Unidad III

Árboles

3.1. Propiedades.

Un árbol es un grafo simple en el cual existe un único camino entre cada par de vértices.

Sea **G =(V,A)** un grafo no dirigido. **G** se denomina **ÁRBOL**, si es conexo con **n** nodos y **n-1** aristas y no contiene ciclos.

Formas equivalentes de definir un árbol.

1. Un árbol es un grafo conexo con n nodos y n-1 aristas.
2. Un árbol es un grafo conexo que no contiene ciclos.
3. Un árbol es un grafo con n nodos, n -1 arista y sin ciclos.
4. Un árbol es un grafo tal que entre cualquier par de nodos distintos existe un camino simple único.

Propiedades:

• Existe un único paseo entre dos vértices cualesquiera de un árbol.
• El número de vértices es mayor en uno al número de aristas de un árbol.
• Un árbol con dos o más vértices tiene al menos dos hojas.
Un árbol T (libre) es una gráfica simple que satisface lo siguiente; si v y w son vértices en T, existe una trayectoria simple única de v a w.

3.2. Árboles generadores.

A esta característica general es posible agregar ciertos teoremas de modo de detallar aún más el alcance de la definición. Es asi como el Grafo que contiene a T debe ser conexo, pues de lo contrario no existiría un subgrafo que contuviera todos sus vértices.

En general un grafo G tendrá varios árboles generadores ,como el del ejemplo 1 el cual tiene a lo menos dos arboles generadores T1 yT2.

3.3. Árboles generadores mínimos.

El peso de un árbol generador es la suma de los pesos de las ramas del árbol. Un árbol generador mínimo es uno con peso mínimo.

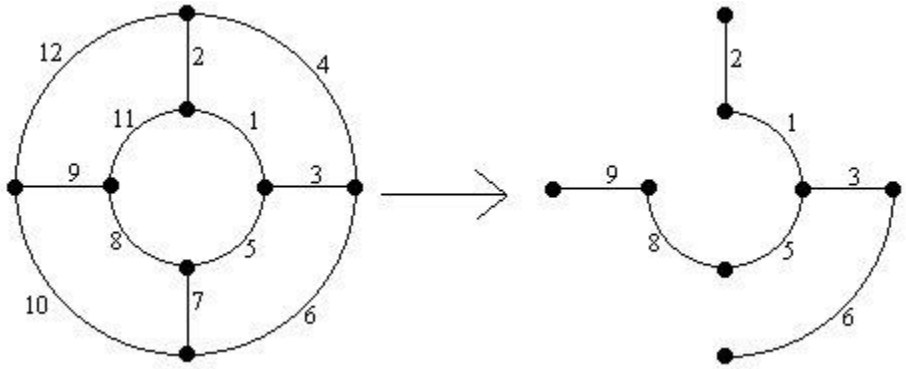
Una interpretación física de este problema es considerar los vértices de un grafo como ciudades, y los pesos de las aristas como los costos de construcción y mantenimiento de vías de comunicación entre las ciudades. Supongamos que queremos construir una red de comunicaciones que conecte a todas las ciudades a un costo mínimo. Entonces el problema es determinar un árbol generador mínimo.

Un procedimiento para resolver este problema se base en la observación de que entre todas las aristas en un circuito, la arista con mayor peso no esta en el árbol generador mínimo. Sea "C" un circuito en un grafo pesado, y "E" la arista con el mayor peso en "C". Supongamos que "E" es una rama de un árbol generador de T. Sea d el conjunto de corte correspondiente a la rama a la rama "E" como el circuito C y el conjunto de corte d deben tener un numero par de aristas en común además de la arista "E" deberán existir al menos una o más aristas que estén tanto en C como en D. Sea F una de estas aristas. Observemos que F es una cuerda del árbol generador t debido a que D es un conjunto de corte. Agreguemos la arista F al árbol generador T y denotemos el subgrafo resultante como U. Es obvio que U es un subgrafo generador que contiene exactamente un circuito, el circuito correspondiente a F. Si eliminamos E de U, obtenemos un árbol generador cuyo peso es menor que T.

Construiremos un subgrafo del grafo pesado paso por paso, al ir examinando cada arista en orden creciente de pesos. Se agregara una arista al subgrafo parcialmente construido si no origina un circuito, y será descartada en caso contrario. La constricción termina cuando todas las aristas han sido examinadas. Es claro que nuestra construcción de origen a un subgrafo que no contiene un circuito. El subgrafo también es conexo. Así el subgrafo construido es un árbol.

Además, este es un árbol generador debido a que el grafo originales es conexo. Finalmente, el árbol generador es mínimo por que en el proceso de construcción una arista era excluida a favor de las aristas de pesos mayores solo si sé sabia que la arista excluida no podía estar en un árbol generador mínimo. En otras palabras, las v - 1 aristas en el subgrafo son efectivamente las v - 1 aristas con los pesos menores que pueden ser incluidas en un árbol generador mínimo.

Ejemplo:



3.5. Ordenamientos

Es la operación de arreglar los registros de una tabla en algún orden secuencial de acuerdo a un criterio de ordenamiento.
El ordenamiento se efectúa con base en el valor de algún campo en un registro.
El propósito principal de un ordenamiento es el de facilitar las búsquedas de los miembros del conjunto ordenado.
Ej. de ordenamientos:
Dir. telefónico, tablas de contenido, bibliotecas y diccionarios, etc.
El ordenar un grupo de datos significa mover los datos o sus referencias para que queden en una secuencia tal que represente un orden, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.
¿Cuándo conviene usar un método de ordenamiento?
Cuando se requiere hacer una cantidad considerable de búsquedas y es importante el factor tiempo.
Tipos de ordenamientos:
Los 2 tipos de ordenamientos que se pueden realizar son: los internos y los externos.
Los internos:
Son aquellos en los que los valores a ordenar están en memoria principal, por lo que se asume que el tiempo que se requiere para acceder cualquier elemento sea el mismo (a[1], a[500], etc).
Los externos:
Son aquellos en los que los valores a ordenar están en memoria secundaria (disco, cinta, cilindro magnético, etc), por lo que se asume que el tiempo que se requiere para acceder a cualquier elemento depende de la última posición accesada (posición 1, posición 500, etc).
Eficiencia en tiempo de ejecución:
Una medida de eficiencia es:
Contar el # de comparaciones (C)

Contar el # de movimientos de items (M)
Estos están en función de el #(n) de items a ser ordenados.
Un "buen algoritmo" de ordenamiento requiere de un orden *nlogn* comparaciones.
La eficiencia de los algoritmos se mide por el número de comparaciones e intercambios que tienen que hacer, es decir, se toma n como el número de elementos que tiene el arreglo o vector a ordenar y se dice que un algoritmo realiza O(n2) comparaciones cuando compara n veces los n elementos, n x n = n2

Algoritmos de ordenamiento:

Internos:

1.
    1. Inserción directa.
    2. Inserción binaria.
2. Inserción directa.
    1.
    2. Selección directa.
3. Selección directa.
    1. Burbuja.
    2. Shake.
4. Intercambio directo.
    1. Shell.
5. Inserción disminución incremental.
    1. Heap.
    2. Tournament.
6. Ordenamiento de árbol.
    1.
    2. Quick sort.
7. Sort particionado.
8. Merge sort.
9. Radix sort.
10. Cálculo de dirección.

Externos:

1. Straight merging.
2. Natural merging.
3. Balanced multiway merging.
4. Polyphase sort.
5. Distribution of initial runs.

Clasificación de los algoritmos de ordenamiento de información:

El hecho de que la información está ordenada, nos sirve para poder encontrarla y accesarla de manera más eficiente ya que de lo contrario se tendría que hacer de manera secuencial.
A continuación se describirán 4 grupos de algoritmos para ordenar información:

Algoritmos de inserción:

En este tipo de algoritmo los elementos que van a ser ordenados son considerados uno a la vez. Cada elemento es INSERTADO en la posición apropiada con respecto al resto de los elementos ya ordenados.

Entre estos algoritmos se encuentran el de INSERCION DIRECTA, SHELL SORT, INSERCION BINARIA y HASHING.

Algoritmos de intercambio:

En este tipo de algoritmos se toman los elementos de dos en dos, se comparan y se INTERCAMBIAN si no están en el orden adecuado. Este proceso se repite hasta que se ha analizado todo el conjunto de elementos y ya no hay intercambios.

Entre estos algoritmos se encuentran el BURBUJA y QUICK SORT.

Algoritmos de selección:

En este tipo de algoritmos se SELECCIONA o se busca el elemento más pequeño (o más grande) de todo el conjunto de elementos y se coloca en su posición adecuada. Este proceso se repite para el resto de los elementos hasta que todos son analizados.

Entre estos algoritmos se encuentra el de SELECCION DIRECTA.

Algoritmos de enumeración:

En este tipo de algoritmos cada elemento es comparado contra los demás. En la comparación se cuenta cuántos elementos son más pequeños que el elemento que se está analizando, generando así una ENUMERACION. El número generado para cada elemento indicará su posición.

Los métodos simples son: Inserción (o por inserción directa), selección, burbuja y shell, en dónde el último es una extensión al método de inserción, siendo más rápido. Los métodos más complejos son el quick-sort (ordenación rápida) y el heap sort.

A continuación se mostrarán los métodos de ordenamiento más simples.

METODO DE INSERCIÓN.

Este método toma cada elemento del arreglo para ser ordenado y lo compara con los que se encuentran en posiciones anteriores a la de él dentro del arreglo. Si resulta que el elemento con el que se está comparando es mayor que el elemento a ordenar, se recorre hacia la siguiente posición superior. Si por el contrario, resulta que el elemento con el que se está comparando es menor que el elemento a ordenar, se detiene el proceso de comparación pues se encontró que el elemento ya está ordenado y se coloca en su posición (que es la siguiente a la del último número con el que se comparó).

Procedimiento ***Insertion Sort***

Este procedimiento recibe el arreglo de datos a ordenar **a[]** y altera las posiciones de sus elementos hasta dejarlos ordenados de menor a mayor. **N** representa el número de elementos que contiene a[].

paso 1: [Para cada pos. del arreglo] For i <- 2 to N do
paso 2: [Inicializa v y j] v <- a[i]
j <- i.
paso 3: [Compara v con los anteriores] While a[j-1] > v AND j>1 do
paso 4: [Recorre los datos mayores] Set a[j] <- a[j-1],
paso 5: [Decrementa j] set j <- j-1.
paso 5: [Inserta v en su posición] Set a[j] <- v.
paso 6: [Fin] End.

Leer más: http://www.monografias.com/trabajos/algordenam/algordenam.shtml#ixzz2mFK0hWO5