ESTRUCTURAS DE DATOS

9 **Relacion 5**

1. La implementación de conjuntos por vectores de bit puede ser utilizada cuando el conjunto universal puede ser trasladado a los enteros entre 1 y N.
   a. Implementar esta estructura.
   b. Describir la función de traslación dada para los siguientes conjuntos universales:
      b.1. Los enteros 0,1,..,99
      b.2. Los enteros n y m para n<=m.
      b.3. Los enteros n, n+2, n+4, ..., n+2k para algún n y k.
      b.4. Los caracteres 'a', ..., 'z'.

2. Suponer que estamos "hashing" enteros mediante una tabla hash con 7 celdas usando la función h(i)=i mod 7.
   a. Comprobar el resultado de insertar los siguientes cubos perfectos: 1,8,27,64,125,216,343.
   b. Repetir a usando una tabla hash cerrada con resolución lineal.

3. Suponer que estamos usando una tabla hash cerrada con 5 celdas y como función h(i)=i mod 5. Comprobar como queda la tabla hash usando una resolución lineal si se inserta la secuencia 23,48,35,4,10 en una tabla vacia.

4. Suponer que se tiene la siguiente secuencia de entradas:
185, 99, 145, 71, 197, 129, 72, 172, 48, 108, 142, 122.
Para una tabla de tamaño 13, y que usamos 'Hashing doble' para marcar las localizaciones:
hi(k)= (hi-1(k) + h0(k)) mod hsiz i=2,3,...
h0(k)= 1+k mod (hsiz-2) con hsiz y hsiz-2 primos relativos
h1(k)= k mod (hsiz)
a. Indicar los contenidos de cada localización de la tabla hash y el número de intentos requeridos para la correspondiente inserción. Suponer que hsiz es igual a 13.
b. Lo mismo que en a si h0(k)=1 ('rehashing lineal').

5. Investigar la resolución de colisiones en una tabla hash usando la secuencia de pruebas:
hi+1(k)= (h1(k) + a * i + b * i * i) mod hsiz i=1,2,...
h1(k)= k mod hsiz, para ciertos a,b,hsiz.
(busqueda de residuos cuadráticos).

6. Diseñar un procedimiento para efectuar borrados a partir de una tabla hash usando rehashing lineal, de forma que vaya reindexando la tabla en el proceso para que se mantengan las secuencias de pruebas válidas.

7. Diseñar una función booleana para detectar la presencia de un determinado substring en un texto, utilizando alguna técnica basada en el uso de tablas hash. Cada string viene caracterizado

por algún número entero (es necesario diseñar por tanto una función para convertir caracteres en enteros dentro de un rango adecuado), y será necesario también el uso de alguna función hash apropiada para realizar la búsqueda.

8. Escribir un procedimiento para construir una tabla hash abierta con B_2 celdas a partir de una tabla hash abierta con B_1 celdas ($B_2>B_1$).

9. Las funciones hash mejoran la resolución de colisiones utilizando "aleatoriedad", es decir, si $h_i(x)$ es la posición de la celda que va a ser intentada después de i colisiones, entonces

$$h_i(x)=(h(x)+d_i) \bmod B$$

para alguna secuencia $d_1,d_2,d_3,...,d_{B-1}$. Una forma de elegir automáticamente esos valores es tomar

$$d_i=(ad_{i-1}+c) \bmod B$$

en donde i>1, y a,c son números reales. Si B=16 encontrar algún valor de a y c para los que la secuencia $d_1,d_2,...,d_{15}$ incluya a todos los enteros 1,2,...,15 (comenzar con $d_0=0$).

10. a. Obtener el árbol parcialmente ordenado que resulta si los enteros 5,6,4,9,3,1,7 se insertan en un árbol vacío.
b. ¿Cual será el resultado de aplicar DELETEMIN tres veces seguidas en el árbol obtenido en a?

11. ¿Cuántos árboles de búsqueda binarios diferentes se pueden construir con los números 10,20,30,40 y 50? Calcula la longitud de búsqueda esperada para cada arbol.

12. Insertar los enteros 7,2,9,0,5,6,8,1 en un árbol binario de búsqueda aplicando reiteradamente el procedimiento de inserción. Obtener el árbol resultante de aplicar el borrado de 7 y de 2.

13. Escriba un procedimiento borra (ABB,clave1,clave2) para eliminar todos los registros con claves entre clave1 y clave2 (inclusive) de un arbol de búsqueda binario.

14. Diseñar procedimientos recursivos y no recursivos para realizar las operaciones de inserción y borrado en árboles binarios de búsqueda.

15. En un sistema de archivos el catálogo de todos ellos está organizado como árbol de busqueda binaria. Cada entrada contiene un nombre de fichero y, entre otras cosas, la fecha de su último acceso, codificada con un entero. Diseñe un procedimiento para borrar los ficheros del catálogo cuyo último acceso sea anterior a una fecha dada.

16. Diseñe algoritmos necesarios para variar las colisiones en las tablas de hash abiertas que utilicen árboles enlazados de búsqueda binarios en vez de listas enlazadas para almacenar las entradas excedentes.

17. Diseñar un procedimiento MEZCLAR(X,Y) para combinar dos colas con prioridad referenciadas por X e Y respectivamente en una sola cola con prioridad referenciada por X.

18. Diseñar procedimientos de inserción y borrado en colas con prioridad implementadas con árboles parcialmente ordenados.

19. Diseñar un procedimiento que dada una lista de enteros construya su ABB asociado. Indicar qué da el recorrido en inorden de dicho árbol.

20. Usualmente en un árbol binario de búsqueda (ABB), los nuevos nodos se insertan en las hojas, pero también es posible hacer crecer un ABB por la raiz, usando una clave de búsqueda K para partir el ABB en 3 componentes: un ABB izquierdo conteniendo todos los nodos con claves menores que K, el propio nodo K y un ABB derecho conteniendo todos los nodos con claves mayores que K.
Diseñar un procedimiento para insertar nuevos nodos en un ABB construido de ésta manera.

21. Un árbol cartesiano es un árbol definido sobre parejas de valores (xi,yi) con la propiedad de que es un árbol binario de búsqueda con respecto a xi y una cola con prioridad con respecto a yi. Puede suponerse que no hay valores duplicados en cada variable.

Más formalmente, y si para un nodo N denotamos sus hijos izquierda y derecha por I, D respectivamente se cumple que:

(a) XI < XN y XN < XD
(b) YN <= YI y YN < YD

Construir el árbol cartesiano para los pares:
(8,35), (21,5), (15,17), (2,22), (12,3), (28,53), (3,48), (6,97) (5,13), y esbozar un procedimiento general de construcción.

22. Se define un P-árbol como un árbol binario parcialmente ordenado que además cumple la condición de que para cada nodo N, su hijo a la derecha tiene un valor intermedio entre él y su hijo a la izquierda, es decir, etiqueta(hizqda(N,A),A) > etiqueta(hdrcha(N,A),A) > etiqueta(N,A). Diseñar un procedimiento de inserción de un nuevo nodo.

23. Los empleados de una cierta compañía se representan en la base de datos de la compañía por su nombre, número de empleado y número de la seguridad social. Construir una estructura de tablas hash que permita acceder al registro de un empleado por cualquiera de estos tres datos. (Nota: No se dispone de memoria suficiente para duplicar los registros de los empleados).

24. Supongamos que disponemos de una estructura de datos conteniendo registros con diversos campos, dos de los cuales **clave1** y **clave2** permiten estar ordenados. Se tiene definida una función hash f sobre el segundo campo y se desea construir un procedimiento que nos inserte esta estructura en un ABB y una **tabla hash** en función de los campos **clave1** y **clave2** respectivamente de modo que no se duplique el espacio a utilizar por un elemento. Definir las estructuras a utilizar y diseñar dicho procedimiento.

25. El departamento de investigación de un gran almacen, decide realizar un estudio relativo a la productividad de las distintas secciones. Con este fin, será necesario obtener para cada sección el volumen de ventas totales en el último mes expresado en millones de pesetas. Supuesto que el almacén tiene un número N de plantas y en cada planta un número de secciones que puede variar de una planta a otra, definir la estructura de datos más optima para mantener la información relativa a las ventas por sección, justificando detalladamente su elección en base a que la cantidad de memoria requerida sea proporcional al número de datos y el tiempo de acceso a la información mínimo.

26. Se desea analizar la productividad en una empresa informática. Tal empresa está dividida en secciones, departamentos y unidades. Cada sección (hasta un total de N), puede tener un número fijo pero indeterminado de departamentos (hasta un total de m) y cada departamente tiene asímismo un número fijo pero indeterminado de unidades (hasta un total de p).

Definir la estructura de datos más eficiente posible. teniendo en cuenta que se ha de emplear un espacio en cada caso proporcional al número de elementos que se tengan, y se ha de minimizar el tiempo de acceso a cada item particular.