arbol que resulta de la aplicación de la segunda definición de gráfica.

En la terminología que se emplea para el estudio de los árboles encontramos entre otros, los términos siguientes:

Se define como grado o grado externo de un nodo al número de sus subárboles.

Una hoja o nodo terminal es un nodo de grado cero.

Terminología de Arboles

Un nodo ramal es un nodo de grado mayor que cero.

El nivel de un nodo es el nivel de su antecesor directo más uno. El nivel de la raíz es 1.

También es frecuente que los nodos de un árbol reciban nombres, tales como: el nodo (a) es padre de b, c, d, o que b, c, d, son hijos de (a), o que b, c, d son hermanos.

Ejemplo IV.4

Consideremos al árbol de la figura IV.16 para el que definiremos los siguientes conceptos:

- el nodo (a) es la raíz del árbol
- los grados de los nodos a y c son tres y dos respectivamente
- los nodos d, b, e y f son hojas
- los nodos a y c son nodos ramales
- el nivel del nodo e es tres
- el nodo c es padre o antecesor directo de f y e
- los nodos d, c y b son hermanos

La representación del árbol utilizada en las figuras anteriores no es única. Existen diversas formas de representar un árbol, pero es posible que no muestren con claridad las relaciones entre los nodos y a esto se debe su poca utilización. Algunas de estas representaciones se muestran en la figura IV.19.

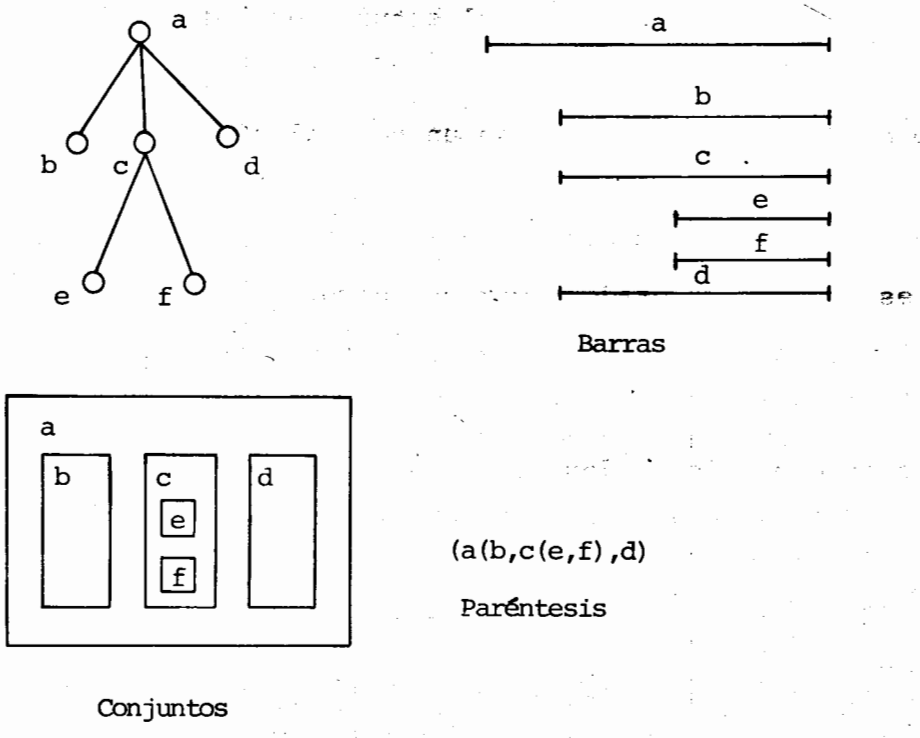


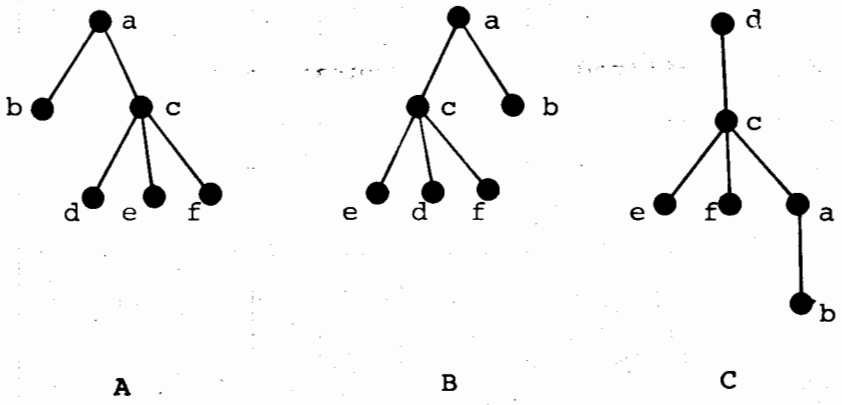
Figura IV.19 Representaciones alternas para un árbol.

Arboles ordenados, orientados y libres

Un árbol es ordenado cuando el orden de los subárboles es importante; cuando no se considera un orden para los subárboles, el árbol es orientado. En este último caso, si la dirección de los arcos se ignora, el árbol es libre.

Ejemplo IV.5

Consideremos los tres árboles siguientes:



Si los árboles son ordenados $A \neq B \neq C$

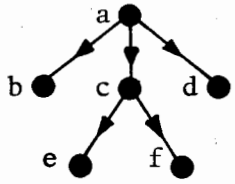
Si los árboles son orientados $A = B \neq C$

Si los árboles son libres $A = B = C$

IV.2.2 REPRESENTACION DE ARBOLES EN LA COMPUTADORA

Un árbol puede ser completamente especificado por una matriz de adyacencia de forma similar a una gráfica dirigida. Para especificar el nodo raíz, es posible utilizar los elementos de la diagonal principal que siempre estarán disponibles. (Véase figura IV.20).

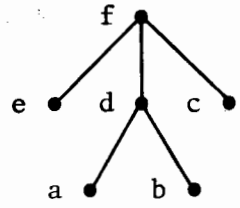
Matriz de adyacencia para Arboles



	a	b	c	d	e	f
a	●	1	1	1	0	0
b	0	0	0	0	0	0
c	0	0	0	0	1	1
d	0	0	0	0	0	0
e	0	0	0	0	0	0
f	0	0	0	0	0	0

Figura IV.20 Representación matricial de un árbol.

Cuando el árbol es libre, la matriz de adyacencia es triangular, y se especifica en forma similar a una gráfica no dirigida. (Véase figura IV.21).



	a	b	c	d	e	f
a	0	0	0	1	0	0
b	0	0	0	1	0	0
c	0	0	0	0	0	1
d	0	0	0	0	0	1
e	0	0	0	0	0	1
f	0	0	0	0	0	●

Figura IV.21 Representación para un árbol libre.

Otra alternativa es la utilización de un arreglo unidimensional en el que, cada elemento represente a cada uno de los nodos del árbol y su contenido a su antecesor directo. (Véase figura IV.22).

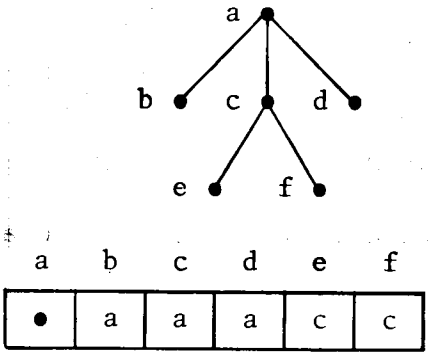


Figura IV.22 Representación vectorial para un árbol.

IV.3 ARBOLES BINARIOS

Un árbol particular que tiene gran importancia en el área de las ciencias de la computación es el árbol binario. Esto se debe fundamentalmente a la sistematización que puede lograrse para su representación.

IV.3.1 DEFINICIONES

Arbol de Knuth y estrictamente binario

Un árbol binario es un árbol en el que cualquier nodo tiene cero, uno o dos subárboles. Cuando tiene exactamente cero o dos subárboles es llamado árbol estrictamente binario, de otra forma, es un árbol de Knuth. (Véase figura IV.23).

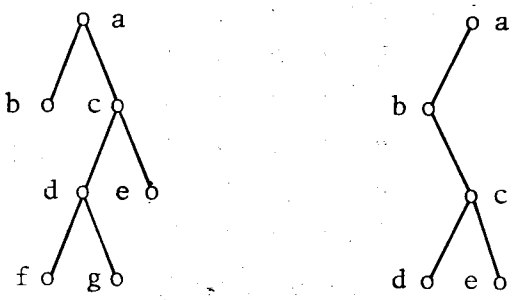


Figura IV.23 Arboles binarios.

Un árbol binario estricto es balanceado cuando el número de nodos terminales, n, es igual a 2↑m, y la longitud de cualquier trayectoria de la raíz a cualquier nodo terminal es m, donde m es cualquier número entero no negativo; o si 2↑m<n<2↑(m+1) y la longitud de las trayectorias son m ó m+1. (Véase figura IV.24).

Arbol balanceado

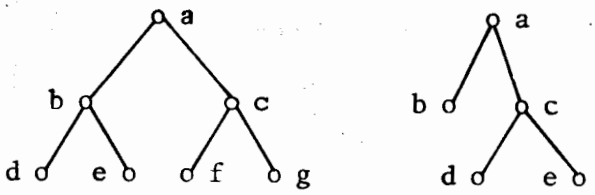


Figura IV.24 Arboles balanceados.

El árbol es completo cuando es balanceado y n = 2↑m. Al considerar un orden para los subárboles, inclusive cuando sólo existe uno de ellos, el árbol es B. Un árbol B tiene asociado en los arcos un peso cuyo valor es cero, cuando el arco está orientado a la izquierda y uno cuando está a la derecha. Dos arcos que se originan en el mismo nodo no pueden tener la misma orientación. (Véase figura IV.25).

Arbol completo

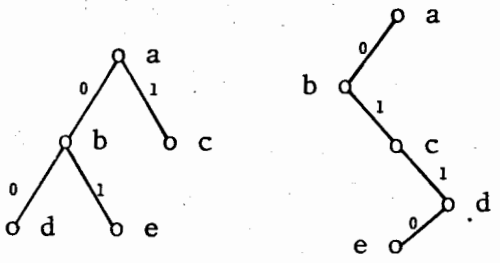


Figura IV.25 Arboles B.

IV.3.2 TRANSFORMACION DE ARBOLES A ARBOLES BINARIOS

Un árbol de grado > 2 puede ser transformado a su árbol binario estricto o de Knuth equivalente.

Un bosque es un conjunto de árboles disjuntos, (véase figura IV.26), y puede ser transformado a un árbol de Knuth con el uso del algoritmo siguiente:

1. Ligar las raices de los árboles del bosque y seleccionar a la raíz del árbol a la izquierda como la raíz del nuevo árbol.
2. Ligar a todos los hermanos de cada padre, y
3. Retirar todas las ligas de un padre a sus hijos excepto la del hijo a la izquierda.

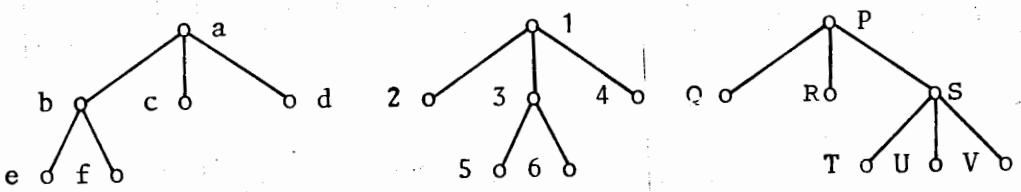
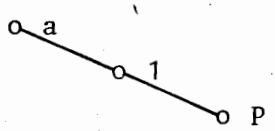


Figura IV.26 Bosque de árboles de grado > 2.

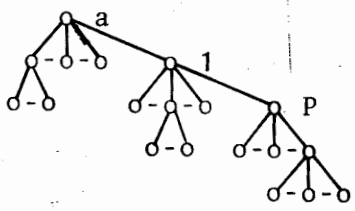
Ejemplo IV.6

Aplicar el algoritmo para transformar el bosque de la figura IV.26 a un árbol de Knuth.

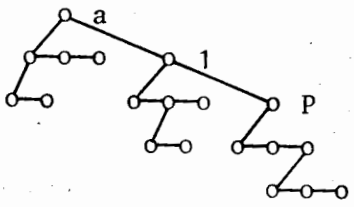
Paso 1.



Paso 2.



Paso 3.



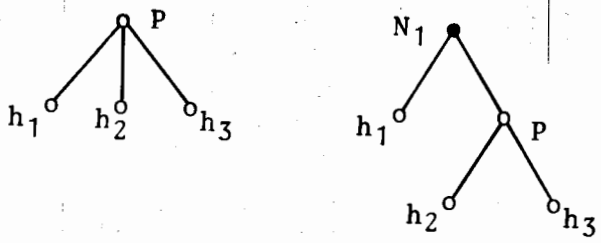
Para transformar un árbol a un árbol estrictamente binario, deberán considerarse los pasos del algoritmo siguiente:

1. Se selecciona la raíz del árbol original, se genera un nuevo nodo cuyo hijo izquierdo es el subárbol izquierdo de la raíz, y el derecho lo que queda del árbol original incluyendo la raíz.
2. Este proceso se repite recursivamente sobre los subárboles del nuevo árbol, hasta que los nodos de los subárboles sean transformados a nodos terminales.

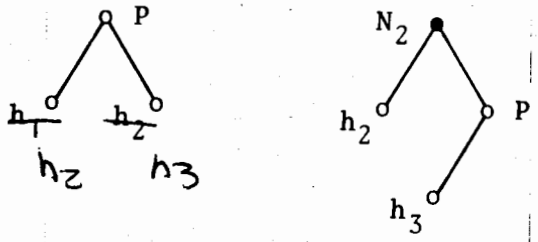
Ejemplo IV.7

Transformar el árbol siguiente a su equivalente árbol binario estricto.

Paso 1. Tomemos P como el nodo inicial y N1, como el nuevo nodo generado.

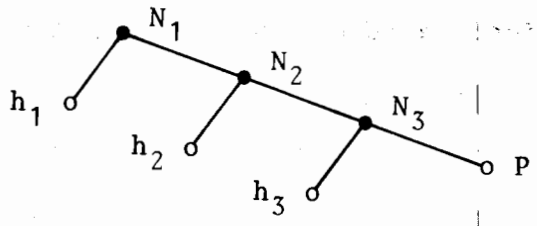


Paso 2. De las nuevas partes generadas una de ellas sólo tiene nodos terminales, mientras que la otra todavía no alcanza esta condición. Tomemos nuevamente P.



Paso 3. Una de las partes tiene nodos terminales, pero la otra no. Tomemos P.

Paso 4. Finalmente ambas partes contienen nodos terminales y el árbol estrictamente binario es:



IV.3.3 RECORRIDO DE ARBOLES

Recorrer un árbol, es un método de visitas de los nodos con el objeto de sistematizar la recuperación de la información almacenada en los nodos de un árbol.

Una de las formas más simples de recorrer un árbol de arriba hacia abajo (top-down), es iniciar las visitas por la raíz y continuar sobre los nodos del nivel 2 de izquierda a derecha, continuar sobre los del nivel 3, hasta alcanzar el nivel n.

TOP-DOWN
BOTTOM-UP

Cuando procedemos desde el nivel n hacia la raíz, el recorrido es de abajo hacia arriba (bottom-up). (Véase figura IV.27).

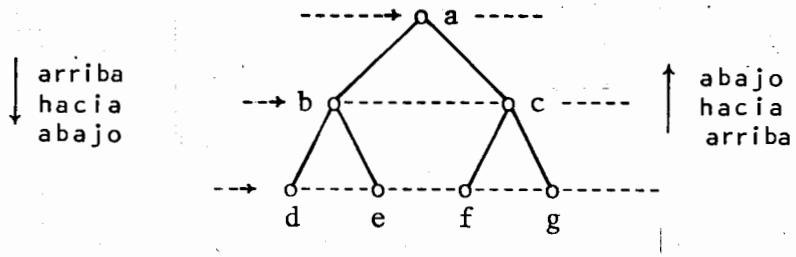


Figura IV.27 Recorridos nivel por nivel.

Los recorridos también pueden practicarse sistematizando la visita de los subárboles. (Véase figura IV.28).

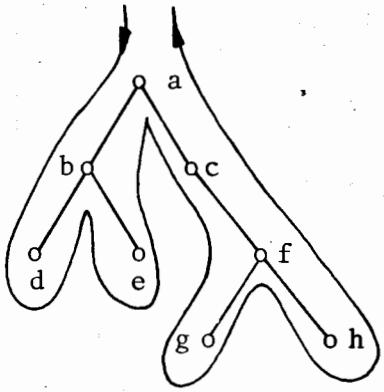


Figura IV.28 Recorrido de un árbol.

Cuando los recorridos se practican de acuerdo a la trayectoria que se muestra en la figura IV.28, podemos ver que la información contenida en un nodo puede ser recuperada antes de visitar sus subárboles, después de visitar alguno de sus subárboles o después de visitar los dos subárboles. Estas formas de recuperar la información de los nodos de un árbol caracterizan a los recorridos llamados: Preorder, Inorder y Postorder respectivamente.

PREORDER
INORDER
POSTORDER

Los algoritmos que realizan estos recorridos pueden ser postulados en forma recursiva o no recursiva. Por facilidad, usaremos una definición recursiva para los algoritmos.

RECORRIDO PREORDER

En forma abreviada:
- se visita el nodo
- se recorre el subárbol izquierdo
- se recorre el subárbol derecho

```
FUNCION<PREORDER>
(
SE VISITA EL NODO
SI EL SUBARBOL IZQUIERDO EXISTE Y NO SE HA VISITADO
                          LLAMADA A PREORDER
FIN
SI EL SUBARBOL DERECHO EXISTE Y NO SE HA VISITADO
                          LLAMADA A PREORDER
FIN
RETORNAR
)
```

RECORRIDO INORDER

En forma abreviada:
- se recorre el subárbol izquierdo
- se visita el nodo
- se recorre el subárbol derecho

```
FUNCION<INORDER>
(
SI EL SUBARBOL IZQUIERDO EXISTE Y NO SE HA VISITADO
                          LLAMADA A INORDER
FIN
SE VISITA EL NODO
SI EL SUBARBOL DERECHO EXISTE Y NO SE HA VISITADO
                          LLAMADA A INORDER
FIN
RETORNAR
)
```

RECORRIDO POSTORDER

En forma abreviada:
- se recorre el subárbol izquierdo
- se recorre el subárbol derecho
- se visita la raíz

```
FUNCION<POSTORDER>
(
SI EL SUBARBOL IZQUIERDO EXISTE Y NO SE HA VISITADO
                          LLAMADA A POSTORDER
FIN
SI EL SUBARBOL DERECHO EXISTE Y NO SE HA VISITADO
                          LLAMADA A POSTORDER
FIN
SE VISITA EL NODO
RETORNAR
```

Ejemplo IV.8

Consideremos el árbol de la figura IV.28, para él, la visita de los nodos en los modos vistos sería:

NIVEL/NIVEL(top-down)	a b c d e f g h
NIVEL/NIVEL(bottom-up)	g h d e f b c a
PREORDER	a b d e c f g h
INORDER	d b e a c g f h
POSTORDER	d e b g h f c a

IV.3.4 REPRESENTACION EN LA COMPUTADORA

Representación vectorial de un Arbol binario

Un árbol binario, puede ser representado en la computadora con su matriz de adyacencia o con un arreglo unidimensional, como ya se ha mencionado en párrafos anteriores. Otra alternativa es la utilización de un arreglo unidimensional, en el que los hijos de cualquier nodo k estén en el elemento 2k, el izquierdo, y el 2k + 1, el derecho. La longitud del arreglo es de 2↑n - 1 en donde n es el número de niveles del árbol. (Véase figura IV.29).

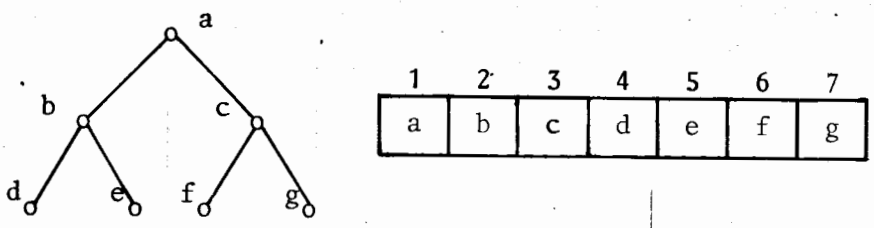


Figura IV.29 Representación vectorial para un árbol binario.

Representación ligada de un Arbol binario

Otra forma de representar un árbol binario es con la utilización de listas ligadas. Una de las representaciones directas es la utilización de un nodo con tres campos, uno para el dato, uno para la liga izquierda y el otro para la liga derecha. (Véase figura IV.30).

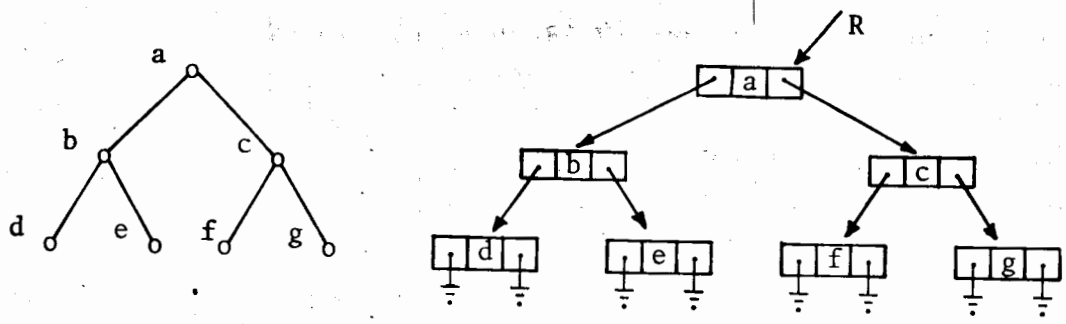


Figura IV.30 Representación ligada de un árbol binario.

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

1.	Lista no lineal	( ) A x B
2.	Gráfica dirigida	( ) Los arcos de la gráfica tienen un valor asociado llamado peso.
3.	Producto cartesiano	( ) Es un árbol en el que los nodos son de grado 0, 1, ó 2.
4.	Relación	( ) Nodo de un árbol con grado cero.
5.	Grado interno de un nodo	( ) Las relaciones entre los nodos de la lista no están dadas por un solo criterio.
6.	Grado externo de un nodo	( ) Es importante en la gráfica que <u, v> sea diferente que < v, u > .
7.	Trayectoria	( ) Es un árbol en el que < u, v > es igual que < v, u > .
8.	Ciclo	( ) Gráfica sin ciclos con todos los nodos de grado interno 1 excepto un nodo, llamado raíz.
9.	Arco no dirigido	( ) Es un subconjunto de A x B.
10.	Gráfica pesada	( ) No importa el orden de sus subárboles.
11.	Arbol	( ) Número de arcos que parten de un nodo.
12.	Raíz de un árbol	( ) < u, v > es igual que < v, u >.
13.	Nodo ramal	( ) Nodo con grado externo igual a cero.
14.	Hojas	( ) Arcos de la forma < a1, a2 > < a2, a3 > < a3, a4 > ...
15.	Arbol ordenado	( ) Número de arcos que llegan a un nodo.
16.	Arbol orientado	( ) Nodos con grado externo mayor que cero.

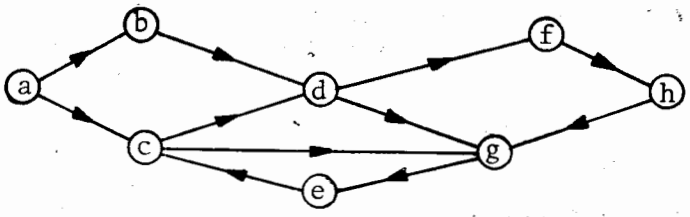
17. Arboles libres ( ) Trayectoria en la que $a_1 = a_n$.

18. Arboles binarios ( ) Es importante el orden de los subárboles.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

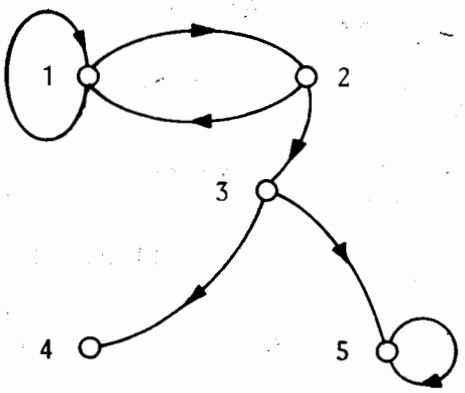
1. Considere la siguiente gráfica y describa:

   a. una trayectoria de (a) a (h)
   b. una trayectoria hamiltoniana
   c. una trayectoria $|P| = 6$ de (a) a (d)
   d. un ciclo
   e. los nodos de grado externo 1
   f. los nodos de grado interno 2



2. Encuentre las matrices de adyacencia para las gráficas siguientes:

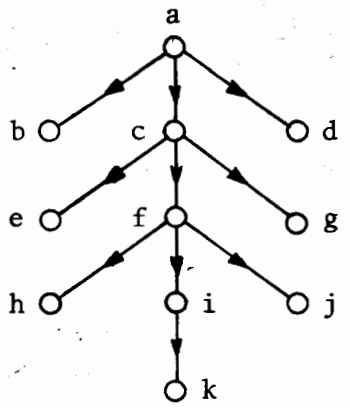
   a. A = (a, b, c, d) y R = A x A
   b. A = (1, 2, 3, 4, 5, 6) y R = mayor que (>)
   c.



3. Escriba un programa que a partir de la matriz de adyacencia para una gráfica dirigida, encuentre una trayectoria entre dos nodos.

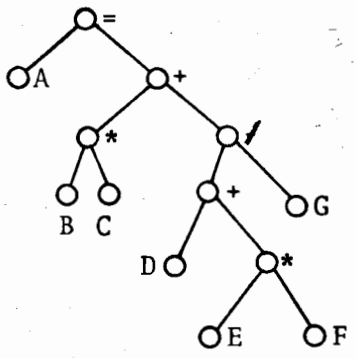
4. Considere el siguiente árbol:

	F	V
- (k) es la raíz del árbol	( )	( )
- (e) es de grado cero	( )	( )
- (a) es de grado tres	( )	( )
- b, e y h son hermanos	( )	( )
- el nivel de (f) es dos	( )	( )
- (b) y (d) no son hojas	( )	( )
- es un árbol binario	( )	( )
- sólo (a) es nodo ramal	( )	( )
- es un árbol libre	( )	( )

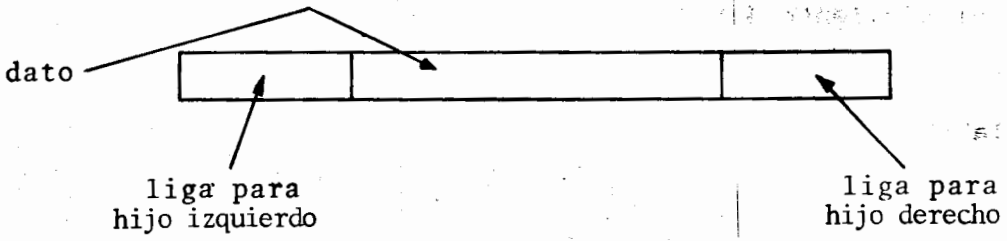


5. Transforme el árbol del problema anterior a un árbol estrictamente binario y a un árbol de Knuth.

6. Recorra el árbol siguiente, nivel por nivel (top-down y bottom-up), preorder, inorder y postorder.



7. Escriba un algoritmo para recorrer un árbol binario en modo postorder. El árbol está almacenado en una estructura cuyos nodos tienen el formato siguiente:



UNIDAD V ARCHIVOS

OBJETIVO GENERAL

El alumno comprenderá las organizaciones básicas de los archivos, las operaciones que se pueden realizar sobre ellos y su representación en diferentes medios de almacenamiento secundario.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Reconocerá qué es un archivo y las operaciones que se realizan en ellos.
2. Diferenciará la organización lógica y física de los archivos en cinta magnética y en disco magnético.
3. Señalará los métodos de acceso a los archivos lógicos y físicos.
4. Describirá la utilización del sistema de archivos.

INTRODUCCION

Para el estudio de los archivos usaremos los conceptos expuestos en la unidad I, referentes a la organización física y lógica de la memoria.

La unidad se inicia con una presentación en la que se comenta la importancia de los archivos en los sistemas de cómputo; a continuación se da una definición y se presentan las operaciones más comunes realizadas sobre los archivos.

Posteriormente se explica la organización de los archivos al nivel lógico y al nivel físico, haciendo referencia en este último caso a la cinta magnética y al disco magnético. Finalmente se analizan los métodos de acceso a los archivos lógicos y físicos, así como la función del sistema de archivos.

V.1 GENERALIDADES

El estudio de los archivos es importante dentro de las estructuras de datos, pues a través de ellos se puede dar un sentido a los datos almacenados en la memoria secundaria, además de permitir la organización de la información que será guardada en forma permanente.

El establecimiento de los archivos está íntimamente ligado a la idea de guardar información organizada por largo tiempo en medios de almacenamiento masivo, de tal forma que sólo cuando se necesita esta información, es copiada en la memoria primaria.

La información transferida a la memoria primaria se almacena en un área temporal llamada buffer, de donde el programa de usuario toma los datos que necesita. De la misma forma cuando se van a guardar datos, primero se depositan en el buffer y de ahí son pasados a la memoria secundaria. (Véase figura V.1). Buffer

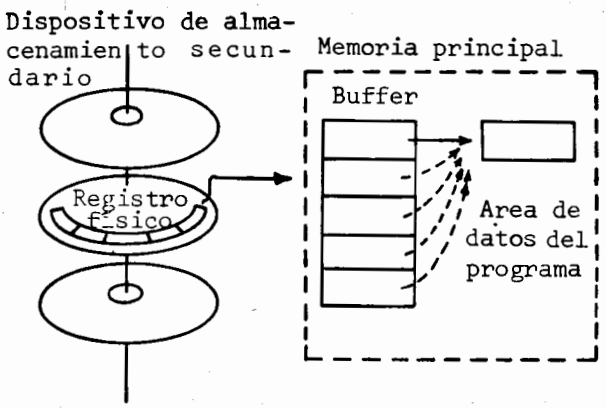


Figura V.1 Uso del Buffer en la lectura de datos.

V.2 DEFINICION Y OPERACIONES

Un archivo es la unidad básica de almacenamiento de información a largo plazo y está formado por un conjunto de elementos de información o registros que tienen significado para el usuario.

El estudio de los archivos puede dividirse en dos niveles: lógico y físico. El nivel lógico se refiere a la forma como el usuario ve el archivo y el nivel físico a la forma como el archivo se encuentra almacenado en la memoria secundaria, pudiéndose establecer diferentes organizaciones de archivos y métodos de acceso en ambos niveles. Entonces, se debe contar con alguna interfaz que permita al usuario manipular sus archivos independientemente de la forma como éstos se almacenan en la memoria secundaria. Esto se logra con varios programas denominados sistema de archivos, que forman parte del sistema operativo, a través de los cuales se puede establecer una correspondencia entre los archivos lógicos y los archivos físicos. (Véase figura V.2).

Interfaz

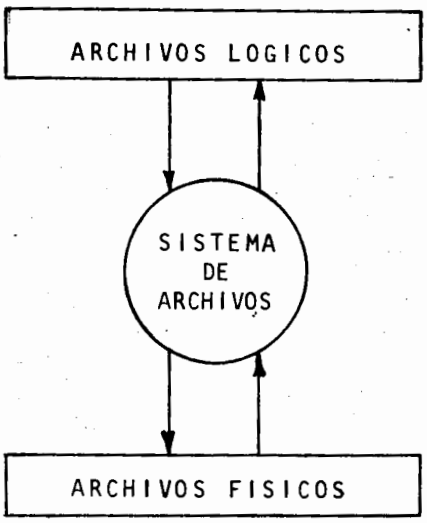


Figura V.2 Interfaz entre los archivos lógicos y los archivos físicos.

Lectura y escritura de registros

Las operaciones básicas que se pueden realizar sobre un archivo son la lectura y escritura de registros consistentes en la transmisión de información de la memoria secundaria a la memoria primaria para la lectura, y en la transmisión de información de la memoria primaria a la memoria secundaria en la escritura.

La transmisión de información antes mencionada se lleva a cabo en unidades de tamaño fijo denominadas bloques. En muchcs casos el tamaño de los bloques corresponde al de los registros físicos por lo que se les usa en forma indistinta. Asimismo, es común que en un registro físico se almacenen varios registros lógicos de acuerdo al factor de bloqueo usado.

V.3 ORGANIZACION DE ARCHIVOS

Se analizarán las diversas formas de organización de los archivos lógicos y físicos.

V.3.1 ORGANIZACION LOGICA

El usuario siempre ve al archivo formado por un conjunto de registros contiguos, uno a continuación de otro. Existe diferencia entre los archivos debido al tipo de registros que los constituyen y a la forma de acceso a los mismos.

Tipos de registros lógicos

Un archivo puede estar constituido por registros sin llave o con llave y a su vez ser de longitud fija o longitud variable.

Los archivos con registros sin llave son vistos por el usuario como una secuencia de elementos lógicos del mismo tamaño (registros de longitud fija) o de diferente tamaño (registros de longitud variable). (Véase figura V.3).

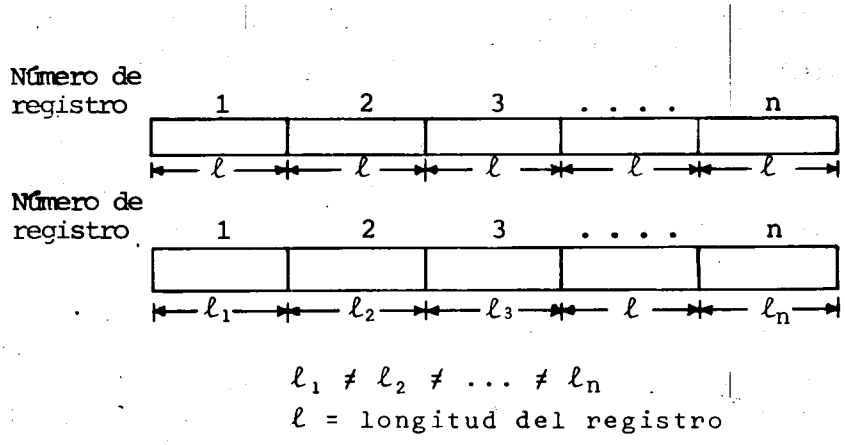
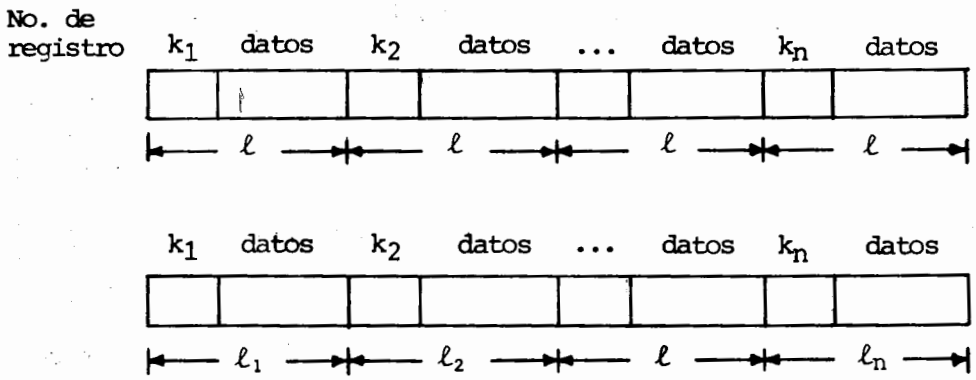


Figura V.3 Archivos de registros sin llave de longitud fija y longitud variable.

Los archivos constituidos por registros con llave utilizan en forma explícita un identificador para cada registro y son vistos por el usuario como una secuencia de elementos lógicos, cada uno de los cuales está precedido por su llave, pudiendo a su vez ser de longitud fija o longitud variable. (Véase figura V.4).



ki = llave del registro i

Figura V.4 Archivos de registros con llaves de longitud fija y longitud variable.

V.3.2 ORGANIZACION FISICA

Tipos de registros físicos

En la organización física los registros son de tamaño fijo o de tamaño variable y pueden organizarse de varias formas para constituir archivos físicos, los cuales pueden residir en cinta magnética o disco magnético.

CINTA MAGNETICA

Archivo en cinta magnética

En una cinta magnética un archivo físico está formado por un conjunto de registros físicos, al final de los cuales se encuentra un caracter delimitador de fin de archivo.

Los bloques que constituyen un archivo siempre están organizados en forma contigua, ya que se asignan en forma consecutiva. Las restricciones físicas de este medio de almacenamiento impiden la realización práctica de otras formas de organización de archivos.

Tipos de etiquetas en cinta magnética

Los archivos en cinta pueden contener etiquetas que permiten tener un mayor control sobre la información almacenada. En general pueden clasificarse como etiquetas de volumen, etiquetas de archivo y etiquetas de usuario.

Las etiquetas de volumen contienen información que permiten identificar la cinta, el nombre del propietario y cualquier otra información de tipo general que pueda requerirse para la utilización adecuada de la cinta.

Las etiquetas de archivo se utilizan por pares para indicar el inicio y el fin del archivo, conteniendo información acerca del nombre del archivo, fecha de creación, sistema operativo empleado, etc.

Las etiquetas de usuario sirven para guardar información adicional de importancia para el usuario y, aunque son leídas y escritas por el sistema operativo, no son procesadas por éste. (Véase figura V.5).

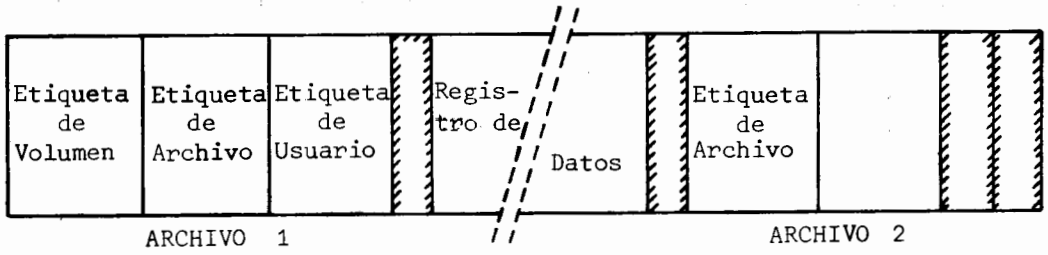


Figura V.5 Uso de etiquetas en cinta magnética.

DISCO MAGNETICO

Archivo en disco magnético

Un archivo físico en disco es una colección de registros físicos del mismo tamaño, mismos que pueden estar organizados en forma contigua, ligada o con una tabla de mapeo.

Organización contigua

En la organización contigua, el archivo utiliza registros físicos contiguos, siguiendo la secuencia normal de direcciones, por ejemplo, registros sobre la misma pista. (Véase figura V.6).

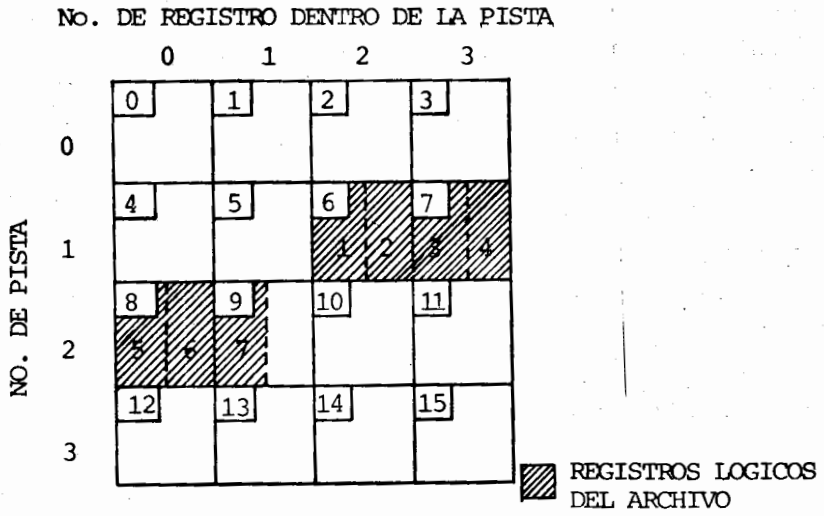


Figura V.6 Organización contigua de registros.

En este caso, para saber qué registros físicos forman el archivo, se necesita conocer la dirección del primer registro y el número de registros contiguos.

La organización ligada consiste de un conjunto de registros físicos, cada uno de los cuales tiene un campo destinado para indicar la dirección del siguiente registro. (Véase figura V.7).

Organización ligada

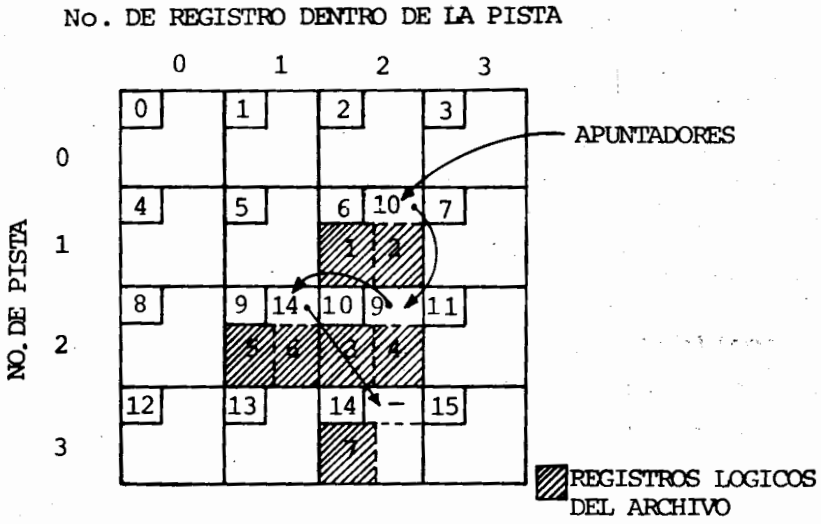


Figura V.7 Organización de registros ligados.

Para conocer qué registros forman el archivo se necesita un apuntador al primer registro físico del archivo.

Organización en tabla de mapeo

Otra forma de organización es la tabla de mapeo, ésta consiste en una tabla o lista de apuntadores a los registros físicos que forman el archivo. Para este caso puede considerarse que se han extraído todos los campos de liga de los registros y que se construye una tabla con ellos. (Véase figura V.8).

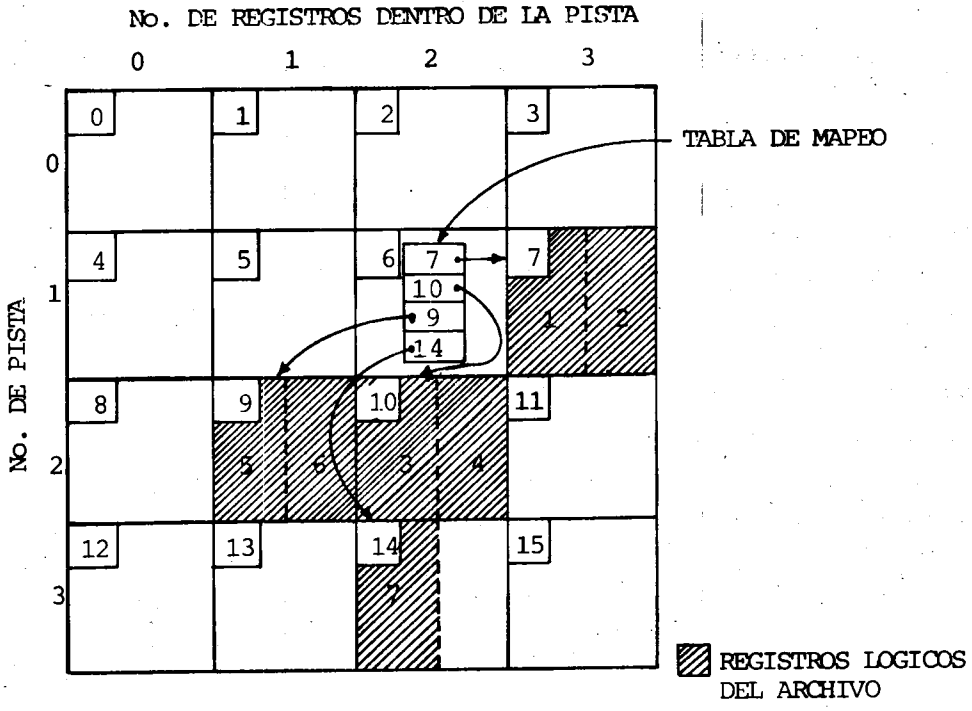


Figura V.8 Organización de registros con tabla de mapeo.

Para conocer qué registros forman el archivo, se necesita un apuntador al primer registro de la tabla.

V.4 ACCESO A ARCHIVOS

Una vez establecido cómo se organizan los registros que forman un archivo, analizaremos el acceso a los mismos, esto es, la forma de llegar al nodo deseado para leer o escribir información.

V.4.1 ACCESO LOGICO

El acceso a los registros que van a ser procesados en un archivo lógico se lleva a cabo usando dos métodos: acceso secuencial y acceso directo.

Para identificar a los registros lógicos que forman el archivo, se asigna a cada uno de ellos un número de acuerdo a la posición relativa que ocupan con respecto al inicio del archivo.

Acceso secuencial lógico

El acceso secuencial consiste en la obtención de los registros en forma consecutiva utilizando el número asociado a cada uno de ellos.

Este tipo de acceso se lleva a cabo sobre archivos formados por registros sin llave, con llave, de longitud fija o longitud variable.

El procesamiento de los registros de un archivo en forma secuencial es recomendable cuando se va a realizar la misma operación sobre todos los registros del archivo, y no se requiere de un tiempo de respuesta breve.

Acceso directo lógico

El acceso directo a los registros que forman un archivo lógico se lleva a cabo en cualquier orden. Esto quiere decir que se puede obtener un registro sin importar cuál ha sido procesado o cuál se procesará después.

En los archivos formados por registros sin llave (longitud fija o longitud variable) el acceso se realiza utilizando el número del registro como un índice, mientras que en los archivos formados por registros con llave (longitud fija o longitud variable), el acceso se realiza utilizando la llave del registro como índice.

El uso de los registros de un archivo en forma directa es recomendable cuando se desea tener un tiempo de respuesta breve, independientemente del número de registros que vayan a ser procesados.

V.4.2 ACCESO FISICO

El acceso a los registros de un archivo físico se realiza mediante dos métodos: acceso secuencial y acceso directo. La diferencia con los métodos lógicos del mismo nombre radica en que el acceso físico usa las direcciones físicas de los registros, mientras que el acceso lógico usa las direcciones relativas.

CINTA MAGNETICA

En este tipo de memoria secundaria se da acceso a los registros en forma secuencial, uno después de otro, de acuerdo a su posición física en la cinta. Esto se debe a las restricciones de uso de este medio de almacenamiento que no permiten otras formas de acceso.

DISCO MAGNETICO

Este tipo de memoria secundaria tiene características, que permiten dar acceso a los registros físicos de un archivo en forma secuencial o en forma directa.

Acceso secuencial físico

El acceso secuencial consiste en la obtención de los registros físicos para su procesamiento de acuerdo a su posición relativa a la dirección de inicio en el archivo.

El modo de acceso secuencial puede llevarse a cabo en cualquiera de las tres organizaciones de archivo.

En la organización contigua, el acceso se lleva a cabo usando la dirección del primer registro físico del archivo, para procesar los registros en la secuencia de direcciones normal.

En el caso de la organización ligada, el acceso se lleva a cabo usando la dirección del primer registro físico del archivo y siguiendo la secuencia indicada por las direcciones guardadas en el campo de liga de los registros que forman el archivo.

La organización en tabla de mapeo utiliza la dirección de inicio de la tabla para recorrerla y de esta manera procesar los registros en la secuencia en la que aparecen dentro del archivo.

Acceso directo físico

El acceso directo permite obtener cualquier registro del archivo sin importar el orden de procesamiento. Este tipo de acceso puede realizarse sobre archivos organizados en forma contigua o en tabla de mapeo.

Para la organización contigua, el acceso se lleva a cabo utilizando el número físico de registro como índice.

En la organización con la tabla de mapeo es necesario realizar dos accesos: el primero de ellos, a la tabla de mapeo para obtener la dirección física del registro y, el segundo, para obtener el registro especificado.

V.5 SISTEMA DE ARCHIVOS

Como se mencionó anteriormente, el sistema de archivos tiene como funciones principales transformar una solicitud de acceso a un archivo lógico en un acceso a un archivo físico. El sistema de archivos cuenta con diferentes módulos que permiten realizar esta interfaz, los más importantes son: el sistema de archivos simbólicos, el sistema de archivos básicos, el sistema de archivos lógicos, el sistema de archivos físicos y el sistema de manejo del dispositivo.

Para explicar la forma como estos módulos transforman la información, asumiremos que la solicitud de acceso a un archivo realizada por un usuario indica: una operación sobre un archivo (lectura o escritura), el nombre del archivo, el número de registro lógico sobre el que se efectúa la operación y una dirección de memoria de donde la información será leída o escrita.

La función del sistema de archivos simbólicos consiste en transformar el nombre simbólico del archivo a un identifi

cador interno. Esto se logra usando un directorio de archivos simbólicos en donde aparezcan en la primera columna, los nombres simbólicos de los archivos y, en la segunda los identificadores correspondientes a cada uno de ellos.

Sistema de archivos simbólicos

El sistema de archivos básicos utiliza el identificador asociado a cada archivo como un índice para dar acceso al directorio de archivos básicos, en donde se encuentra la información acerca del tamaño de registro lógico, el número de registros lógicos que forman el archivo y la dirección de inicio del primer registro físico. Para cada archivo en uso, esta información se copia a la memoria primaria y permanece ahí durante el tiempo de procesamiento del archivo.

Sistema de archivos básicos

El sistema de archivos lógicos transforma la solicitud de acceso a un registro lógico en una solicitud equivalente para accesar una cadena de bytes. Para esto se necesita obtener la dirección lógica de la cadena y la longitud de la misma, las cuales pueden calcularse utilizando el número de registro lógico y la longitud del mismo.

Sistema de archivos lógicos

El sistema de archivos físicos convierte la dirección lógica de la cadena y su longitud en un número físico de registro y un desplazamiento dentro de este registro, que corresponden a la dirección de inicio de la cadena de bytes.

Sistema de archivos físicos

Finalmente, el módulo que maneja al dispositivo, transforma el número de bloque físico en una dirección física, de acuerdo al formato requerido por el dispositivo y realiza la operación solicitada.

Sistema de manejo del dispositivo

Ejemplo V.1

Considere un archivo, llamado prueba que tenga diez registros lógicos de longitud fija y cada uno de ellos sea de cien bytes. Este archivo se organizaría físicamente en forma contigua, utilizando registros físicos de quinientos bytes. Entonces el bloqueo apropiado sería de cinco registros lógicos en uno físico, ocupando el archivo físi

co un total de dos registros. La memoria secundaria, en la que se almacenaría este archivo, sería un disco que utilizaría una sola superficie con seis pistas y cuatro registros por pistas. Esto daría un total de veinticuatro registros numerados del cero al veintitres, como se muestra a continuación:

NUMERO DE REGISTRO DENTRO DE LA PISTA

NUMERO DE PISTA	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19
5	20	21	22	23

Ahora analizaremos la forma como el sistema de archivos intervendría y transformaría la información cuando un usuario solicite una operación de lectura del registro lógico número nueve.

Primero. Solicitud del usuario.

Leer archivo (prueba) registro (9) en localidad (14000).

Segundo. Sistema de archivos simbólicos.

El sistema de archivos simbólicos utilizará el directorio de archivos simbólicos para transformar el nombre prueba en un identificador interno. Tal como se muestra a continuación:

Archivo simbólico	Identificador interno
XYZ	3
Prueba	2
Maestro	6

Tercero. Sistema de archivos básicos.

El sistema de archivos básicos utilizará el identificador interno 2, para dar acceso al directorio de archivos básicos y copiar la información correspondiente al archivo en la memoria principal.

IDENTIFICADOR INTERNO	TAMAÑO DE REGISTRO LOGICO	REGISTROS LOGICOS QUE FORMAN EL ARCHIVO	DIRECCION DE INICIO DEL PRIMER R.F.
2	100	10	9
3	500	1	20
4			
5			
6	50	10	4

Cuarto. Sistema de archivos lógicos.

El sistema de archivos lógicos tomará como entrada el número de registro lógico (NRL) y el tamaño del mismo (TRL) para dar como salida la dirección lógica de la cadena de bytes (DLCB) y el tamaño de la misma (TCB). Para esto utilizará la información del directorio de archivos básicos en las siguientes fórmulas:

$$DLCB = (NRL - 1) * TRL$$
$$TCB = TRL$$

Entonces, sustituyendo los valores del ejemplo, obtendremos:

$$DLCB = (9 - 1) * 100 = 8 * 100 = 800$$
$$TCB = 100$$

Quinto. Sistema de archivos físicos.

El sistema de archivos físicos tomará como entrada la DLCB, el tamaño del registro físico (TRF) y la dirección del primer registro físico (DPRF) para dar como salida el número de registro físico (NRF) y el desplazamiento (D). La trans

formación se llevará a cabo utilizando las siguientes fór mulas:

$$NRF = (DLCB \text{ DIV } TRF) + DPRF$$
$$D = (DLCB \text{ MOD } TRF)$$

Sustituyendo los valores del ejemplo obtendremos:

$$NRF = (800 \text{ DIV } 500) + 9 = 1 + 9 = 10$$
$$D = (800 \text{ MOD } 500) = 300$$

Sexto. Sistema de manejo del dispositivo.

Este módulo tomará como entrada el NRF y lo transformará en una dirección compuesta por el número de pista (NP) y el número de registro dentro de la pista (NRDP).

Para esto utilizará el número total de pistas (NTP) y el número total de registros en una pista (NTRP), de acuerdo a las siguientes fórmulas:

$$NP = (NRF \text{ DIV } NTRP)$$
$$NRDP = (NRF \text{ MOD } NTRP)$$

en donde se podrán sustituir los valores para este ejemplo, quedando las siguientes expresiones:

$$NP = (10 \text{ DIV } 4) = 2$$
$$NRDP = (10 \text{ MOD } 4) = 2$$

Con esta información se construirá una instrucción para que la unidad de disco lea el segundo registro físico en la pista dos y lo transfiera a un buffer en la memoria principal.

Una vez que se tenga el registro físico en la memoria principal, el sistema operativo utilizará el desplazamien to calculado anteriormente para extraer la información co rrespondiente al registro lógico del usuario y copiarla a partir de la dirección 14 000.

Ejemplo V.2

Se desean especificar las características de un archivo para guardar en la computadora un directorio de tres mil clientes, a los que se enviará información periódicamente.

Para esto será necesario hacer las siguientes consideraciones:

Cada vez que se envíe la información se procesará todo el archivo sin importar el orden, por lo que se puede utilizar una organización secuencial.

Cada registro contendrá la información concerniente a una persona, los campos y su tamaño son los siguientes:

DESCRIPCION DEL CAMPO	TAMAÑO (CARACTERES)
1. Nombre del cliente	40
2. Puesto y categoría	50
3. Calle	20
4. Número exterior	6
5. Número interior	4
6. Colonia	25
7. Población	50
8. Código postal	5

De acuerdo al número de campos y de su longitud el tamaño del registro lógico es de 200 bytes. Entonces el tamaño del archivo será de tres mil registros de doscientos caracteres cada uno, lo que da un total de seiscientos mil caracteres en el archivo.

La organización lógica de este archivo se muestra a continuación:

1	2	3	4	5	6	7	8	
								Registro 1
								Registro 2
		. . .						
								Registro 3000

ARCHIVO DE CLIENTES

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

1.	Archivo físico	( ) Conjunto de registros con significado para el usuario almacenados en memoria secundaria.
2.	Archivo	( ) Interfaz entre los archivos lógicos y físicos.
3.	Acceso directo	( ) Estructura de datos formada por registros físicos de longitud fija o variable.
4.	Acceso secuencial	( ) Obtención de los registros de un archivo utilizando el orden indicado por el número de registro.
5.	Cinta magnética	( ) Medio de almacenamiento donde el acceso lógico y físico de los archivos es en forma secuencial.
6.	Disco magnético	( ) Permite la organización de archivos en forma contigua y en tabla de mapeo.
7.	Sistema de archivos	( ) Obtención de los registros de un archivo en cualquier orden.
		( ) Medio de almacenamiento donde el acceso lógico y físico a los archivos es forma secuencial o directa.
		( ) Estructura de datos sobre la que se realizan las operaciones de lectura y escritura de registros.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

1. Se desea crear un archivo de facturación en el que cada registro contiene la información concerniente a una factura. Los campos del registro y el número de caracteres son los siguientes:

CAMPO	DESCRIPCION	TAMAÑO (Caracteres)
1	Fecha	6
2	Número factura	5
3	Número vendedor	3
4	Número artículo	4
5	Descripción	19
6	Cantidad	4
7	Monto	7

Si el archivo tendrá 6000 facturas y la longitud del registro físico es de 512 bytes; entonces calcular:

a. El número de registros físicos.
b. El número de caracteres por registro.
c. El número de caracteres en el archivo.

2. Considere que el archivo del problema anterior va a ser almacenado en forma contigua en un disco magnético que utiliza una superficie de grabación con 20 pistas y 10 registros por pista. El tamaño de estos registros físicos es de 512 bytes y en cada uno de ellos se utilizan 32 bytes para información de control.

Si el archivo se inicia a partir del registro físico número doce, y se solicita leer el registro lógico 23, determine:

a. El factor de bloqueo óptimo.
b. El tamaño de la cadena de bytes (TCB).
c. La dirección lógica de la cadena de bytes (DLCB).
d. El número de registro físico (NRF).
e. El desplazamiento (D).
f. El número de pista (NP).
g. El número de registro dentro de la pista (NRDP).

UNIDAD VI METODOS DE ORDENAMIENTO

OBJETIVO GENERAL

El alumno aplicará los métodos internos y externos más importantes para efectuar ordenamientos en la computadora.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Identificará el proceso de ordenamiento.
2. Diferenciará un método interno de un método externo de ordenamiento.
3. Diferenciará los métodos de ordenamiento basados en selección, intercambio, inserción, distribución e intercalación.
4. Aplicará los métodos a grupos de datos para efectuar su ordenamiento.
5. Escribirá programas de computadora para ordenar vectores y archivos.
6. Analizará los métodos de ordenamiento.

INTRODUCCION

El ordenamiento es una de las operaciones más importantes que se practica sobre una estructura de datos. Ordenar una estructura de datos es establecer un orden de precedencia entre los elementos de la estructura, de acuerdo a uno o más campos (llaves) que se seleccionen para tal fin.

Para lograr esto, se han desarrollado una gran cantidad de algoritmos, los cuales son el objeto de estudio de esta unidad. En ésta, se dan elementos para el análisis y la selección de los métodos, así como los algoritmos más importantes de los ordenamientos internos y externos.

VI.1 GENERALIDADES

Es frecuente encontrar operaciones de ordenamiento como parte de los procesos para el manejo electrónico de datos. Es conveniente mantener ordenado un conjunto de datos para facilitar la actualización o recuperación de ellos. Ordenar un conjunto de datos puede parecer una operación trivial para un programador con poca experiencia, pero en realidad es una operación costosa que deberá realizarse solamente cuando sea estrictamente necesario y en este caso seleccionar el método apropiado.

CONSIDERACIONES PARA LA SELECCION DEL METODO.

Para seleccionar un método de ordenamiento en una cierta situación, es aconsejable considerar lo siguiente:

1. El tipo de memoria en la que se encuentran los datos. Esta puede ser de acceso directo y de alta velocidad, de acceso directo y de mediana velocidad, y de acceso secuencial.

2. Las características del sistema operativo para el manejo de archivos y de la memoria.

Criterios de selección

3. Los tiempos de acceso a los dispositivos.

4. La cantidad, el tipo y la distribución inicial de los datos.

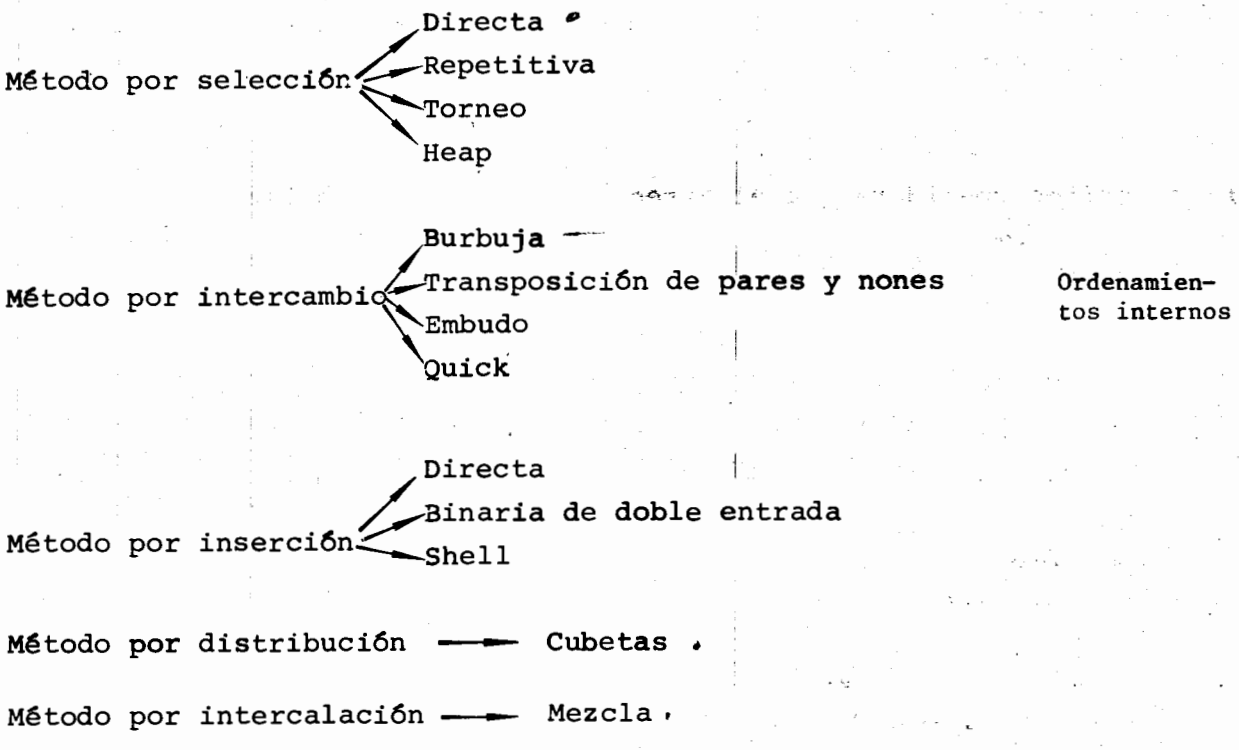
5. La eficiencia del método. Para establecer la eficiencia de un método de ordenamiento, basta con determinar el número promedio de comparaciones y de intercambios, así como la cantidad de memoria que requiere.

Para tener una idea del comportamiento de los algoritmos de ordenamiento, los mejores métodos realizan un número de comparaciones proporcional a $n \log_2 n$. (n = número de elementos a ordenar).

VI.2 ORDENAMIENTOS INTERNOS

Los métodos de ordenamiento que se utilizan para un conjunto de datos almacenados en una memoria de acceso directo de alta velocidad, son llamados métodos de ordenamiento interno. Estos se encuentran clasificados en métodos de selección, intercambio, inserción, distribución e intercalación de acuerdo al principio en el que se basan.

La tabla siguiente muestra los algoritmos más importantes en cada caso.



En muchas de las aplicaciones de los ordenamientos, la información contenida en los registros es grande y, generalmente, el o los campos que la identifican son pequeños. En los algoritmos que estudiaremos, se tomarán de cada registro solamente los campos considerados como llaves.

Liberación de llaves

VI.2.1 METODOS POR SELECCION

Los métodos de selección, como su nombre lo indica, seleccionan del conjunto de datos, según el criterio que se siga, al mayor o al menor de los datos del conjunto y lo excluye para proceder sobre los restantes de forma similar.

SELECCION DIRECTA

Este método de ordenamiento consiste en seleccionar del conjunto de datos el elemento más pequeño en valor y excluirlo del conjunto de datos para repetir el procedimiento sobre los restantes.

El algoritmo que realiza esta operación supone que los datos están almacenados en un arreglo A, de una dimensión y para el ahorro de memoria, intercambia en la primera pasada al dato menor con el que está en la posición uno del arreglo, en la siguiente, el dato menor con el que está en la posición dos, etc.

ALGORITMO

El algoritmo considera que el arreglo A, contiene las llaves:

```
DESDE  I=1     HASTA  N-1
   K=1
   DESDE  J=I+1   HASTA N
       SI  A(J)<A(K)
           K=J
       FIN
   FIN
   SI    I<>K
         C=A(L)
         A(I)=A(K)
         A(K)=C
   FIN
FIN
```

Ejemplo VI.1

Consideremos el grupo de datos siguientes:
(10, 12, 8, 1, 6, 9).

Pasadas	1		2		3	4	5
	i10	1	1	1	1	1	1
	12	12	i12	6	6	6	6
	8	8	8	8	ik8	8	8
	k1	1Ø	1Ø	1Ø	1Ø	ik1Ø	1Ø
	6	6	k6	12	12	12	ik12
	19	19	19	19	19	19	19

i y k posiciones que se intercambian

ANALISIS

El número de comparaciones que este algoritmo realiza puede ser calculado de la forma siguiente: en la primera pasada realiza n - 1 comparaciones para obtener un dato, el cual se elimina del conjunto, en la segunda, realiza n - 2 comparaciones y, siguiendo este criterio, el total, de comparaciones es $(n - 1) + (n - 2) + ... + 2 + 1 = (n\uparrow 2 - n)/2$. El número total de comparaciones que realiza es muy grande con respecto a $n \log_2 n$.

Comparaciones del algoritmo de selección directa $\approx n^2$

El número de intercambio es de n - 1, ya que en cada pasada se realiza uno, y la memoria adicional que requiere es muy poca.

SELECCION REPETITIVA

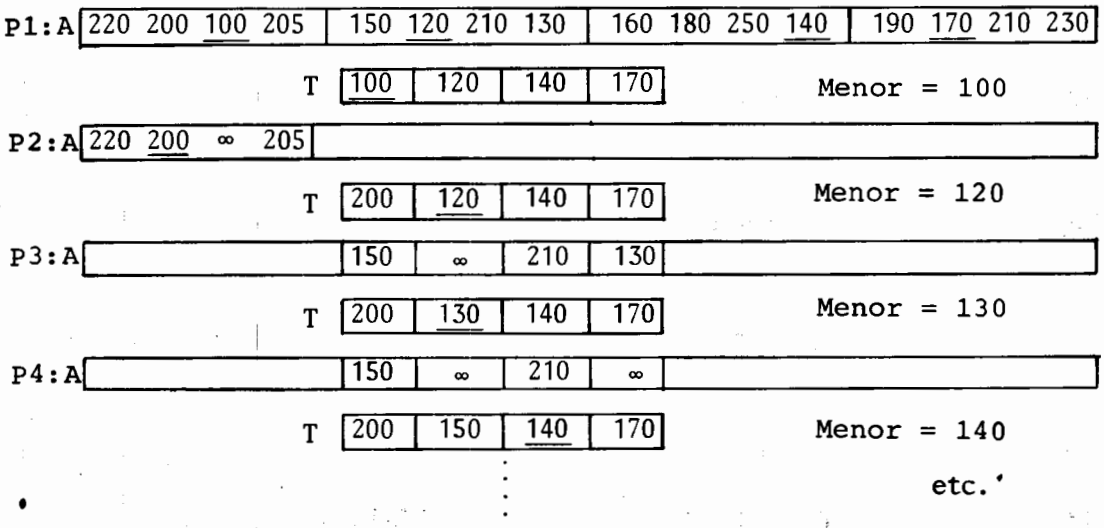
Este método, es una variante del anterior, y consiste en dividir al grupo de datos en raíz de n grupos de raíz de n datos. De cada grupo se selecciona el dato menor y es llevado a un arreglo temporal T de longitud raíz de n, del cual se selecciona al menor, que resulta ser el ganador en esta pasada. Este dato es eliminado de T y del grupo de donde provino. Posteriormente se selecciona nuevamente al menor de este grupo y es llevado a T para continuar con el procedimiento.

```
ALGORITMO

   I=√N
   P=Ø
   K=1-I
   DESDE  J=1  HASTA I
          P=P+1
          K=K+I
          M=K
          DESDE  L=K  HASTA  K+I-1
                 SI A(L)<A(M)
                      M=L
                 FIN
          FIN
          T(P,1)=A(M)
          T(P,2)=M
   FIN
   DESDE  J=1  HASTA N
          M=1
          DESDE  L=1  HASTA I
                 SI T(L,1)<T(M,1)
                 M=L
                 FIN
          FIN
          P=M
          MENOR=T(M,1)
          A(T(M,2))=INFINITO
          K=I*(M-1)+1
          M=K
          DESDE  L=K  HASTA  K+I-1
                 SI A(L)<A(M)
                      M=L
                 FIN
          FIN
          T(P,1)=A(M)
          T(P,2)=M
   FIN
```

Ejemplo VI.2

Para el conjunto de llaves (220, 200, 100, 205, 150, 120, 210, 130, 160, 180, 250, 140, 190, 170, 210, 230) se ordenan:



ANALISIS

Si consideramos que raíz de n es un número entero, el número de comparaciones en cada grupo es raíz de n, menos uno.

El número de grupos considerando a T es de raíz de n más uno. Para la primera pasada, el número de comparaciones es igual a la multiplicación del número de grupos por el número de comparaciones que da como resultado exactamente n - 1.

Pero, para las siguientes pasadas, sólo se examina el grupo de donde provino la menor y T, dando un número de comparaciones igual a dos veces raíz de n menos 1. De esta manera el número total de comparaciones es:

Comparaciones del algoritmo de selección repetitiva de orden $n\sqrt{n}$

$$(n - 1) + (n - 1)\left[2(\sqrt{n} - 1)\right]$$

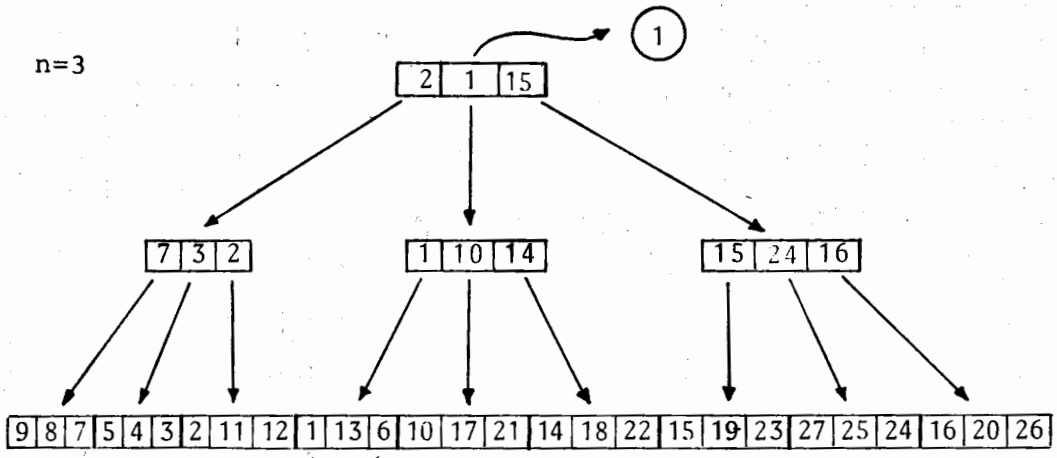
En el proceso no hay intercambio de datos, pero sí la escritura de un dato mayor en cada selección. El espacio de memoria que se requiere es muy pequeño, solamente raíz de n, pero, si se desea tener un vector ordenado, se requiere adicionalmente, de n localidades de memoria.

Esta forma de particionar los datos, puede extenderse a una selección cúbica, en la cual el conjunto de datos se divide en $\sqrt[3]{n}$ grupos con $\sqrt[3]{n}$ pequeños grupos de $\sqrt[3]{n}$ datos.

Ejemplo VI.3

Consideremos el grupo de llaves (9, 8, 7, 5, 4, 3, 2, 11, 12, 1, 13, 6, 10, 17, 21, 14, 18, 22, 15, 19, 23, 27, 25, 24, 16, 20, 26).

La gráfica siguiente muestra la primera pasada del método.



En el método de selección cúbica el número de comparaciones es menor que en el de selección cuadrática, pero el número de localidades de memoria es mayor.

Este procedimiento de división de datos puede extenderse hasta alcanzar la estructura de un árbol binario, llamándose en este caso, ordenamiento de torneo debido a la forma como se desarrollan las partidas en un torneo relámpago de beisbol, tenis u otro.

Comparaciones del algoritmo de selección cúbica $n\sqrt[3]{n}$

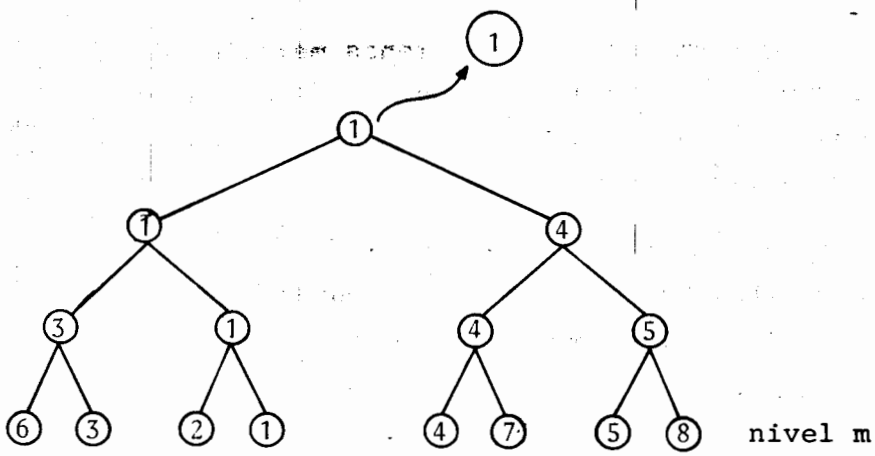
TORNEO

El conjunto de datos es estructurado como un árbol binario en el que las hojas del árbol son los datos originales. En el nivel m - 1 se tiene a los datos que resultan ser los menores de cada par. Esta idea se extiende a to-

dos los niveles hasta tener en la raíz al menor de todo el grupo. Para continuar con el procedimiento, se sustituyen los nodos que contienen al menor por un número muy grande y se realizan nuevamente las comparaciones entre los pares de nodos afectados por la sustitución.

Ejemplo VI.4

Para el conjunto de llaves (6, 3, 2, 1, 4, 7, 5, 8), la primera pasada del algoritmo de torneo es la que se muestra en la siguiente figura.



ANALISIS

El número de comparaciones para el método puede calcularse de la siguiente manera: en el nivel m, el número de comparaciones es n/2, en el m - 1 es n/4, hasta el nivel 2, donde se requiere una comparación. De esta forma el número de comparaciones para la pasada uno es de $n/2 + n/4 + n/8 + \ldots\ldots + 4 + 2 + 1$, esto es: $n - 1$.

Para las siguientes $n - 1$ pasadas, se requieren tantas comparaciones como niveles tenga el árbol menos una, ya que en la raíz no hay comparación.

Comparaciones del algoritmo de torneo $n \log_2 n$

El número de niveles en un árbol binario completo es de $\log_2 n - 1$. De esta manera el número de comparaciones en esta etapa es de $(n - 1)(\log_2 n)$.

El total de comparaciones es entonces $(n - 1) + (n - 1)(\log_2 n)$. Que, en forma aproximada, nos da un comportamiento proporcional a $n \log_2 n$.

Intercambios no se practican en el algoritmo, pero hay que recorrer la ruta del menor para cambiarlo.

El espacio de memoria utilizado es muy grande, ya que se requiere de una cantidad casi igual al número de datos y, si se desea un vector ordenado, el total es de casi 2n.

HEAPSORT

El heapsort es un algoritmo que utiliza menos memoria adicional de la que requieren los algoritmos de selección repetitiva. Se basa en la utilización de una estructura llamada heap y de la práctica de un recorrido llamado borrado del heap.

Heap y borrado de heap

Un heap es un árbol binario completo al que pueden faltarle algunas de las hojas situadas más a la derecha en el último nivel.

En cualquier nodo, el dato es mayor que los que se encuentran en sus subárboles. (Véase la figura VI.1).

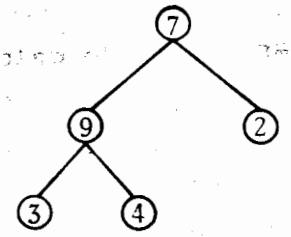


Figura VI.1 Arbol binario heap.

El algoritmo para borrar el heap consiste en remover el dato en la raíz e intercambiarlo con otro nodo, de acuerdo al método siguiente se procede con los nodos nivel por nivel, de abajo hacia arriba y de derecha a izquierda.

El dato con el que se intercambia la raíz debe ser acomodado dentro del árbol, de tal forma que se mantenga la estructura de un heap considerando solamente los nodos activos.

ALGORITMO

Fase I construcción del heap. (El algoritmo considera la existencia del árbol).

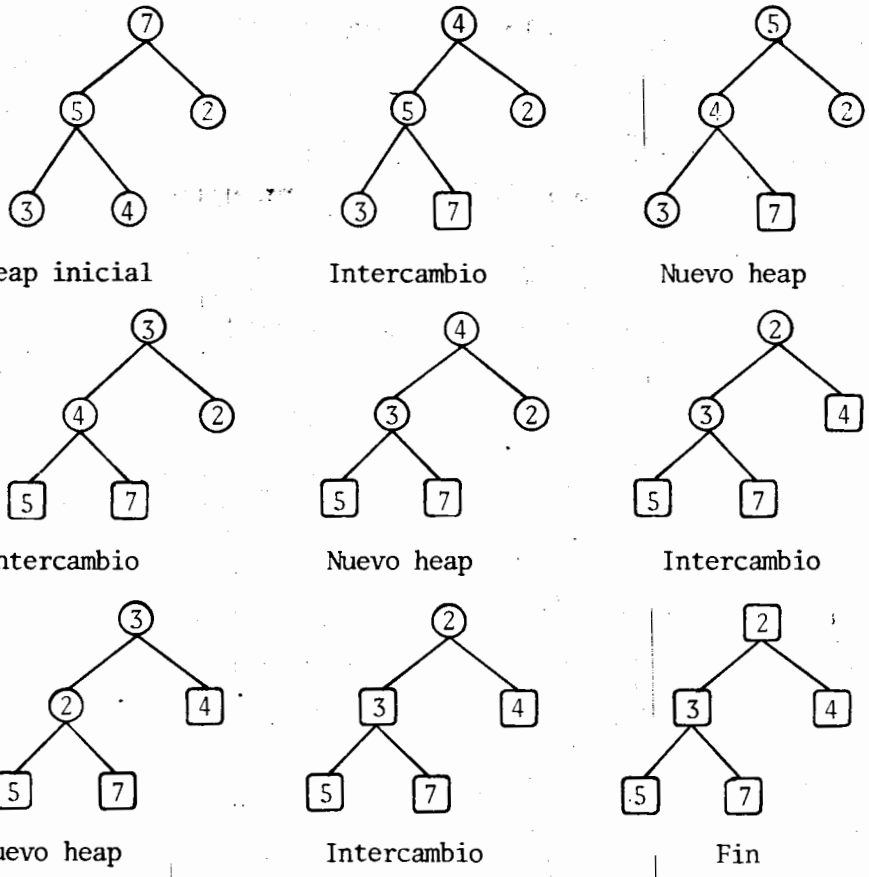
```
D=PROFUNDIDAD DEL ARBOL
DESDE L=D-1  HASTA 1 PASO -1
        ENTANTO CADA NODO P QUE NO SEA HOJA EN EL NIVEL L
             K=LLAVE(P)
             ENTANTO P NO SEA HOJA
                  M=APUNTADOR AL HIJO DE P CON MAYOR LLAVE
                  SI K<LLAVE(M)
                       INTERCAMBIAR
                       P=M
                  OBIEN
                       P=HOJA
                  FIN
             FIN
        FIN
FIN
```

Fase II borrado del heap.

```
R=RAIZ
ENTANTO EL NUMERO DE NODOS DEL HEAP>UNO
        K=HOJA MAS A LA DERECHA DEL ULTIMO NIVEL
        P=R
        INTERCAMBIAR NODO (K) CON NODO (P) Y ELIMINAR (K) DEL ARBOL
        ENTANTO P NO SEA HOJA
             M=APUNTADOR AL HIJO MAYOR DE P
             SI LLAVE (P) < LLAVE (M)
                          INTERCAMBIAR
                          P=M
             OBIEN
                          P=HOJA
             FIN
        FIN
FIN
```

Ejemplo VI.5

Consideremos el grupo de llaves (7, 5, 2, 3, 4) arreglado en forma de heap. La fase de borrado se describe en los árboles que se presentan a continuación.



ANALISIS

El algoritmo de borrado del heap realiza en el peor de los casos $2(n - 1)d$ comparaciones donde $d = \lfloor \log n \rfloor - 1$. El número de comparaciones para la construcción del heap en el peor de los casos es de $(n - 1) + 2d$.

Comparaciones del algoritmo de heap $n \log_2 n$

En forma general, podemos decir que el algoritmo tiene un promedio de comparaciones proporcional al $n \log_2 n$.

VI.2.2 METODOS POR INTERCAMBIO

Los métodos por intercambio transponen o intercambian sistemáticamente pares de datos que se encuentran fuera de

orden hasta que dejen de existir. Los métodos representativos de este grupo, mismos que serán tratados en esta unidad son la burbuja, transposición de pares y nones, embudo y quick.

BURBUJA

El nombre burbuja se debe a la manera como los datos se mueven dentro del conjunto, aparentando ser burbujas en el agua subiendo a la superficie.

El algoritmo se inicia comparando las llaves n y n - 1, y las intercambia si n<n-1, compara después n - 1 y n - 2 y las intercambia, si n - 1 < n - 2 . Este procedimiento se practica hasta comparar los datos 1 y 2 y, cuando esto sucede, el menor dato ha alcanzado su posición final.

El procedimiento se practica entonces sobre los datos comprendidos entre 2 y n. Para el proceso se requiere como máximo n - 1 pasadas, pero el algoritmo puede ordenar los datos antes de dar la pasada n - 1. Para identificar esto, hay que observar cuando ya no existe intercambio en una pasada.

```
ALGORITMO
   K=Ø
   I=VERDADERO
   MIENTRAS I=VERDADERO
        K=K+1
        I=FALSO
        DESDE  J=N HASTA K PASO -1
              SI A(J)<A(J-1)
                   I=VERDADERO
                   C=A(J)
                   A(J)=A(J-1)
                   A(J-1)=C
              FIN
        FIN
   FIN
```

Ejemplo VI.6

Consideremos el grupo de llaves (10, 12, 8, 1, 6, 19, 4)

	pasadas				
datos	1	2	3	4	5
10	1	1	1	1	1
12	10	4	4	4	4
8	12	10	6	6	6
1	8	12	10	8	8
6	4	8	12	10	10
19	6	6	8	12	12
4	19	19	19	19	19

detalle de la pasada 1

10	10	10	10	10	10	1
12	12	12	12	12	1	10
8	8	8	8	1	12	12
1	1	1	1	8	8	8
6	6	4	4	4	4	4
19	4	6	6	6	6	6
4	19	19	19	19	19	19

ANALISIS

El análisis del algoritmo no es simple, ya que el número de pasadas depende de los datos. Si asumimos que siempre se darán n - 1 pasadas, el número de comparaciones es el mismo que el del algoritmo de selección directa, pero para un conjunto desordenado de datos, es probable que el número de comparaciones sea menor que n - 1. De todas formas, cuando el algoritmo realiza un número de pasadas menor que n - 1, las pasadas que nos hemos ahorrado son las que contienen el menor número de comparaciones.

Comparaciones del algoritmo de burbuja $\frac{n^2}{4}$

La ventaja principal es que el algoritmo puede resultar muy apropiado para datos que se encuentran cercanos a su posición final. El número de intercambios también depen-

de de los datos, pero en promedio es mayor que n - 1 y el espacio de memoria adicional utilizado es muy reducido. El comportamiento promedio del algoritmo es de (n↑2)/4.

TRANSPOSICION DE PARES Y DE NONES

El algoritmo consiste de dos etapas. En la primera, se comparan los datos nones con su sucesor y se les intercambia si es necesario. En la segunda, se comparan los datos pares con su sucesor y se les intercambia si es necerio. Este proceso se practica hasta que en ambas pasadas no se encuentre un intercambio.

Ejemplo VI.7

Consideremos el grupo de llaves (10, 12, 8, 1, 6, 19, 4).

d	n1 (non)	p1 (par)	n2	p2	n3	p3	n4	p4
1Ø	1Ø	1Ø	1	1	1	1	1	1
12	12	1	1Ø	6	6	4	4	4
8	1	12	6	1Ø	4	6	6	6
1	8	6	12	4	1Ø	8	8	8
6	6	8	4	12	8	1Ø	1Ø	1Ø
19	19	4	8	8	12	12	12	12
4	4	19	19	19	19	19	19	19

EMBUDO

Este algoritmo es muy similar al de la burbuja, excepto que las pasadas se alternan, una para colocar la llave menor, otra para la llave mayor, y así sucesivamente.

Ejemplo VI.8

Consideremos el grupo de llaves (6, 5, 9, 7, 10, 1, 3, 2, 4, 8).

d	p1 ↑	p2 ↓	p3 ↑	p4 ↓	p5 ↑	p6 ↓	p7 ↑	p8 ↓	p9 ↑
6	1								
5	6	5	2						
9	5	6	5	5	3				
7	9	7	6	6	5	5	4		
1Ø	7	9	7	7	6	6	5	5	5
1	1Ø	2	9	3	7	4	6	6	6
3	2	3	3	4	4	7	7	7	
2	3	4	4	8	8	8			
4	4	8	8	9					
8	8	1Ø							

QUICK

El algoritmo quicksort es un procedimiento recursivo en el que se intercambian los datos para colocarlos en orden con respecto a uno de ellos, llamado D, de tal manera que, a la derecha de D, queden los elemento Li > D y, a la izquierda los elementos Li < D.
Este proceso se repite sobre la lista de datos a la derecha de D y sobre la lista de datos a la izquierda de D. El proceso continúa hasta que en todas las sublistas se tenga únicamente un solo dato.
Para seleccionar D en nuestro algoritmo, escogemos el primer elemento de la lista.

ALGORITMO

En forma muy general el algoritmo contempla una fase de división de la lista con respecto a D y procede sobre las dos sublistas generadas. Este proceso se practica recursivamente hasta que se tenga únicamente un solo elemento en todas las sublistas.

```
QUICKSORT(L,I,S)
     SI  I<S
          DIVIDE (L,I,S,P)
          QUICKSORT (L,I,P-1)
          QUICKSORT (L,P+1,S)
     FIN
FIN
```

L = lista de datos
I = límite inferior de la lista
S = límite superior de la lista

Ejemplo VI.9

Consideremos el grupo de llaves (10, 12, 8, 1, 6, 19, 4).

1Ø	12	8	1	6	19	4	Pasada 1
-	12	8	1	6	19	4	
4	12	8	1	6	19	-	
4	-	8	1	6	19	12	
4	6	8	1	-	19	12	
4	6	8	1	-	19	12	
4	6	8	1	-	19	12	
(4	6	8	1)	1Ø	(19	12)	
-	6	8	1				Pasada 2
1	6	8	-				
1	-	8	6				
(1)	4	(8	6)				

ANALISIS

El algoritmo es muy eficiente para llaves que se encuentran aleatoriamente distribuidas. El método ha recibido muchos refinamientos, tal como no permitir que las sublistas alcancen un solo elemento, sino que para $s-I \leq c$, se opte por llamar a otro algoritmo que continúe con el proceso, c es una constante previamente definida respecto al número de llaves en cada sublista. Respecto a la selección de D, también se han sugerido varias estrategias, como sería tomar un valor cercano a la mediana.

VI.2.3 METODOS POR INSERCION

Los algoritmos de inserción suponen que el conjunto de llaves se encuentra ordenado. Para una llave K que se desea agregar al conjunto, se determina el lugar que debe ocupar y se mueven los datos una posición para insertarlo en su posición correcta.

INSERCION DIRECTA

El algoritmo considera que el conjunto ordenado sólo tiene un elemento y a partir de éste se irán insertando los demás.

ALGORITMO

```
    DESDE  J=2  HASTA N
           I=J-1
           K=A(J)

    X:          SI K<A(I)
                    A(I+1)=A(I)
                    I=I-1
                    SI  I>Ø
                          VE A X
                    FIN
                OBIEN
                    A(I+1)=K
                FIN
    FIN
```

Ejemplo VI.10

Consideremos el grupo de llaves (6, 3, 2, 5, 4, 1, 8, 9, 7).

	6	3	2	5	4	1	8	9	7
p1	3	6							
p2	2	3	6						
p3	2	3	5	6					
p4	2	3	4	5	6				
p5	1	2	3	4	5	6			
p6	1	2	3	4	5	6	8		
p7	1	2	3	4	5	6	8	9	
p8	1	2	3	4	5	6	7	8	9

INSERCION BINARIA

A diferencia del algoritmo de inserción directa, el procedimiento de inserción binaria consiste en comparar el dato con el elemento central del conjunto de datos y determinar si las comparaciones continúan sobre los datos a la derecha del elemento central o a su izquierda. Este proceso se repite hasta determinar entre qué elementos debe estar el dato.

INSERCION DE DOBLE ENTRADA

Con el objeto de reducir el número de intercambios, se propone colocar al primer elemento en el centro del área de salida e ir colocando los demás a la derecha o a la izquierda; dependiendo de por dónde se muevan menos elementos.

Ejemplo VI.11

5Ø3	Ø87	512	Ø61	17Ø	897	275
			^5Ø3			
		Ø87	5Ø3^			
		^Ø87	5Ø3	512		
	Ø61	Ø87	^5Ø3	512		
	Ø61	Ø87	17Ø	5Ø3	512^	
	Ø61	Ø87	17Ø	^5Ø3	512	897
Ø61	Ø87	17Ø	275	5Ø3	512	897

SHELL

El algoritmo Shell clasifica un conjunto de llaves ordenando sucesivamente subconjuntos de llaves que están entremezclados en todo el conjunto.

Estos subconjuntos están definidos por una secuencia h(1), h(2), h(3),..., h(t - 1), h(t), llamados incrementos, que determinan qué elementos pertenecen a un subconjunto. El incremento final h(t), es siempre 1.

ALGORITMO

El algoritmo considera que el arreglo B, contiene a los incrementos, y A, las llaves.

```
DESDE  M=1  HASTA T
       H=B(M)
       DESDE  J=H+1  HASTA N
              I=J-H
              K=A(J)
              REPETIR
                   SI K<A(I)
                        A(I+H)=A(I)
                        I=I-H
                   OBIEN
                        VE A X
                   FIN
              HASTA I<=Ø
       X:     A(I+H)=K
       FIN
```

Ejemplo VI.12

Consideremos el conjunto de datos siguientes:

datos	5Ø3	Ø87	512	Ø61	9Ø8	17Ø	Ø79
h=3	Ø61			5Ø3			
	Ø61	Ø87		5Ø3	9Ø8		
	Ø61	Ø87	17Ø	5Ø3	9Ø8	512	
	Ø61	Ø87	17Ø	Ø79	9Ø8	512	5Ø3
h=2	Ø61		17Ø				
	Ø61	Ø79	17Ø	Ø87			
	Ø61	Ø79	17Ø	Ø87	9Ø8		
	Ø61	Ø79	17Ø	Ø87		512	
	Ø61	Ø79	17Ø	Ø87	5Ø3	512	9Ø8
h=1	Ø61	Ø79					
	Ø61	Ø79	17Ø				
	Ø61	Ø79	Ø87	17Ø			
	Ø61	Ø79	Ø87	17Ø	5Ø3		
	Ø61	Ø79	Ø87	17Ø	5Ø3	512	
	Ø61	Ø79	Ø87	17Ø	5Ø3	512	9Ø8

ANALISIS

El número de comparaciones que hace el algoritmo está en función de la secuencia de incrementos usada.

Dos buenas secuencias de valores para h son:

- Cuando t=2, entonces $h_1 = 1.72\sqrt[3]{n}$ y $h_2 = 1$, con estos valores el número de comparaciones es proporcional a $n^{5/3}$.

Comparaciones del algoritmo Shell $n^{3/2}$

- Si se usa más de dos incrementos la secuencia de valores está definida por $h(k) = 2^k - 1$ para $1 \leq k \leq \log n$, con estos valores el número de comparaciones es proporcional a $n^{3/2}$.

VI.2.4 METODOS POR DISTRIBUCION

Los métodos de distribución son análogos al proceso de ordenamiento que emplea un clasificador de tarjetas, método que consiste en una colección de casilleros, donde se depositan las tarjetas que coincidan con la marca del casillero.

CUBETAS

El algoritmo de radix sort o bucket sort es de los más representativos entre los métodos de distribución y se basa en la idea general de los métodos de distribución. En estos algoritmos podemos diferenciar dos fases: una de distribución de los datos sobre los buckets o cubetas, y la otra, de recolección de los datos.

ALGORITMO

Para este algoritmo, las estructuras de datos que se utilizan son: una lista ligada, donde se encuentran los datos por ordenar, un arreglo en donde sus elementos son utilizados como cabeza de lista para cada cubeta y un arreglo para indicar el último elemento en cada cubeta.

Fase I. Distribución en las cubetas:

```
DESDE  J=1  HASTA NUMERO DE CUBETAS
       ULTIMO (J)=LOCALIDAD DE LA CABEZA DE LISTA J
FIN
P=LIGA(T) ; APUNTADOR A LA LISTA DE DATOS
ENTANTO P<>VACIO
     EXTRAER DIGITO Y ASIGNAR CUBETA I
     LIGA(ULTIMO(I))=P
     ULTIMO(I)=P
     P=LIGA(P)
FIN
```

Fase II. Recoger de las cubetas:

```
P=LOC DE T
DESDE  J=1  HASTA NUMERO DE CUBETAS
       SI ULTIMO(J)<>LOCALIDAD DE CABEZA DE J
            LIGA(P)=LIGA(CUBETA(J))
            P=ULTIMO(J)
       FIN
FIN
LIGA(P)=VACIO
```

Ejemplo VI.13

Consideremos el grupo de llaves siguiente:

135 2Ø9 1Ø7 1Ø6 124 221 1Ø3 2Ø8 2Ø4

distribución		
cub1	221	
cub3	1Ø3	
cub4	124	2Ø4
cub5	135	
cub6	1Ø6	
cub7	1Ø7	
cub8	2Ø8	
cub 9	2Ø9	

recolección

221 1Ø3 124 2Ø4 135 1Ø6 1Ø7 2Ø8 2Ø9

distribución						
cubØ	1Ø3	2Ø4	1Ø6	1Ø7	2Ø8	2Ø9
cub2	221	124				
cub3	135					

recolección

1Ø3 2Ø4 1Ø6 1Ø7 2Ø8 2Ø9 221 124 135

distribución					
cub1	1Ø3	1Ø6	1Ø7	124	135
cub2	2Ø4	2Ø8	2Ø9	221	

recolección

1Ø3 1Ø6 1Ø7 124 135 2Ø4 2Ø8 2Ø9 221

VI.2.5 METODOS POR INTERCALACION

Los métodos de ordenamiento por intercalación entremezclan dos o más conjuntos de datos para formar un nuevo conjunto ordenado. El caso típico en los procesos de mezcla consiste en entremezclar dos archivos, A y B, ordenados para formar un nuevo archivo C ordenado. Para realizar esta operación, los dos primeros elementos de A y B, son comparados, y el menor se mueve a C. Después se compara el dato siguiente del conjunto ganador, con el otro, y el menor se mueve a C; este proceso continúa hasta que uno de los conjuntos se agote y simplemente el otro sea copiado a C.

```
ALGORITMO
   I=1
   J=1
   K=1
   ENTANTO I<=N  y  J<=M
        SI A(I)<B(I)
             C(K)=A(I)
             I=I+1
        OBIEN
             C(K)=B(J)
             J=J+1
        FIN
   K=K+1
   FIN
   SI   I>N
        DESDE  L=J  HASTA M
               K=K+1
               C(K)=B(L)
        FIN
   OBIEN
        DESDE  L=I  HASTA N
               K=K+1
               C(K)=A(L)
        FIN
   FIN
```

Ejemplo VI.14

Consideremos los dos conjuntos de datos siguientes:

conjunto A (3 5 7 8 9)

conjunto B (1 2 4 6)

3>1	C	(1
3>2	C	(1 2
3<4	C	(1 2 3
5>4	C	(1 2 3 4
5<6	C	(1 2 3 4 5
7>6	C	(1 2 3 4 5 6

se han agotado los elementos de B

C (1 2 3 4 5 6 7 8 9)

VI.3 ORDENAMIENTOS EXTERNOS

Los algoritmos que estudiamos bajo la clasificación de ordenamientos internos suponen que el conjunto total de llaves por clasificar reside en la memoria primaria. Ahora bien, existe un problema en el que el conjunto de llaves por clasificar es mayor que la capacidad de almacenamiento de la memoria primaria. Este problema pertenece al campo de los ordenadores externos los cuales, dan diferentes alternativas para resolverlo.

Supongamos que tenemos un conjunto de 5000 registros $R_1, R_2, \ldots R_{5000}$ para ordenar, si las llaves de estos registros sólo pueden ser llevadas a la memoria en grupos de 1000, es posible dividir el conjunto en cinco subconjuntos de R_1 a R_{1000}, R_{1001} a R_{2000} .... R_{4001} a R_{5000}, ordenarlos en forma independiente y posteriormente entremezclarlos por algún proceso de intercalación ya estudiado.

El esquema general de los ordenamientos externos está compuesto de dos fases:

1. Construir y distribuir arreglos de llaves ordenadas en dos o más archivos auxiliares.
2. Intercalar repetidamente estos arreglos hasta que solamente exista uno, totalmente ordenado.

Para entender los algoritmos, vamos a definir a m como el número máximo de llaves que pueden ser llevadas a la memoria primaria en cualquier momento, y a n como el número total de llaves por ordenar.

VI.3.1 METODO POR POLIFASE

Este método se apoyó en los algoritmos de intercalación y debido a que recurre a estos algoritmos repetidamente, ha sido llamado algoritmo de ordenamiento de intercalación en polifase.

Este algoritmo de ordenamiento externo maneja cuatro cintas T_0, T_1, T_2, T_3 como archivos auxiliares para guardar la distribución de los arreglos, y supone que en T_0 se encuentran las llaves originales.

El algoritmo que se describe a continuación es el que define a los procesos de ordenamiento de intercalación en polifase.

ALGORITMO

Ordenamiento externo de cuatro cintas.

Fase I. Construcción y distribución de los arreglos ordenados. Mientras existan datos en T_0, los pasos a seguir son:

a. Lectura de m llaves.
b. Ordenar las m llaves por método interno.
c. Si las m llaves anteriores se colocaron en T_2 colocar éstas en T_3, si no, colocarlas en T_2.

Fin

Rebobinar las cintas.

Fase II. Intercalación de los arreglos.

Se asignan los índices de las cintas:

$i \leftarrow 2$ $\quad$ $j \leftarrow 3$

$k \leftarrow 0$ $\quad$ $\ell \leftarrow 1$

Mientras exista más de un arreglo, los pasos a seguir son:

a. Intercalar el primer arreglo en T_i con el primer arreglo en T_j y dejar el resultado en T_k.
b. Intercalar los siguientes arreglos en T_i y T_j y dejar el resultado en T_ℓ.
c. Repetir los pasos (a) y (b) colocando los resultados alternativamente en T_k y T_ℓ hasta que los datos en T_i y T_j se agoten.

d. Rebobinar las cintas. Sumar dos (módulo 4) a i, j, k y ℓ (esto es para invertir el orden de entrada y salida de las unidades de cinta).

Fin

El algoritmo no contempla la posibilidad de que existan más arreglos en una cinta que en la otra, esto se debe a la cantidad de datos y de m. Si esto sucede, el arreglo que no tenga pareja será copiado a la cinta de salida correspondiente.

Ejemplo VI.15

Consideremos m = 4.

Entrada en T_0.

10, 42, 50, 80, 20, 15, 19, 70, 78, 69, 55, 8, 14, 30

Después de la Fase I tenemos:

T_2 |10, 42, 50, 80||8, 55, 69, 78|

T_3 |15, 19, 20, 70||14, 30|

Después de la primera pasada de la Fase II, tenemos:

T_0 |10, 15, 19, 20, 42, 50, 70, 80|

T_1 |8, 14. 30, 55, 69, 78|

Después de la segunda pasada de la Fase II, tenemos:

T_2 |8, 10, 14, 15, 19, 20, 30, 42, 50, 55, 69, 70, 78, 80|

ANALISIS

Para analizar este algoritmo debemos considerar tres aspectos:

a. Comparación de llaves
b. Rebobinado de cintas
c. El número de pasadas sobre las llaves

Parte del problema del análisis de la primera fase ya ha sido estudiado en los algoritmos de los métodos internos.

Para simplificar el análisis de la segunda fase, consideremos que cada arreglo tiene el mismo número de llaves, y que el número total es par; así, el número total de arreglos, cuando el proceso se inicia es r=n/k, donde k es el número de llaves de cada arreglo y n el número total de llaves. Al término de la primera pasada tendremos en las cintas T_2 y T_3,(r/2) arreglos; al término de la siguiente pasada tendremos en las cintas T_0 y T_1,(r/4) arreglos, y al término de la pasada i tendremos $(r/2^i)$ arreglos. El algoritmo termina cuando sólo queda en alguna cinta un arreglo, de esta manera el número de pasadas para intercalar los archivos es $\log_2 r$. Como parte de la construcción y distribución de los arreglos en la primera fase, se requirió de una pasada sobre el total de datos, de esta manera, el total de las pasadas es: $\log_2 r + 1$.

El número de comparaciones depende parcialmente del algoritmo de ordenamiento que se empleó en la fase de construcción de los arreglos. Supongamos que este algoritmo realiza un número proporcional de comparaciones a $m \log_2 m$, de esta manera el número de comparaciones será la multiplicación del número de arreglos distribuidos (n/m) por $m \log_2 m$ o sea, $n \log_2 m$.

Si para la primera pasada de la fase dos, son mezclados r/2 pares de arreglos de tamaño m, se requerirán cuando mucho (r/2)(2m - 1) comparaciones.

La iteración i requiere $(r/2^i)(2^i m - 1) = n - (r/2^i)$, de esta manera el número de comparaciones es:

$$\sum_{i=1}^{\log_2 r} (n - (r/2^i))$$

El número total de comparaciones, incluyendo fase I y II, es:

$$n \log m + \sum_{i=1}^{\log_2 r} (n - (r/2^i))$$

Hemos asumido un conjunto específico de recursos para el diseño del algoritmo. Estos son: una memoria principal, para almacenar m llaves, y cuatro unidades de cinta. Dado que muchos de los sistemas existentes no cuentan con los recursos aquí considerados, será necesario adaptar el algoritmo.

VI.3.2 METODO POR CASCADA

El método de cascada, al igual que el de polifase, se inicia con una distribución de llaves. En este caso la distribución de las llaves en cada uno de los archivos auxiliares, sigue una secuencia que involucra a los números de Fibonacci. La distribución se hace sobre seis cintas T_0, T_1, T_2, T_3, T_4 y T_5 utilizadas como archivos auxiliares.

ALGORITMO

Ordenamiento externo de seis cintas.

Fase I. Construcción y distribución de los arreglos.
Mientras existan registros en T_5 los pasos a seguir son:

a. Lectura de m llaves.
b. Ordenar las m llaves por un método interno.
c. Distribución en las cintas usando una secuencia de Fibonacci.

Fin

Fase II. Intercalación de los arreglos.
Mientras exista más de un arreglo en alguna cinta, los pasos a seguir son:

a. Intercalar el primer arreglo de las cintas T_0, T_1, T_2, T_3, T_4 en T_5 hasta agotar T_4, intercalar T_0, T_1, T_2, T_3 en T_4 hasta agotar T_2, etc., finalmente copiar T_0 a T_1.
b. Intercalar T_1, T_2, T_3, T_4, T_5 en T_0 hasta agotar T_1, intercalar T_2, T_3, T_4, T_5 en T_1 hasta agotar T_2, etc., finalmente intercalar T_4, T_5 en T_3 hasta agotar T_4, por último copiar T_5 a T_4.

Fin

Ejemplo VI.16

Consideremos este ejemplo ilustrado para seis cintas T_0, T_1, ... T_5 y 190 llaves.

	T_0	T_1	T_2	T_3	T_4	T_5
distribución inicial	1^{55}	1^{50}	1^{41}	1^{29}	1^{15}	-

Durante la primera fase del algoritmo se distribuyen las llaves de la siguiente forma: 55 areglos de un solo elemento en T_0, 50 arreglos de un elemento en T_1, etc.

	T_0	T_1	T_2	T_3	T_4	T_5
primera pasada de la fase II	-	1^5	2^9	3^{12}	4^{14}	5^{15}

Durante la pasada número uno de la fase II se intercalaron los arreglos de tal forma que: 15 arreglos de 5 elementos quedaron en T_5, 14 arreglos de 4 elementos T_4, etc.

Para obtener estas distribuciones se intercalaron los arreglos de T_0, T_1, T_2, T_3, T_4, en T_5 hasta agotar T_4, esto hace que 15 arreglos de longitud 5 queden en T_5. Posteriormente se intercalaron los elementos de T_0, T_1, T_2, T_3, en T_4 hasta agotar T_3, etc.

	T_0	T_1	T_2	T_3	T_4	T_5
segunda pasada de la fase II	15^5	14^4	12^3	9^2	5^1	-

Para obtener la distribución de los arreglos, en esta pasada se intercalaron los elementos de T_1, T_2, T_3, T_4, T_5 en T_0 hasta agotar T_1, se procedió con T_2, T_3, T_4, T_5 hasta agotar T_2 y obtener T_1, etc.

	T_0	T_1	T_2	T_3	T_4	T_5
tercera pasada de la fase II	-	15^1	29^1	41^1	50^1	55^1

y finalmente:

	T_0	T_1	T_2	T_3	T_4	T_5
cuarta pasada de la fase II	190^1	-	-	-	-	-

En este método los arreglos se distribuyen en las cintas en cantidades diferentes y conforme se desarrolla el algoritmo, este número se reduce hasta alcanzar el valor de uno en cada cinta, estado que precede la conclusión del algoritmo.

VI.3.3 METODO OSCILANTE

En los métodos hasta aquí analizados notamos dos fases muy claras en sus algoritmos: una de distribución y otra de intercalación, con la característica común de que nunca se pasa a la segunda fase hasta no haber completado la primera. Ahora bien, en este ordenamiento se propone un algoritmo que combina la distribución y la mezcla de los arreglos, de tal manera que gran parte del proceso de ordenamiento toma lugar antes de que la entrada sea completamente examinada.

La distribución y mezcla de los arreglos se practica sobre cinco unidades de cinta T_0, T_1, ... T_4, usadas como archivos auxiliares.

ALGORITMO

Ordenamiento externo con cinco cintas.

Hacer la fase I y II alternadamente hasta que la entrada se agote:

Fase I. Construcción y distribución de los arreglos.
Hacer para i=1 a 4.

a. Lectura de m llaves.
b. Ordenar las m llaves con un método interno.
c. Distribuir los arreglos (para la primera pasada se acomodan en T_0, T_1, T_2 y T_3 dejando T_4 para el resultado, en la segunda pasada se acomoda en T_1, T_2, T_3, T_4, en la tercera pasada en T_0, T_2, T_3, T_4, etc.)
Fin

Fase II. Intercalación de los arreglos.

a. Intercalar para la primera pasada los arreglos de T_0, T_1, T_2, T_3, en T_4, el arreglo en T_4 tiene orden decreciente, para la siguiente pasada intercalar los arreglos de T_1, T_2, T_3, T_4, en T_0, el que también tiene orden decreciente, etc.; el proceso continúa hasta acomodar en las cintas solamente arreglos en orden decreciente, los que formarán un archivo final ascendente durante un proceso de intercalación. Se liberan T_0, T_1, T_2, y T_4 para continuar el proceso.

Fin

Ejemplo VI.17

Supongamos que existen cinco cintas disponibles para procesos de intercalado. El método ordenará 16 arreglos de la siguiente manera: usaremos a Ar y Dr como nomenclatura para indicar arreglos en orden ascendente y descendente de longitud r respectivamente.

El método comenzará con la distribución de un arreglo inicial en A_1 en cada una de las cuatro cintas y los intercalará en la quinta cinta, para producir un arreglo que llamaremos D_4. Posteriormente distribuirá otros cuatro arre

glos, incluyendo la cinta donde está D_4 para intercalarlos y producir un nuevo D_4 en la cinta T_0, etc.

La distribución de los arreglos quedaría:

	Operación	T_0	T_1	T_2	T_3	T_4
Fase I	Distribución	A_1	A_1	A_1	A_1	-
Fase II	Mezcla	-	-	-	-	D_4
Fase I	Distribución	-	A_1	A_1	A_1	D_4A_1
Fase II	Mezcla	D_4	-	-	-	D_4
Fase I	Distribución	D_4A_1	-	A_1	A_1	D_4A_1
Fase II	Mezcla	D_4	D_4	-	-	D_4
Fase I	Distribución	D_4A_1	D_4A_1	A_1	-	D_4A_1
Fase II	Mezcla	D_4	D_4	D_4	-	D_4
Fase II	Mezcla	-	-	-	A_{16}	-

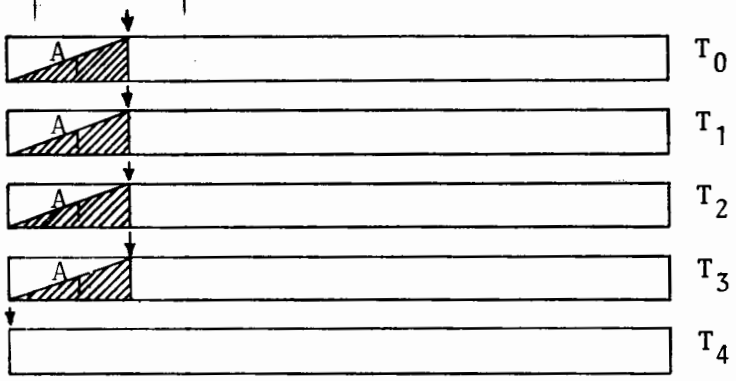
OBSERVACIONES

Es importante hacer notar que el algoritmo optimiza el movimiento de las unidades de cinta. Esto es, cuando en las unidades T_0, T_1, T_2, y T_3 se guardan los arreglos en orden ascendente no se rebobinan para intercalarlos, sino que se toman los datos, empezando por los números mayores para generar un archivo D_4 en orden descendente. Este archivo, llamado D_4, es empujado en la cinta, ya que otros cuatro archivos serán guardados y se les practicará el mismo procedimiento.

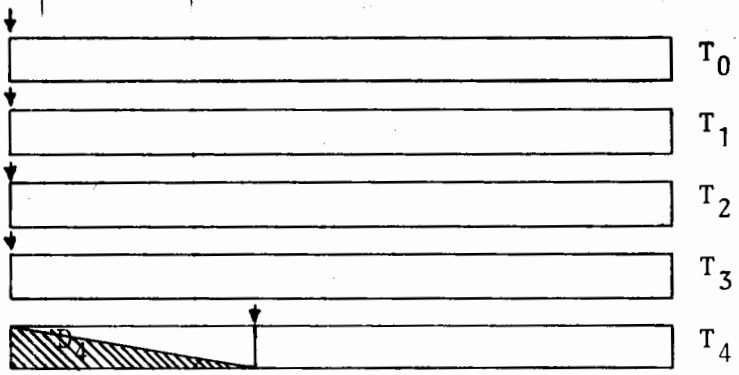
Este proceso continuará hasta lograr cuatro archivos D_4 que se tomarán para producir el archivo A_{16} en orden ascendente.

Gráficamente podemos ilustrar esto:

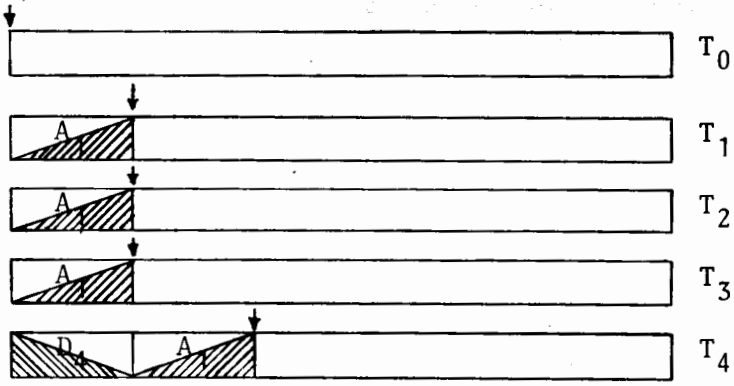
Fase I



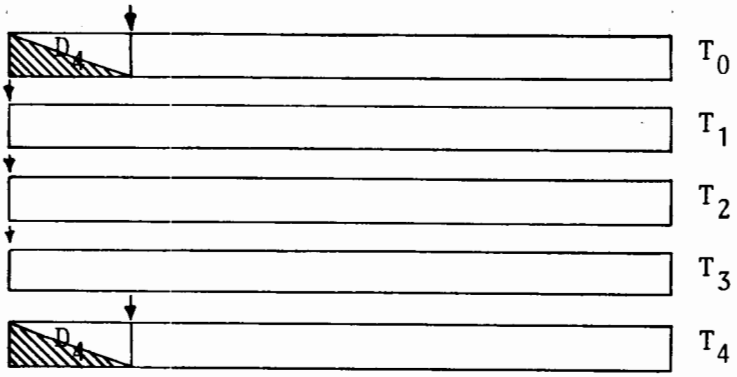
Fase II



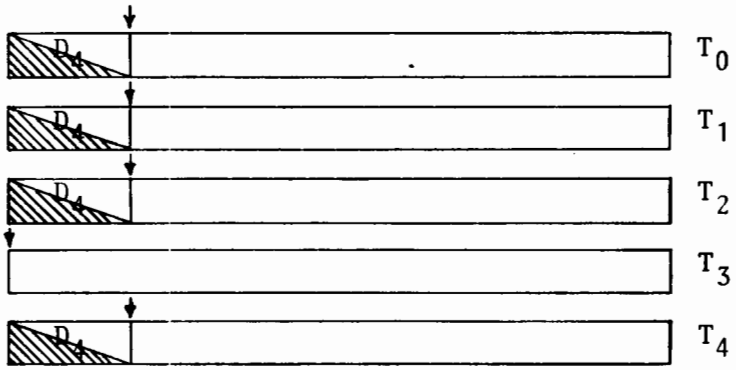
Fase I



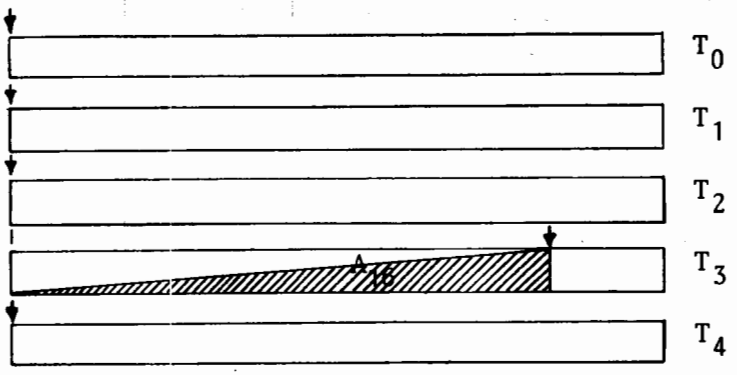
Fase II



Hasta



Finalmente



Es importante hacer notar que el manejo de la cinta implica mayor complejidad en el algoritmo.

VI.3.4 METODO POR DISTRIBUCION

Los algoritmos anteriores basan el proceso de ordenamiento en el principio de intercalación, sin embargo existen otras formas de ordenamiento con cintas.

Una de estas formas se basa en el principio de ordenamiento por distribución, ya estudiada en los métodos internos. Este método también es llamado ordenamiento por raíz, ordenamiento digital, etc., y, de hecho, se opone a los conceptos de intercalación.

Para tener una idea de cómo opera este método utilizando cintas, consideremos el siguiente conjunto de llaves que solamente involucra a los dígitos 0, 1, 2 y 3.

Archivo original desordenado: 1023, 0122, 1131, 3123, 0012, 0132, 2013, 1110

primera pasada.- Tomemos el primer dígito de derecha a izquierda y coloquemos la llave en la cinta correspondiente de acuerdo a:

en T_0 si el dígito es 0

en T_1 si el dígito es 1

en T_2 si el dígito es 2

en T_3 si el dígito es 3

de acuerdo con esto, las cintas quedan:

	T_0	T_1	T_2	T_3
	1110	1131	0122 0012 0132	1023 3123 2013
segunda pasada.-	—	1110 0012 2013	0122 1023 3123	1131 0132
tercera pasada.-	0012 2013 1023	1110 0122 3123 1131 0132	—	—
cuarta pasada.-	0012 0122 0132	1023 1110 1131	2013	3123

Este método puede ser reducido a dos unidades de cinta, si cambiamos la base de los números a base 2, por ejemplo: 11011001, 10110011, etc. Es obvio que el número de pasadas sobre los datos se incrementaría.

VI.4 ARCHIVOS AUXILIARES ALMACENADOS EN DISCO

Hasta aquí, hemos considerado a las cintas como el vehículo principal de almacenamiento, pero existen otros dispositivos que facilitan el manejo de los archivos auxiliares, éstos son llamados discos magnéticos. Sus características principales son: acceso directo a la información y la velocidad de transmisión.

Los algoritmos que utilizan discos magnéticos para el almacenamiento de los arreglos auxiliares, para una mayor eficiencia en los métodos, procuran optimizar el tiempo de acceso a los datos. Los algoritmos que hemos descrito requieren un mínimo de cambios para ser implementados en los procesos de ordenamiento en disco e, inclusive, resultan ser más sencillos, debido a que ya no es necesario restringirse a las limitaciones de acceso secuencial características de las cintas magnéticas.

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO EL NUMERO QUE CORRESPONDA.

1. Ordenar	( ) Shell sort.
2. Llave	( ) Quich sort.
3. Método de intercambio	( ) Nombre general que se le da a los métodos que se practican sobre datos almacenados en la memoria primaria.
4. Método de intercalación	( ) En este método se determina el lugar que debe ocupar la llave dentro del conjunto ya ordenado y se recorren los datos una posición para dejarle su lugar.
5. Método de inserción	( ) Se escoge un elemento del conjunto y se excluye, después se practica el mismo procedimiento sobre los que quedan.
6. Método de selección	( ) Campo de un registro que se utiliza como base para efectuar un ordenamiento.
7. Método de distribución	( ) Se trasponen pares de datos que están fuera de orden hasta que todos están ordenados.
8. Ordenamientos internos	( ) Se basa en la formación de un nuevo conjunto ordenado a partir del entremezclamiento de dos conjuntos de datos.
9. Ordenamientos externos	( ) Así se les llama a los métodos de ordenamiento que se utilizan cuando los datos están en un dispositivo de almacenamiento secundario (disco o cintas).
	( ) Establecer una precedencia entre los elementos de una estructura de datos.
	( ) Se reparte primero en grupos o casilleros análogamente al proceso que usa un clasificador de tarjetas.
	( ) Método de la burbuja.
	( ) Método del heap.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

1. Ordenar el siguiente grupo de llaves con el método del heap. Mostrar claramente la fase de construcción y la de borrado:

   8, 1, 5, 9, 7, 12, 14, 6, 10, 4, 11

2. Ordenar con el método de distribución (Radix o bucket) el siguiente grupo de llaves y mostrar para cada pasada la distribución en las cubetas. Las llaves son de longitud 5 y sólo contienen vocales:

   IOEAU, AEAEA, AIOUA, IEOUA, EAIOU, OUAEA, UAEOU, EUAEU

3. Considerar las siguientes ocho llaves: 6, 5, 7, 1, 3, 2, 4, 8. ¿Cuántas comparaciones e intercambios son requeridas para los siguientes métodos?

   a. Burbuja
   b. Selección directa
   c. Quick

4. Aplicar el método de un POLIFASE de 4 archivos auxiliares al siguiente grupo de llaves. Muestre claramente las dos fases de que se compone el método (distribución e intercalación) M = 3.

   1200, 1100, 1600, 2000, 100, 300, 700, 500, 400, 200, 600, 900, 1000, 800, 1300.

UNIDAD VII METODOS DE BUSQUEDA

OBJETIVO GENERAL

El alumno aplicará el método de búsqueda apropiado a conjuntos de datos residentes, tanto en la memoria principal, como en la memoria secundaria.

OBJETIVOS ESPECIFICOS

Al finalizar el estudio de esta unidad, el alumno:

1. Explicará la operación de búsqueda.
2. Seleccionará el método de búsqueda apropiado a un problema.
3. Explicará la diferencia entre los métodos de búsqueda por comparación de llaves y por transformación de llaves.
4. Aplicará la técnica de resolución de colisiones adecuada a un problema.

INTRODUCCION

El desarrollo de esta unidad se inicia con una definición de la operación de búsqueda. Posteriormente se analizan dos grupos importantes de operaciones: búsquedas por comparación de llaves y búsquedas por transformación de llaves.

Los métodos por comparación de llaves presentados son la búsqueda lineal y la binaria, mientras que los de transformación de llaves son los de suma, multiplicación, división y transformación de base.

También se explica el problema de las colisiones y su resolución por direccionamiento abierto y encadenamiento.

VII.1 GENERALIDADES

En el procesamiento de información se tienen estructuras de datos sobre las cuales se efectúan diversas operaciones solicitadas por el sistema. En general, las operaciones requeridas con mayor frecuencia en estas estructuras formadas por archivos y tablas son las de agregar, borrar y consultar elementos. La consulta a su vez, puede ser de escritura (modificación de un elemento existente), de lectura (obtención de los datos de un elemento existente) o de verificación (revisión de la existencia de un elemento).

La realización de cualquiera de las operaciones mencionadas anteriormente, requiere de una operación común denominada: búsqueda sobre las estructuras de datos.

En todos los casos se asumirá que las estructuras de datos están formadas por nodos que tienen un campo de llave y un campo de información.

VII.2 DEFINICION DE LA OPERACION DE BUSQUEDA

La operación de búsqueda sobre una estructura de datos es aquella que permite localizar un nodo en particular si es que éste existe.

Operación de búsqueda

Esta operación se puede realizar de muy diversas formas, las cuales se agrupan bajo la clasificación de comparación de llaves y transformación de llaves. La selección de cualquier técnica dependerá del tiempo de búsqueda y de las características de los datos.

VII.3 BUSQUEDA POR COMPARACION DE LLAVES

Existen varias técnicas agrupadas bajo esta clasificación con la característica común de realizar la comparación directa de la llave buscada con las llaves de los nodos en la estructura de datos.

Búsqueda por comparación de llaves

VII.3.1 LINEAL

La búsqueda lineal es el método más sencillo, pues consiste en el recorrido secuencial de principio a fin de los elementos de una lista, pudiendo estar almacenada en forma continua o ligada.

Búsqueda lineal

Este método es aplicable tanto a estructuras sin ordenar como a estructuras ordenadas. En cualquier caso, es útil cuando la lista es pequeña y reside en la memoria principal de la computadora, ya que si es grande deberá considerarse en su procesamiento el tiempo de acceso a memoria secundaria, adicionalmente al tiempo de recorrido de la lista.

A continuación se presenta un algoritmo de búsqueda lineal considerando que se tiene una lista de nodos $N_1, N_2, \ldots, Nn$ con llaves $K_1, K_2, \ldots, Kn$ donde $n \geq 1$ y x es la llave del registro buscado:

Algoritmo de búsqueda lineal

```
DESDE  I=1  HASTA N
  SI  K(I)=X
    ESCRIBE('BUSQUEDA CON EXITO')
    ALTO
  FIN
FIN
ESCRIBE ('BUSQUEDA SIN EXITO')
```

La eficiencia de este algoritmo puede medirse tomando como parámetros el número de comparaciones en el caso promedio y en el peor caso. Si suponemos que la lista contiene n elementos, el número promedio de comparaciones para buscar un elemento es de n/2, mientras que en el peor caso (cuando el elemento no se encuentra en la lista) se requieren n comparaciones.

Eficiencia de la búsqueda lineal

Ejemplo VII.1

Considere un vector V formado por los siguientes doce elementos:

V(1)=6	V(5)=0	V(9)=11
V(2)=7	V(6)=4	V(10)=10
V(3)=3	V(7)=23	V(11)=9
V(4)=15	V(8)=22	V(12)=17

entonces:

a. Para saber si se encuentra el número 23 en este vector, se necesitan siete comparaciones.

b. Se necesitan realizar doce comparaciones para darse cuenta que un elemento no está en la lista, siendo éste el peor caso.

c. El número promedio de comparaciones para buscar un elemento en el vector es aproximadamente igual a n/2=6.

VII.3.2 BINARIA

La búsqueda binaria o por bisección es de gran utilidad,

Búsqueda binaria

debido a su sencillez y velocidad, pues está basada en la bipartición repetida del intervalo de búsqueda.

Este método se aplica sobre datos ordenados almacenados en forma contigua, de acuerdo a los siguientes pasos:

- Determinar el elemento central de la tabla.
- Comparar el elemento buscado con el elemento central.
- Si el elemento buscado es igual al buscado, el proceso termina; en caso contrario continúa de acuerdo a:

Si el elemento buscado es mayor (menor) que el elemento central, desechar la primera (segunda) mitad de la tabla y considerar la segunda (primera) mitad de la tabla como la siguiente tabla de búsqueda.

- Regresar al primer punto y continuar hasta que se encuentre el elemento buscado o la tabla de búsqueda esté vacía.

Algoritmo de búsqueda binaria

El procedimiento anterior puede formalizarse para una lista de nodos N_1, N_2, ..., Nn con llaves K_1, K_2, ..., Kn ordenadas en forma ascendente donde $n \geq 1$ y x es la llave del registro buscado. El algoritmo siguiente utiliza las variables INF, MED y SUP para indicar los límites inferior, medio y superior del intervalo de búsqueda:

```
INF=1
SUP=N
ENTANTO  INF≤SUP
  MED=⌊(INF+SUP)/2⌋
  SI  X<K(MED)
      SUP=MED-1
  OSI  X>K(MED)
       INF=MED+1
  OBIEN
       ESCRIBE('BUSQUEDA CON EXITO')
       ALTO
  FIN
  ESCRIBE('BUSQUEDA SIN EXITO')
FIN
```

Eficiencia de la búsqueda binaria

La eficiencia de este método puede obtenerse considerando que en cada iteración el intervalo de búsqueda se divide por la mitad, por lo que el número de comparaciones para el caso promedio y el peor caso tiende hacia $\lceil \log_2 n \rceil$ en la medida en que n aumenta.

Este método presenta ventajas sobre el lineal, pues el número de comparaciones es menor; sin embargo, se ha determinado experimentalmente un punto óptimo de uso para cada uno de estos métodos. (Véase figura VII.1).

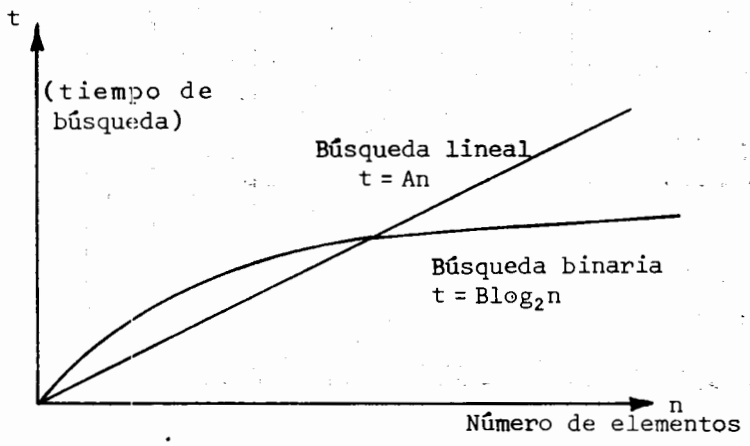


Figura VII.1 Comparación entre la búsqueda lineal y la búsqueda binaria.

El punto de cruce de las curvas depende de la computadora empleada que determina el valor de las constantes A y B.

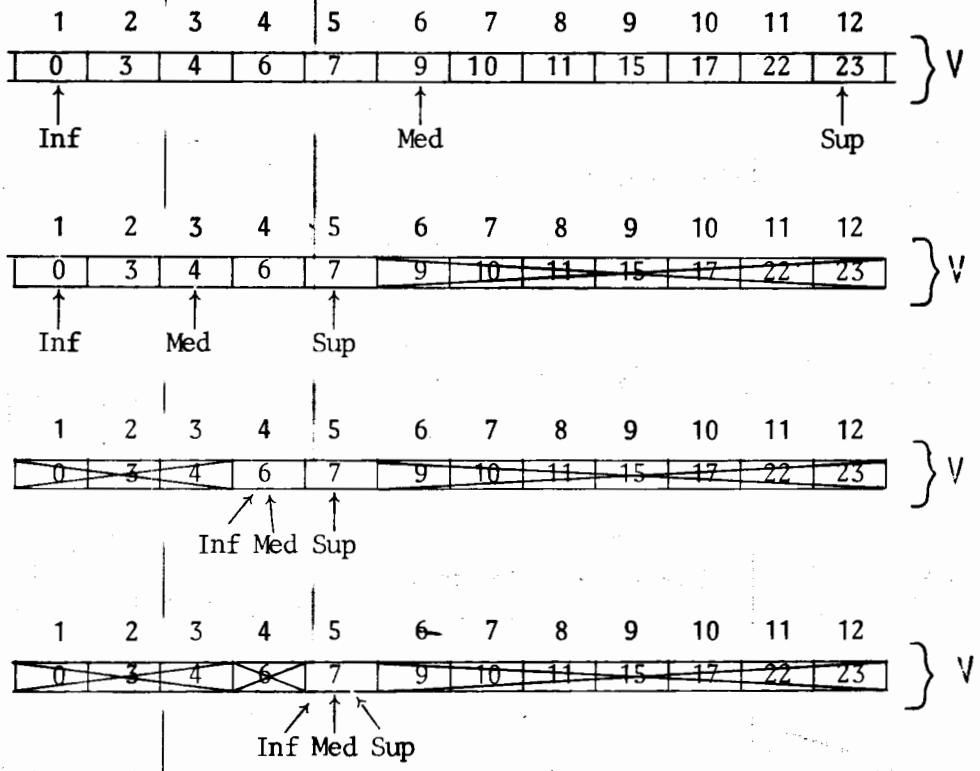
Ejemplo VII.2

Considere un vector V formado por los siguientes doce elementos:

V(1)=Ø	V(5)=7	V(9)=15
V(2)=3	V(6)=9	V(1Ø)=17
V(3)=4	V(7)=1Ø	V(11)=22
V(4)=6	V(8)=11	V(12)=23

entonces:

a. Para saber si se encuentra el número 7 en este vector, se aplica el algoritmo anterior como sigue:



b. El número promedio de comparaciones para el peor caso es aproximadamente igual a $\lceil \log 2(n) \rceil = 4$

VII.4 BUSQUEDA POR TRANSFORMACION DE LLAVES

Búsqueda por transformación de llaves

Las técnicas de búsqueda por transformación de llaves (hash), están basadas en la idea de calcular una dirección en forma directa, a partir de la llave de un nodo. Esto puede expresarse utilizando una función de mapeo H: K → Y, donde K es el dominio de la función formado por el conjunto de llaves que pueden ser transformadas y Y es el rango de la función formado por un conjunto de números enteros que representan a las direcciones de almacenamiento.

Con este método se tiene un tiempo de búsqueda independiente del número de nodos en la estructura y su eficiencia es proporcional al tiempo utilizado para llevar a cabo la transformación.

Es importante mencionar que la técnica de hash se utiliza, tanto en la recuperación de información como en el almacenamiento de la misma.

VII.4.1 FUNCIONES DE HASH

Existen muchas funciones de transformación (hashing funtions) que permiten realizar el mapeo anterior, siendo importante la elección de la función adecuada, ya que de ello dependerá en parte la eficiencia de los algoritmos de almacenamiento y recuperación de información.

En lo siguiente se consideran funciones de mapeo que pueden usarse sin que se conozca previamente la distribución inicial de los nodos y que utilizan las llaves o parte de las mismas, independientemente de si son numéricas o alfabéticas, como cadenas de bits sobre las que se llevan a cabo las transformaciones.

Hash por suma

El método por suma (folding) secciona la llave en varias partes que se suman y de cuyo resultado se toman m bits para formar $n = 2\uparrow m$ direcciones. Entonces, n representa el número de localidades que tiene la tabla y cuyas direcciones están dadas por los elementos del rango de la función, es decir $(\emptyset, 1, 2, \ldots, 2\uparrow m - 1)$. Otra posibilidad consiste en tomar m dígitos decimales de la suma resultante para formar hasta $n = 10\uparrow m$ direcciones en cuyo caso el rango de la función está formado por los elementos $(\emptyset, 1, 2, \ldots, 10\uparrow m - 1)$.

Ejemplo VII.3

Usando el método de suma, obtener la función de mapeo para las llaves K1 = 201, K2 = 1024 y K3 = 564, usando m = 2 dígitos decimales y n = 100 elementos en la tabla. Se utilizarán los dígitos de la extrema derecha de la suma.

En este caso se sumará cada uno de los dígitos que forman la llave para obtener la función pedida.

a. $H(K1) = H(201) = 2 + \emptyset + 1 = 3$

b. H(K2) = H(1024) = 1 + Ø + 2 + 4 = 7

c. H(K3) = H(564) = 5 + 6 + 4 = 15

El método por multiplicación se basa en la idea de multiplicar la llave por una cierta constante, pudiendo ser ella misma, para tomar m bits de este producto y direccionar n = 2↑m localidades. Otra posibilidad consiste en tomar m dígitos decimales del producto resultante para formar hasta n direcciones.

Hash por multiplicación

Ejemplo VII.4

Usando el método por multiplicación, obtener la función de mapeo considerando los dos dígitos centrales y n = 1ØØ para las llaves K1 = 32, K2 = 41, K3 = 55. Usar como constante para cada caso el mismo valor de la llave.

a. H(K1) = H(32) = 2

b. H(k2) = H(41) = 68

c. H(k3) = H(55) = 2

Ejemplo VII.5

Encontrar la función H(K1) correspondiente a K1 = PEREZ al multiplicarse por sí misma. Para este caso se asignará un valor a cada letra de acuerdo a su posición en el alfabeto, se obtendrá la suma de estos valores y se elevará al cuadrado el resultado. En este caso se tomarán los dos dígitos de la extrema derecha, por lo que n = 1ØØ. Entonces el número que se elevará al cuadrado será 72 con lo que se obtendrá H(K1) = H(72) = 84.

Se ha observado que este método da mejores resultados al tomar los bits o dígitos de la parte media del producto.

El método por división consiste en tomar la llave Ki y dividirla por un número n, usando el residuo como la dirección del nodo. La definición anterior corresponde a la función módulo, por lo que esto puede expresarse de la siguiente forma:

Hash por división

H(K) = K Mod n con rango (Ø, 1, ..., n - 1)

Ejemplo VII.6

Utilizando el método de división, encontrar la función H(K1) correspondiente a K1 = PEREZ, cuando n = 16.

Consideremos que cada letra tiene un valor equivalente a la posición que ocupa en el alfabeto. Entonces el número al que se aplica la división se obtiene de: P+E+R+E+Z=17+5+19+5+27, quedando H(K1) = (72 Mod 16) = 8.

Este método tiende a preservar la distribución existente en el dominio de la función, por lo que los grupos de llaves con características similares serán mapeados a direcciones iguales. En general se ha visto que se obtienen buenos resultados, si el divisor es un número primo cercano a la longitud de la tabla.

Hash por conversión de base

El método por conversión de base considera que los bits de la llave representan un número en la base P, el cual se debe convertir a la base Q (P>Q), de donde se toman m bits para formar la dirección en un rango de (Ø, 1, 2, ..., 2↑m - 1) o m dígitos decimales para tener n = 1Ø↑m direcciones.

Ejemplo VII.7

Usando el método de conversión de base, obtener la función de mapeo para las llaves K1 = 94, K2 = 18 y K3 = 3, usando los dos dígitos decimales de la extrema izquierda.

Para este caso se supondrá que las llaves son números en la base 1Ø y que deberán convertirse a la base 8.

a. H(K1) = H(94) = 13

b. H(k2) = H(18) = 22

c. H(K3) = H(3) = 3

Con los métodos expuestos anteriormente, se ve que es posible inventar una gran cantidad de funciones de transformación que pueden ser utilizadas con éxito.

VII.4.2 COLISIONES

Las funciones de transformación mas conocidas no son biyectivas, por lo que para un mapeo de H:K→Y, el rango de la función no es igual a Y, y distintos elementos de K no son mapeados en distintos elementos de Y. Una excepción son las funciones de mapeo de índice usadas en los arreglos y matrices para calcular la dirección relativa de sus elementos, ya que estas funciones son biyectivas y por consiguiente existe una correspondencia uno a uno entre los elementos de K y Y.

La razón por la cual no todas las funciones de hash son biyectivas es eminentemente práctica, pues cada elemento del espacio de llaves K, definiría una clase de equivalencia, lo cual se reflejaría directamente en el número de nodos asignados para almacenamiento de información.

Esto puede entenderse si consideramos un espacio K formado por llaves alfabéticas de cinco letras, en donde se pueden generar 26↑5=11,881,376 posibles combinaciones, mientras que en la práctica sólo se utilizaría un subconjunto de ellas.

Lo importante es encontrar una función de transformación que disperse uniformemente las direcciones calculadas sobre el espacio asignado para su almacenamiento. Idealmente, deberían existir clases de equivalencia con el mismo número de elementos en cada una de ellas.

Debido a lo anterior, el problema de las colisiones, surge cuando se obtiene una dirección para un nodo y ésta ya había sido asignada previamente a otro nodo, por lo que debe contarse con algún método para buscar un espacio disponible. Existen varias técnicas denominadas técnicas de resolución de colisiones, mediante las cuales se resuelve este problema, siendo las más importantes las de direccionamiento abierto y las de encadenamiento.

Colisiones

La técnica de direccionamiento abierto, revisa la tabla en forma secuencial a partir del punto donde ocurrió la

Resolución de colisio-

colisión hasta encontrar el elemento buscado o un espacio libre. Para efecto de este recorrido se considera a la tabla circular.

nes por direccionamiento abierto

Cuando la búsqueda se realiza para insertar un elemento, se asigna la primera localidad disponible al nodo, pero si se trata de una consulta o una supresión, el encontrar una dirección disponible indica que el elemento no se encuentra en la tabla.

El direccionamiento abierto es sencillo de realizar, pero presenta un fenómeno de agrupamiento de nodos alrededor de una misma dirección (clustering), ya que, al irse llenando la tabla, aumenta el número de colisiones y los nodos pertenecientes a una misma clase de equivalencia tienden a estar acomodados contiguamente. Al consultar la tabla, esto tiene el inconveniente de realizar una búsqueda secuencial si el nodo sufrió alguna colisión, lo cual baja la eficiencia de la búsqueda.

Ejemplo VII.8

a. Realizar una inserción sobre la siguiente tabla de diez elementos:

Posición	Contenido
Ø	6Ø
1	59
2	--
3	23
4	4
5	--
6	56
7	66
8	18
9	29

El elemento a insertar por el método de división será el número 31 y se usará el módulo 1Ø. Entonces se obtendría H(31) = 31 Mod 1Ø = 1, pero al encontrarse ocupada esta posición en la tabla se revisarían las siguientes posiciones en forma secuencial hasta encontrar la tercera localidad disponible.

b. Si ahora utilizamos la misma tabla para buscar el elemento 59, pasaríamos por las posiciones nueve, cero y uno.

Resolución de colisiones por encadenamiento

La técnica de encadenamiento forma una lista ligada para cada clase de equivalencia, de tal forma que siempre es posible revisar esta lista secuencialmente para agregar, borrar o consultar nodos. Entonces cada posición en la tabla perteneciente a una cierta clase de equivalencia tendrá un apuntador a la siguiente dirección ocupada y el último nodo indicará el fin de la lista. Esta técnica es más flexible y puede aumentar la rapidez de la búsqueda.

Ejemplo VII.9

Usando el método de la división, realizar la búsqueda del número 69 sobre la siguiente tabla de diez elementos:

Posición	Contenido	Liga
Ø	6Ø	Δ
1	59	5
2	--	--
3	23	Δ
4	4	Δ
5	69	Δ
6	56	7
7	66	Δ
8	18	Δ
9	29	1

Para encontrar el número deseado se recorre la lista formada por los elementos en las posiciones nueve, uno y cinco.

Eficiencia de los métodos de transformación de llaves

De lo expuesto anteriormente es posible establecer que la eficiencia de los algoritmos de búsqueda por transformación de llaves depende de la función de mapeo y de la técnica de resolución de colisiones. Para aumentar esta eficiencia es común asignar un número de loçalidades de almacenamiento mayor al número de nodos a insertar, pues se facilita la asignación de espacios libres. Entonces, se puede establecer la proporción de la tabla que está ocupada como un factor de carga dado por:

α = Num. nodos a insertar/Num. nodos para almacenamiento

CUESTIONARIO DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS EL NUMERO QUE CORRESPONDA.

1.	Búsqueda	( ) Función de mapeo que permite transformar la llave de un nodo en una dirección.
2.	Colisión	( ) Se compara directamente la llave buscada con las llaves de los nodos.
3.	Función de hash	( ) Técnica de resolución de colisiones en la cual se forma una lista ligada para cada clase de equivalencia.
4.	Búsqueda binaria	( ) Localización de un nodo en particular, si es que existe dentro de una estructura de datos.
5.	Búsqueda por comparación de llaves	( ) En estos métodos, el tiempo de búsqueda es independiente del número de nodos en la estructura.
6.	Búsqueda por transformación de llaves	( ) Problema que resulta cuando una función escogida de hash asigna a un nodo, una dirección que ya había sido asignada a otro.
7.	Encadenamiento	( ) Métodos de búsqueda basados en el empledo de una función de hash.
8.	Direccionamiento abierto	( ) En este método por comparación es indispensable que las llaves estén ordenadas para que se pueda aplicar.
		( ) Al ocurrir una colisión en esta técnica se revisa la tabla en forma secuencial hasta encontrar el elemento buscado o un espacio libre.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

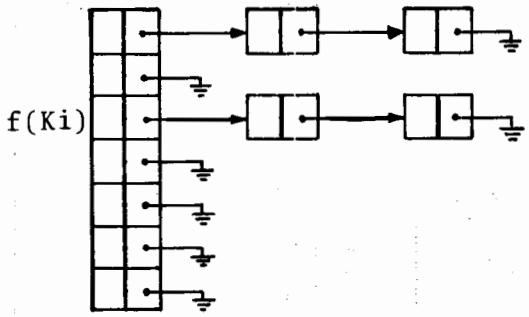
1. Sea el siguiente grupo de llaves:

63, 94, 111, 125, 204, 209, 250, 290, 310, 348, 420

a. Calcular el número de comparaciones necesarias para localizar la llave 290 por los métodos directo y búsqueda binaria.

b. Calcular cuál es la máxima cantidad de comparaciones que habría que efectuar en el peor caso para los métodos de búsqueda directa y binaria.

2. Hacer un programa para agregar y retirar registros a una estructura como la que se muestra a continuación:



f = método de la división

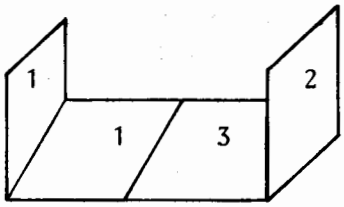
$1 \leq Ki \leq 32\ 000$

$0 \leq f(ki) \leq 99$

Deberá generar para la prueba 200 llaves para altas y 200 llaves para bajas.

3. El método de doblado para el cálculo de las direcciones se puede explicar con la siguiente figura:

Ki = 1132



$$\begin{array}{r} 13 \\ +\ 12 \\ \hline 25 \end{array}$$

f(Ki) = 25

Aplicar este método al grupo de llaves de la izquierda y generar su dirección a la tabla de la derecha. Si hay colisión manejarla por el método de direccionamiento abierto.

$K_1 = 1132$

$K_2 = 2011$

$K_3 = 1111$

$K_4 = 2000$

$K_5 = 2014$

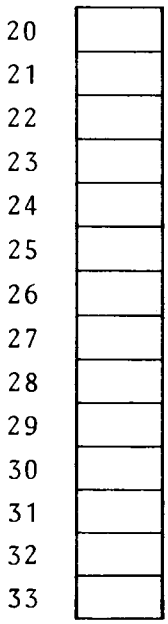
$K_6 = 1101$

$K_7 = 2009$

$K_8 = 1105$

$K_9 = 1100$

$K_{10} = 1133$



B I B L I O G R A F I A

1. BASSE, S.
Computer Algorithms: Introduction to design and Analisys
Addison Wesley
E.U.A. 1978.

2. BERZTISS, A. T.
Data Structures, Theory and Practice
Academic Press
E.U.A. 1975.

3. GEAR, C.W.
Computer Organization and Programming
McGraw-Hill
E.U.A. 1980.

4. KNOTT, G.D.
Hashing Functions
The computer Journal, Volumen 18, Número 3

5. KNUTH, D.E.
The Art of Computer Programming
Addison Wesley
E.U.A. 1975.
Volumen 1, Fundamental algorithms.

6. KNUTH, D.E.
The Art of Computer Programming
Addison Wesley
E.U.A. 1975.
Volumen 3, Sorting and Searching

7. LORIN, H.
Sorting and Sort Systems
Addison Wesley
E.U.A. 1975.

8. MacEWEN, G.
Introduction to Computer Systems
McGraw-Hill
E.U.A. 1980.

9. MADNICK, S. y DONOVAN, J.
Operating Systems
McGraw-Hill
E.U.A. 1974.

10. PAGE, E. y WILSON, L.
Information Representation and Manipulation in a Computer
Cambridge University Press
Gran Bretaña, 1978.

11. PFALTZ, J.
Computer data Structures
McGraw-Hill
E.U.A. 1977.

12. TREMBLAY, J. y MANOHAR, R.
Discrete Mathematical Structures with Applications to Computer Science
McGraw-Hill
E.U.A. 1975.

13. TREMBLAY, J. y SORENSON, P.
An Introduction to Data Structures with Applications
McGraw-Hill
E.U.A. 1976.

A P E N D I C E

ESTRUCTURAS DE CONTROL EMPLEADAS PARA LA ESCRITURA DE LOS ALGORITMOS QUE SE PRESENTAN EN ESTE MATERIAL.

1. SECUENCIA

```
CODIGO  X
```

2. DECISION

a.
```
SI <PREDICADO 1>
     CODIGO A
OSI <PREDICADO 2>
     CODIGO B
OSI <PREDICADO 3>
     CODIGO C
.....................
OBIEN
    CODIGO Z
FIN
```

b.
```
SI <PREDICADO>
    CODIGO A
OBIEN
    CODIGO B
FIN
```

c.
```
SI <PREDICADO>
    CODIGO A
FIN
```

3. ITERATIVAS

a.
```
DESDE I = <EXPRESION 1>  HASTA <EXPRESION 2> PASO N
      CODIGO A
FIN
```

b.
```
EN TANTO <PREDICADO>
    CODIGO A
FIN
```

c.
```
REPETIR
    CODIGO A
HASTA <PREDICADO>
```

EXAMEN DE AUTOEVALUACION

I. RELACIONAR LA COLUMNA DE LA DERECHA CON LA COLUMNA DE LA IZQUIERDA, ESCRIBIENDO DENTRO DEL PARENTESIS LA LETRA QUE CORRESPONDA.

1.	Estructura de Datos	( ) La burbuja, el quick, el shell son métodos de ordenamiento.
2.	Lista	( ) Una colisión es un problema en la búsqueda por ...
3.	Arreglo	( ) Localizar una llave en una estructura de datos, es un problema de ...
4.	Lista lineal	( ) Es un conjunto de datos que no puede crecer o disminuir.
5.	Lista no lineal	( ) Un árbol binario es una lista ...
6.	Búsqueda	( ) Es un conjunto de datos que puede crecer o disminuir.
7.	Ordenamiento	( ) La pila, cola, cola doble son listas ...
8.	Archivo	( ) Establecer un orden de precedencia en una estructura de datos es una operación de ...
9.	Comparación de llaves	( ) El heap sort es un método de ordenamiento por .....
10.	Transformación de llaves	( ) El algoritmo de polifase de 4 cintas es un ordenador.
11.	Ordenamiento interno	( ) Conjunto de registros almacenados en dispositivos de memoria secundaria.
12.	Ordenamiento externo	( ) Conjunto de nodos o registros que guardan importantes relaciones entre sí.

II. RESOLVER LOS SIGUIENTES PROBLEMAS.

1. Considere los siguientes datos técnicos de una cinta magnética:

- Densidad de grabación = 1600 BPI
- Número de pistas = 7
- Velocidad de L/E = 45 pulgadas/segundo

a. ¿Qué longitud de cinta se requiere para grabar un caracter ASCII?

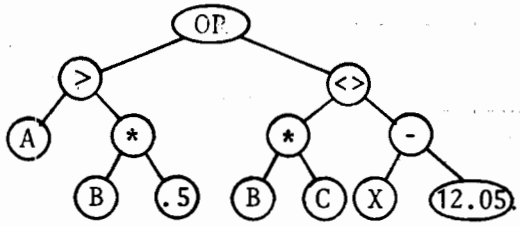
b. ¿Cuántas pulgadas de cinta se requieren para almacenar una caja de 10,000 tarjetas, con 80 caracteres ASCII cada una?

c. ¿Cuánto tiempo se requiere para grabar los 800,000 caracteres?

d. ¿Cuántos caracteres por segundo puede transferir la unidad?

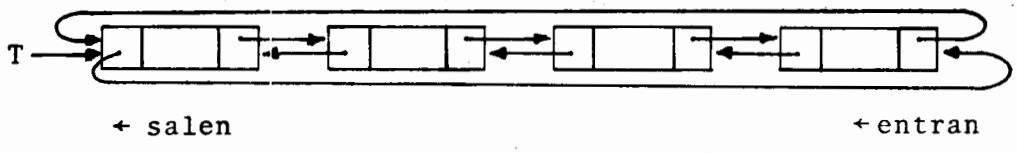
2. Considere el siguiente árbol binario que representa la expresión:

[A > (B * .5)] OR [(B * C) <> (X - 12.05)]



a. Recorra el árbol en postorder.
b. Transforme el árbol a un árbol binario de Knuth.
c. Utilizando el resultado del recorrido postorder, escriba un algoritmo para evaluar la expresión, utilizando una pila.

3. Escriba un algoritmo para agregar elementos a la siguiente cola.



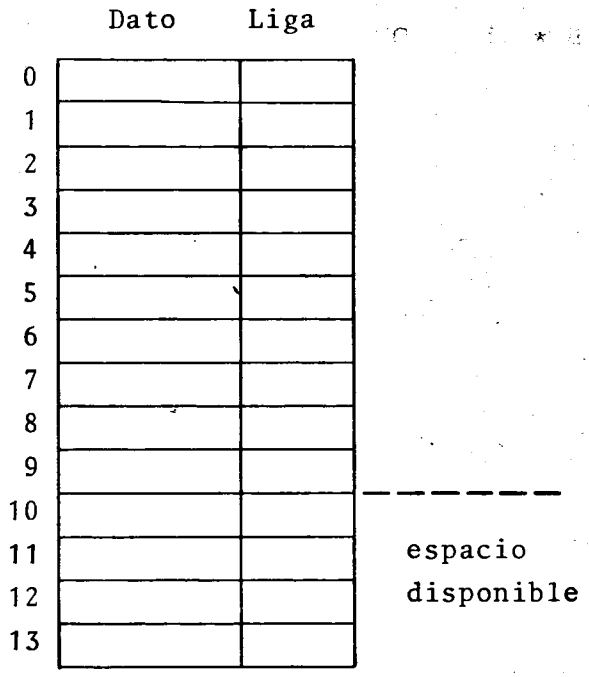
4. Muestre cada uno de los pasos del algoritmo heap para ordenar el grupo de llaves siguiente:

   (8, 5, 4, 2, 1, 9, 10, 14, 3, 7)

5. Aplique el método de la división al grupo de llaves siguiente y genere su dirección en una tabla de 10 elementos.

   Maneje las colisiones con el método de encadenamiento, usando el área de espacio disponible que se marca en la tabla.

Llaves
AA
AZ
ZA
BG
LC
EA
IH
RH
OB
WA



UNIDAD I ELEMENTOS PARA EL ESTUDIO DE LAS ESTRUCTURAS DE DATOS

I. Reactivos:

1. (2)
2. (3)
3. (1)
4. (2)
5. (7)
6. (4)
7. (5)
8. (6)
9. (4)
10. (7)

II. Reactivos:

1. Verdadero
2. Falso
3. Verdadero
4. Falso
5. Falso
6. Verdadero
7. Falso
8. Verdadero
9. Falso
10. Verdadero
11. Verdadero
12. Verdadero
13. Verdadero
14. Verdadero

III. Problemas:

1. 32 bits

2.

900	P	O	902	
902	R		904	
904	M	I	906	
906		R	908	
908	A	Z	910	
910	A		912	
912	H	A	914	
914	B	L	916	

916	A	R	918	
918	A		920	
920	E	L	922	
922		E	924	
924	S	P	926	
926	I	R	928	
928	I	T	930	
930	U		Δ	

ligado

900	P	O	R	
901	M	I		R
902	A	Z	A	
903	H	A	B	L
904	A	R	A	
905	E	L		E
906	S	P	I	R
907	I	T	U	

contiguo

3. 12,598 caracteres
4. 6,500 bits/m = 165.10 bpi

   1.538×10^{4} m = 6.057×10^{-3} in

UNIDAD II ESTRUCTURAS DE DATOS ELEMENTALES

I. Reactivos:

1. (4)
2. (3)
3. (2)
4. (1)
5. (7)
6. (6)
7. (5)
8. (10)
9. (8)
10. (9)

II. Problemas:

1. 1023, 215, 57

2.

+0	00000	+ 8	01000	-16	10000	-8	11000
+1	00001	+ 9	01001	-15	10001	-7	11001
+2	00010	+10	01010	-14	10010	-6	11010
+3	00011	+11	01011	-13	10011	-5	11011
+4	00100	+12	01100	-12	10100	-4	11100
+5	00111	+13	01101	-11	10101	-3	11101
+6	00110	+14	01110	-10	10101	-2	11110
+7	00111	+15	01111	- 9	10111	-1	11111

3.

-7	11111001	+3	00000011
+2	00000010	-5	11111011
-5	11111011	-2	11111101

4. $.2047 \times 10^{7}$, $.0001 \times 10^{-8}$, $-.2047 \times 10^{7}$, 3 dígitos

5. ASCII 31_{16}, 39_{16}, 41_{16}, 42_{16}, 43_{16}
   EBCDIC $F1_{16}$, $F9_{16}$, $C1_{16}$, $C2_{16}$, $C3_{16}$

6. Se almacena sólo el triangular inferior

   Si $j > i$

   $i \longrightarrow j$

   $$\text{Loc } A(i,j) = BA + \frac{i(i-1)}{2} + j$$

7. $$\text{Loc } A(i,j) = BA + n(i-1) - \frac{i(i-1)}{2} + j$$

8. $$\text{Loc } A(i,j) = T(i) + j$$

9. I y J son los índices del elemento pedido, B es el apuntador al inicio de la tabla y en INFO se regresa el valor pedido.

```
REN ← B
COL ← B
DESDE JJ = 1  HASTA  J  HACER
    COL ← LIGA-DER (COL)
FIN
DESDE II = 1  HASTA  I  HACER
    REN ← LIGA-IZQ (REN)
FIN
IREN ← REN
INFO ← O

EN TANTO (INFO=O) Y (LIGA-IZQ(COL) ≠ Δ) Y (LIGA-DER(IREN) ≠ Δ)
     EN TANTO (LIGA-DER(IREN)≠LIGA-IZQ(COL)) Y (LIGA-DER(IREN)≠ Δ )
          IREN ← LIGA-DER (IREN)

     FIN
     SI (LIGA-DER (IREN) = LIGA-IZQ (COL)
            INFO ← VALOR (LIGA-IZQ (COL))
     O BIEN
            IREN ← REN
            COL ← LIGA-IZQ (COL)
     FIN
FIN
```

UNIDAD III. ESTRUCTURAS DE DATOS COMPUESTAS: LISTAS LINEALES

I. Reactivos:

1. (10)	6. (8)
2. (6)	7. (7)
3. (3)	8. (2)
4. (1)	9. (5)
5. (4)	10. (9)
	11. (4)

II. Problemas:

1. a.	pila, cola, cola doble
b. y g.	pila, cola doble
c. d. e. y f.	cola doble

2. Agregar

```
SI SE OBTIENE N
   DATO(N) ← DATO
   LI(N) ← Δ
   LD(N) ← TOPE
   SI TOPE ≠ Δ
      LI(TOPE) ← N
   FIN
   TOPE ← N
OBIEN
   PILA LLENA (OVERFLOW)
FIN
```

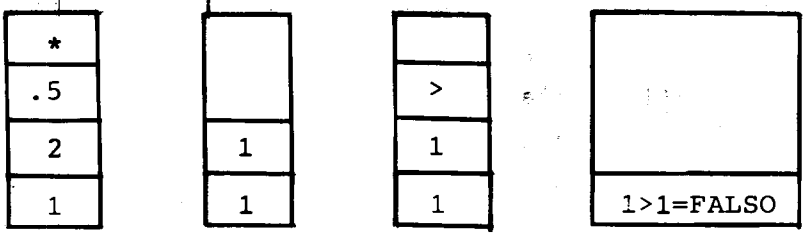
Retirar:

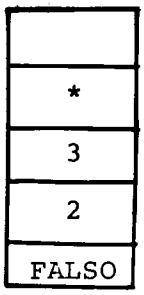
```
SI TOPE=Δ
   PILA VACIA (UNDERFLOW)
OBIEN
   DATO ← DATO (TOPE)
   N ← TOPE
   TOPE ← LD(TOPE)
   SI TOPE ≠ Δ
      LI(TOPE)← Δ
   FIN
   RETORNAR N
FIN
```

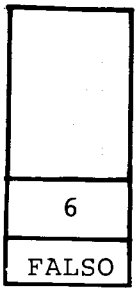
3. SI TOPE=Δ ó LD(TOPE) = LI(TOPE)

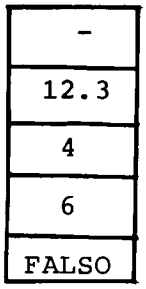
```
   LISTA DE CERO O UN ELEMENTO
OBIEN
   APT ← TOPE
   REPETIR
        T ← LD(APT)
        LD(APT) ← LI(APT)
        LI(APT) ← T
        APT ← LI(APT)
        HASTA QUE (APT = TOPE)
FIN
```

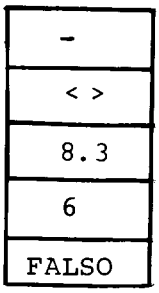
4. AB.5 * > BC * D 12.3 - < > OR DC + AB - ≤ AND

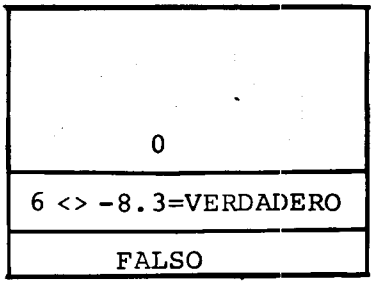


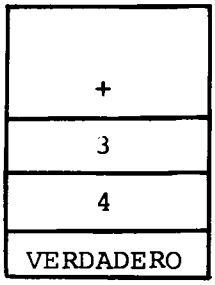


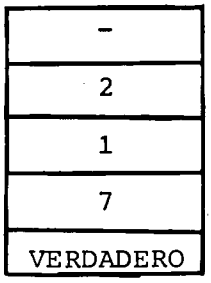


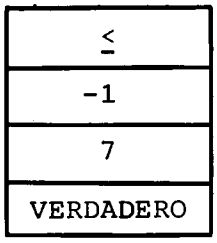


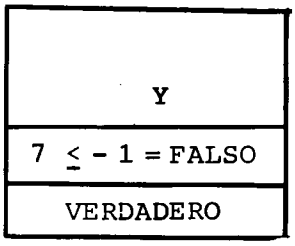


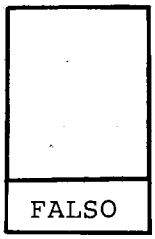












5.
```
FUNCTION RAIZ (n, a, e)
SI (a² - n) < e
   RAIZ ← A
OBIEN
   a ← (a² + n)/2a
   RAIZ ← RAIZ (n, a, e)
FIN
```

6. De modo muy general se esboza una de las formas de resolver el problema:

```
PROGRAMA GASOLINERA
    LIM-TIEM ← 60 * 60 * 24
    TLLEG ← 1
    DESDE  T=1  HASTA LIM-TIEM
       SI  T=TSAL
           LLAMA A RUTINA SALIDA
       FIN
       SI  T=TLLEG
           LLAMA A RUTINA LLEGADA
       FIN
       SI  T=TSERV
           LLAMA A RUTINA SERVICIO
       FIN
    FIN
       CALCULA CANTIDAD-PROMEDIO-EN-COLA
       CALCULA TIEMPO-PROMEDIO-DE-SERVICIO
       IMPRIME RESULTADOS
FIN

RUTINA LLEGADA:
       GUARDA EL TIEMPO-DE-LLEGADA DEL AUTOMOVIL
       SI HAY COLA=BOMBA OCUPADA
          INCREMENTA CANTIDAD-DE-AUTOS-EN-COLA
          ALMACENA CANTIDAD-DE-AUTOS-EN-COLA Y T
          GUARDA EL AUTO N EN COLA
       OBIEN
          PROGRAMA INICIO DE SERVICIO:  TSERV ← T
       FIN
       PROGRAMA SIGUIENTE LLEGADA:  TLLEG ← T + (ALEATORIO ENTRE 10 y 30)
FIN

RUTINA SERVICIO:
       EXTRAE UN AUTO DE LA COLA Y LO PONE EN LA BOMBA
       DECREMENTA AUTOS-EN-COLA
       GUARDA AUTOS-EN-COLA Y T PARA ESTADISTICAS
       CALCULA TIEMPO-DE-CARGA: TCAR ← CAP-TANQUE (ALEAT.)/VEL-CARGA
       CALCULA CARGA-RADIADOR: TRAD ← (ALEATORIO 1 o 0)* 300
       CALCULA AIRE-LLANTAS: TAIRE ← (ALEATORIO 1 o 0)* 120
```

```
        CALCULA LIM-VIDRIO: TLIMP ← (ALEATORIO 1 o 0)* 60
        CALCULA  AÑADIR-ACEITE: TACEITE ← (ALEATORIO 1 o 0)* 240
        CALCULA TIEMPO-DE-PAGO: TPAGO
        TIEMPO-DE-SERVICIO: TSER ← TCAR + TRAD + TAIRE + TLIMP + ...
        PROGRAMA FIN DE SERVICIO: TSAL ← T + TSER
FIN

RUTINA SALIDA:
        RETIRA AUTO DE BOMBA
        CALCULA DURACION-DEL-SERVICIO=T-TIEMPO DE LLEGADA
        GUARDA DURACION-DEL-SERVICIO PARA ESTADISTICA
        SI HAY COLA
                LLAMA A RUTINA SERVICIO
FIN
```

UNIDAD IV ESTRUCTURAS DE DATOS COMPUESTAS: LISTAS NO LINEALES

I. Reactivos:

1. 3	10. 16
2. 10	11. 6
3. 18	12. 9
4. 14	13. 14
5. 21	14. 7
6. 12	15. 5
7. 17	16. 13
8. 11	17. 8
9. 4	18. 15

II. Problemas:

1. a. <a,c>, <c,d>, <d,f>, <f,h>
   b. <a,b>, <b,d>, <d,f>, <f,h>, <h,g>, <g,e>, <e, c>
   c. <a,c>, <c,d>, <d,g>, <g,e>, <e,c>, <c,d>
   d. <g,e>, <e,c>, <c,g>
   e. b, f, h, g, e
   f. d, c, g

2. a.

	a	b	d	d
a	1	1	1	1
b	1	1	1	1
c	1	1	1	1
d	1	1	1	1

b.

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	0	0	0	0	0
3	1	1	0	0	0	0
4	1	1	1	0	0	0
5	1	1	1	1	0	0
6	1	1	1	1	1	0

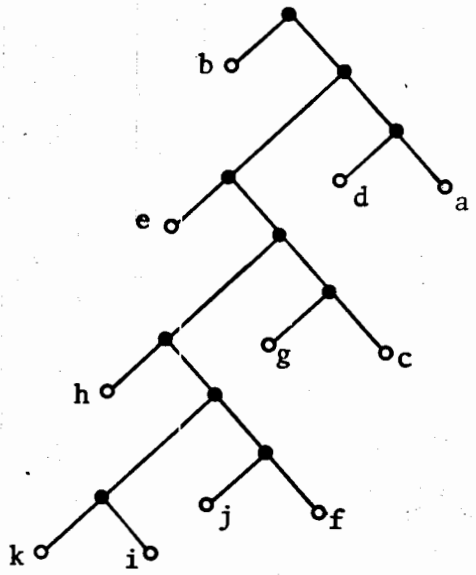
c.

	1	2	3	4	5
1	1	1	0	0	0
2	1	0	1	0	0
3	0	0	0	1	1
4	0	0	0	0	0
5	0	0	0	0	1

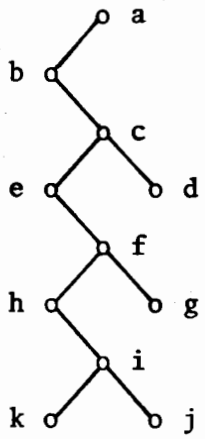
3.
```
LEE X (NODO INICIAL)
LEE Y (NODO FINAL)
GUARDA X EN UNA PILA Y MARCALO COMO VISITADO
MIENTRAS LA PILA NO QUEDE VACIA
                SI  X=Y
                      RECORRER NODOS MARCADOS EN LA PILA
                      (TRAYECTORIA DE X A Y)
                      FIN DEL ALGORITMO
                OSI HAY NODOS ACCESIBLES DESDE X QUE NO ESTEN
                    EN LA PILA MARCAR X
                    GUARDARLOS EN LA PILA
                OBIEN
                    SACAR UN ELEMENTO DE LA PILA
                    Y SI EL SIGUIENTE ESTA MARCADO TAMBIEN
                FIN
                    HACER X IGUAL AL CONTENIDO DEL TOPE DE LA
                    PILA Y MARCARLO COMO VISITADO
FIN
```

4. (F)
(V)
(V)
(F)
(V) F
(F)
(F)
(F)
(F)
(F)

5. Arbol estrictamente binario



Arbol de Knuth



6. Nivel/Nivel (Top-down) = A + * / B C + G D * E F
Nivel/Nivel (Bottom-up) E F D * B C + G * / A + =
Preorder = A + * B C / + D * E F G
Inorder A = B * C + D + E * F / G
Postorder A B C * D E F * + G / + =

```
7.  R = RAIZ DEL ARBOL
    FUNCION POSTORDER (R)
            SI LI(R) ≠ VACIO Y NO SE HA VISITADO
                  R = LI(R)
                  POSTORDER(R)
            FIN
            SI  LD(R) ≠ VACIO Y NO SE HA VISITADO
                  R = LD(R)
                  POSTORDER(R)
            FIN
            SE VISITA DATO(R)
    RETURN
```

UNIDAD V ARCHIVOS

I. Reactivos:

1. (2)
2. (7)
3. (1)
4. (4)
5. (5)
6. (6)
7. (3)
8. (6)
9. (2)

II. Problemas:

1. a. 563
   b. 48
   c. 288,000

2. a. Factor de bloqueo = 10
   b. TCB = 48 bytes
   c. DLCB = (23-1)*(48)+2*32 = 1120
   d. NRF = (1120 DIV 512) + 12 = 14
   e. D = 1120 MOD 512 = 96
   f. NT = 14 DIV 10 = 1
   g. NRDP = 14 MOD 10 = 4

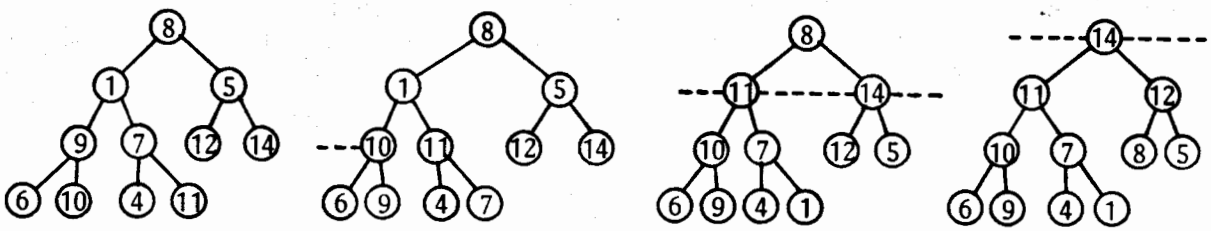
UNIDAD VI METODOS DE ORDENAMIENTO

I. Reactivos:

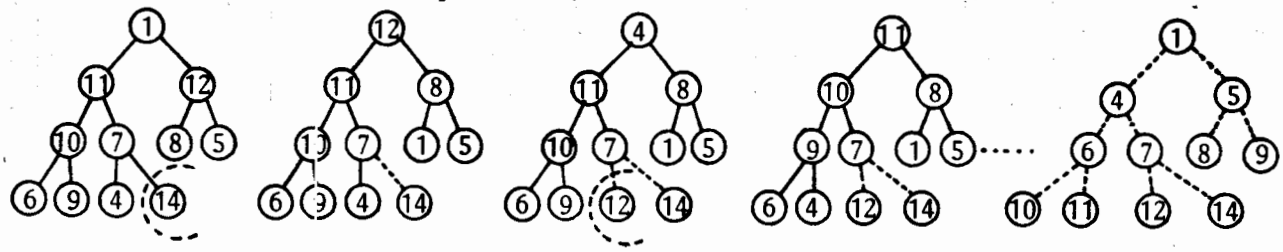
1. (5)
2. (3)
3. (8)
4. (5)
5. (6)
6. (2)
7. (3)
8. (4)
9. (9)
10. (1)
11. (7)
12. (3)
13. (6)

II. Problemas:

1. Fase I. Construcción del heap



Fase II. Borrado del heap



2.

d_1

AEAEA	AIOUA	IEOUA	OUAEA	IOEAU	EAIOU	UAEOU	EUAEU
A				U			

d_2

IOEAU	AEAEA	OUAEA	EUAEU	EAIOU	UAEOU	AIOUA	IEOUA
A		E		O		U	

d_3

AEAEA	OUAEA	EUAEU	IOEAU	UAEOU	EAIOU	AIOUA	IEOUA
A			E		I	O	

d_4

UAEOU	EAIOU	AEAEA	IEOUA	AIOUA	IOEAU	OUAEA	EUAEU
A		E		I	O	U	

d_5

AEAEA	AIOUA	EAIOU	EUAEU	IEOUA	IOEAU	OUAEA	UAEOU
A		E		I		O	U

3. a. Burbuja

1 NC = 7	1 NC = 6	1 NC = 5	1 NC = 4	1 NC = 3	1 NC = 2
6 NI = 4	2 NI = 3	2 NI = 3	2 NI = 3	2 NI = 1	2 NI = 0
5	6	3	3	3	3
7	5	6	4	4	4
2	7	5	6	5	5
3	3	7	5	6	6
4	4	4	7	7	7
8	8	8	8	8	8
P_1	P_2	P_3	P_4	P_5	P_6

Total de comparaciones 7 + 6 + 5 + 4 + 3 + 2 = 27

Total de intercambios 4 + 3 + 3 + 3 + 1 = 14

b. Selección directa

6	1	1	1	1	1	1
5 NI = 1	5 NI = 1	2 NI = 1	2 NI = 1	2 NI = 1	2 NI = 1	2 NI = 0
7 NC = 7	7 NC = 6	7 NC = 5	3 NC = 4	3 NC = 3	3 NC = 2	3 NC = 1
1	6	6	6	4	4	4
3	3	3	7	7	5	5
2	2	5	5	5	7	6
4	4	4	4	6	6	7
8	8	8	8	8	8	8
P_1	P_2	P_3	P_4	P_5	P_6	P_7

Total de comparaciones = 28
Total de intercambios = 6

c. Quick

6 5 7 1 | 3 2 4 8

NC = 7 NI = 3

4 5 2 1 | 3 (6) 7 8

---

4 5 2 1 | 3

NC = 4 NI = 3

3 1 2 (4) | 5

---

3 1 2

NC = 2 NI = 1

2 1 (3)

---

2 1

NC = 1 NI = 1

1 (2)

---

7 8

NC = 1 NI = 0

(7) 8

Total de comparaciones = 15
Total de intercambios = 8

4. Fase I. Distribución

T_0 |1100 1200 1600 2000| 200 400 600 900|

T_1 |100 300 500 700 | 800 1000 1300|

Fase II. Intercalación

T_2 |100 300 500 700 1100 1200 1600 2000|

T_3 |200 400 600 800 900 1000 1300|

T_0 |100 200 300 400 500 600 700 800 900 1000
1100 1200 1300 1600 2000|

UNIDAD VII METODOS DE BUSQUEDA

I. Reactivos:

1. (3)
2. (5)
3. (7)
4. (1)
5. (6)
6. (2)
7. (6)
8. (4)
9. (8)

II. Problemas:

1. a. Por método directo se requieren ocho comparaciones, por búsqueda binaria se requieren cuatro comparaciones.

   b. Por método directo se requieren once comparaciones, por búsqueda binaria se requieren cuatro comparaciones.

2.
```
DESDE  i = 1  HASTA 200
       LEE Ki
       f(Ki)=Ki  MOD 100
       RECORRER LISTA A PARTIR DE f(Ki) HASTA
       ENCONTRAR UNA LIGA = Δ
       INSERTAR NODO AL FINAL DE LA LISTA
FIN
```

```
DESDE  i = 1  HASTA 200
       LEE Ki
       f(Ki) = Ki  MOD 100
       RECORRER LISTA A PARTIR DE f(Ki) HASTA
       ENCONTRAR DATO = Ki
       RETIRAR NODO CON DATO = Ki DE LA LISTA
FIN
```

3.

20	2000
21	1101
22	2011
23	1111
24	1100
25	1132
26	2014
27	1105
28	1133

29 2009

30

31

32

33

SOLUCION AL EXAMEN DE AUTOEVALUACION

I. Reactivos:

1. (11)	5. (5)	9. (9)
2. (10)	6. (2)	10. (12)
3. (6)	7. (4)	11. (8)
4. (3)	8. (7)	12. (1)

II. Problemas:

1. a. 1/1600 pulgadas

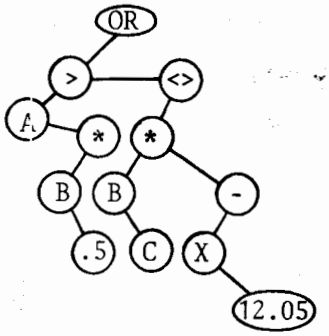
   b. 800 000/1600 = 500 pulgadas

   c. 500/45 = 11.11 segundos

   d. 1600 x 45 = 72 000 caracteres/segundo

2. a. AB.5 * > BC * X 12.05 - <> OR

   b.



   c. TOMAR LA EXPRESION DE IZQUIERDA A DERECHA EN TANTO EXISTAN ELEMENTOS EN LA EXPRESION

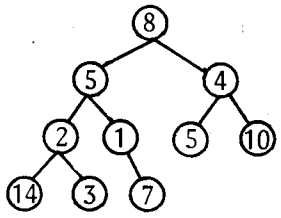
```
      SI  SIMBOLO ≠ OPERADOR
          METERLO A LA PILA
      OSI SIMBOLO = OPERADOR BINARIO
          SACAR DOS ELEMENTOS DE LA PILA
          PRACTICAR LA OPERACION
          DEPOSITAR RESULTADO A LA PILA
      OBIEN
          SACAR UN ELEMENTO DE LA PILA
          PRACTICAR LA  OPERACION
          DEPOSITAR RESULTADOS
      FIN
FIN
```

   El resultado queda en el tope de la pila

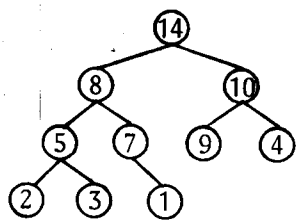
3. Operación para agregar nuevo elemento llamado N

```
LI(N)      LI(T)
LD(N)      T
LD(LI(T))      N
LI(T)      N
```

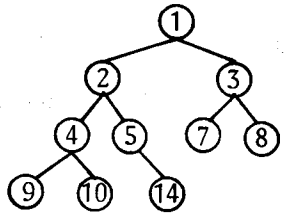
4. Arbol original



Heap



Arbol ordenado



5. Una posible solución es asignar a cada letra un peso igual a la posición que ocupan en el alfabeto, sumar estos valores y aplicar la división por diez para obtener:

```
f(AA) =  2 MOD 10 = 2
f(AZ) = 28 MOD 10 = 8
f(ZA) = 28 MOD 10 = 8
f(BG) =  9 MOD 10 = 9
f(LC) = 15 MOD 10 = 5
f(EA) =  6 MOD 10 = 6
f(IH) = 17 MOD 10 = 7
f(RH) = 27 MOD 10 = 7
f(OB) = 18 MOD 10 = 8
f(NA) = 25 MOD 10 = 5
```

0		
1		
2	AA	Δ
3		
4		
5	LC	13
6	EA	Δ
7	IH	11
8	AZ	10
9	BG	Δ
10	ZA	12
11	RH	Δ
12	OB	Δ
13	NA	Δ

Area de overflow

La composición tipográfica y portada de estos apuntes se hicieron en la Facultad de Ingeniería de la U.N.A.M. Las ilustraciones, impresión y encuadernación se realizaron bajo la supervisión de la Coordinación del Sistema de Universidad Abierta de la U.N.A.M.

Esta edición consta de 1,000 ejemplares y se terminó de imprimir en el mes de noviembre de 1982 en los talleres del S.U.A., calle Medicina No. México 20 D. F.