Práctico 3

Árboles balanceados (AVL)
Tablas de dispersión (Hash)
Colas de prioridad (Heap)

Clasificación de ejercicios:
(I) Imprescindibles
(R) Recomendados
(C) Complementarios

Árboles balanceados (AVL)

EJERCICIO 1 (I)

1. Definir qué es un árbol AVL, e indicar cuál es la propiedad más importante relacionada con el tema que estamos estudiando en esta pregunta.
2. Dibujar el peor árbol AVL de altura h, para h = 0,1,2,3,4.
3. Indicar en el caso general, cómo generar un peor árbol AVL para una altura h, con $h \geq 2$.
4. Mostrar el AVL resultante de insertar en un árbol de enteros inicialmente vacío, los siguientes elementos en este orden: 3, 5, 16, 12, 8, 7, 9. Mostrar claramente todos los pasos intermedios.

EJERCICIO 2 (C)

Muestre el resultado de insertar 2,1,4,5,9,3,6,7 en un árbol AVL inicialmente vacío.

EJERCICIO 3 (I)

Escriba en C las operaciones para realizar rotaciones simples y dobles sobre árboles AVL. Implemente las rotaciones dobles de dos formas: primero como dos rotaciones simples y luego directamente (sin utilizar funciones auxiliares).

EJERCICIO 4 (I)

Implemente en C el TAD AVL de enteros con las operaciones:

```
Crear       : ∅ → AVL
Vacio       : AVL → boolean
Insertar    : AVL × int → AVL
Borrar      : AVL × int → AVL
Pertenece   : AVL × int → boolean
```

Tablas de dispersión (Hash)

EJERCICIO 1 (I)

Implemente en C tablas de dispersión cerrada con operaciones:

Crear : $\emptyset \rightarrow$ hash
Insertar : hash $\times$ entero $\rightarrow$ hash
Pertenece : hash $\times$ entero $\rightarrow$ bool
Borrar : hash $\times$ entero $\rightarrow$ hash

1. Con resolución lineal de colisiones.
2. Con resolución cuadrática de colisiones.
3. Implemeéntelo ahora con una tabla de dispersión abierta.

EJERCICIO 2 (I)

Suponga que tenemos una tabla de dispersión abierta de 7 buckets, con la función de dispersión h(i) = i % 7.

1. Muestre la tabla de dispersión resultante de insertar los cubos perfectos 1, 8, 27, 64, 125, 216, 343.
2. Repita la parte anterior para una tabla de dispersión cerrada, usando resolución lineal de colisiones.
3. Ahora con resolución cuadrática de colisiones.

EJERCICIO 3 (R)

Dada la entrada 4371, 1323, 6173, 4199, 4344, 9679, 1989 y una función de dispersión h(i) = i % 10 muestre las resultantes:

1. tabla de dispersión abierta
2. tabla de dispersión cerrado con resolución lineal de colisiones
3. tabla de dispersión cerrado con resolución cuadrática de colisiones
4. tabla de dispersión cerrado con una segunda función de dispersión $h_2(i) = 7 - (i \% 7)$.

EJERCICIO 4 (C)

Suponga estamos utilizando una tabla de dispersión cerrada con 5 posiciones y la función de dispersión h(i) = i % 5. Muestre la tabla que resulta si la secuencia 23, 48, 35, 4, 10 es insertada en una tabla inicialmente vacía.

EJERCICIO 5 (I)

Implemente las operaciones del TAD mapping, con dominio y codominio en los enteros, usando tablas de dispersión abierta y cerrada con resolución lineal de colisiones.

Crear	: $\varnothing \rightarrow$ mapping
Insertar	: mapping $\times$ entero $\times$ entero $\rightarrow$ mapping
Pertenece	: mapping $\times$ entero $\rightarrow$ bool
Computar	: mapping $\times$ entero $\rightarrow$ entero

EJERCICIO 6 (I)

i) Definir el TAD Diccionario.

ii) Se tiene un universo de registros con claves entre 1…N , donde N es un entero fijo. Definir una estructura de datos que implemente eficientemente las operaciones del TAD Diccionario. Justificar la respuesta.

iii) Se tiene un universo de registros con claves entre 1::M , donde M es un entero fijo, de los cuales solo se precisa trabajar con un conjunto de N registros con N = O(log M). Definir una estructura de datos que implemente eficientemente las operaciones del TAD Diccionario. Justificar la respuesta.

EJERCICIO 7 (R)

Dado un diccionario con N elementos, se proponen las siguientes 3 implementaciones:

- Un vector de N elementos.
- Un árbol binario de búsqueda balanceado (AVL) de N elementos.
- Una tabla de dispersión de N elementos con colisiones resueltas por encadenamiento.

(a) Indicar una situación en la cual la mejor solución es usar el vector de N elementos, y justificar en una frase su respuesta.
En dicho caso, indicar tiempo de ejecución promedio y peor caso para implementar las operaciones del TAD Diccionario.

(b) Indicar una situación en la cual la mejor solución es usar el árbol binario de búsqueda balanceado de N elementos, y justificar en una frase su respuesta.
En dicho caso, indicar tiempo de ejecución promedio y peor caso para implementar las operaciones del TAD Diccionario.

(c) Indicar una situación en la cual la mejor solución es usar la tabla de dispersión de N elementos, y justificar en una frase su respuesta. En dicho caso, indicar tiempo de ejecución promedio y peor caso para implementar las operaciones del TAD Diccionario.

EJERCICIO 8 (R)

Dado el TAD Diccionario de enteros (acotados entre 1 y N) definido con las siguientes operaciones:

Crear : $\emptyset \rightarrow$ Diccionario
Insertar : Diccionario $\times$ entero $\rightarrow$ Diccionario
Pertenece : Diccionario $\times$ entero $\rightarrow$ bool
Eliminar : Diccionario $\times$ entero $\rightarrow$ Diccionario

Suponga que no se van a ingresar al Diccionario más de (1/10 * N) elementos.

a) Proponga una implementación eficiente para este TAD.

b) Utilizando las primitivas del TAD, implemente las siguiente operaciones:

Intersección: : Diccionario $\times$ Diccionario $\rightarrow$ Diccionario
// Retorna el Diccionario con los elementos que pertenecen a los dos Diccionarios pasados como parámetro */

BorrarMinimo: : Diccionario $\rightarrow$ Diccionario
/* Retorna el Diccionario resultante de eliminar al parámetro su menor elemento */

c) Ahora proponga una implementación eficiente para el TAD con las siguiente operaciones:

Crear : $\emptyset \rightarrow$ Diccionario
Insertar : Diccionario $\times$ entero $\rightarrow$ Diccionario
Pertenece : Diccionario $\times$ entero $\rightarrow$ bool
Eliminar : Diccionario $\times$ entero $\rightarrow$ Diccionario
Intersección: : Diccionario $\times$ Diccionario $\rightarrow$ Diccionario
BorrarMinimo: : Diccionario $\rightarrow$ Diccionario

d) Compare el tiempo de ejecución de las implementaciones propuestas en las partes a) y c), para cada una de las seis operaciones.

Colas de prioridad (Heap)

EJERCICIO 1 (I)

1. Muestre el árbol parcialmente ordenado que resulta si los enteros 5, 6, 4, 9, 3, 1, 7 son insertados en un árbol vacio.
2. Cual es el resultado de tres `BorrarMin` sucesivos en el árbol de la parte anterior.

EJERCICIO 2 (I)

Implemente en C el TAD cola de prioridad de enteros (de tamaño acotado) con las siguientes operaciones:

```
Crear       : int → colaP
Insertar    : colaP × int → colaP
Minimo      : colaP → int
BorrarMin   : colaP → colaP
```

EJERCICIO 3 (R)

En ciertas aplicaciones de colas de prioridad, por ejemplo administración de tareas en un sistema operativo, se desea poder aumentar y disminuir la prioridad de un elemento, así como también borrar elementos arbitrarios. Para ello se extiende el TAD cola de prioridad con operaciones

`DecrementarLlave (x,d,M):` reduce el valor de la clave en `x` una cantidad positiva `d`
`IncrementarLlave (x,d,M):` aumenta el valor de la clave en `x` una cantidad positiva `d`
`Eliminar (x):` se utiliza `DecrementarLlave` y luego `BorrarMin`.

Para que la extensión del TAD con estas operaciones sea efectiva, se debe contar con una estructura adicional que permita dado un valor, determinar cual es su posición dentro del heap (en caso de que se utilice un heap para implementarlo). Diseñe estructuras de datos apropiadas que permitan extender el TAD cola de prioridad con estas operaciones. Implemente el TAD cola de prioridad extendido con las operaciones:

```
Crear            : int → colaPE
Insertar         : colaPE × int → colaPE
Minimo           : colaPE → int
BorrarMin        : colaPE → colaPE
DecrementarLlave : colaPE × int → colaPE
IncrementarLlave : colaPE × int → colaPE
Eliminar         : colaPE × int → colaPE
```

EJERCICIO 4 (R)

Indicar cuales de los siguientes secuencias de enteros se corresponden con la implementación de una cola de prioridad como un heap. En lo que sigue solo se muestran las prioridades de los elementos ordenadas de izquierda a derecha tal como están almacenadas en el arreglo (sin tener en cuenta la posición 0):

a) 23, 45, 78, 12, 90, 3, 5
b) 9, 6, 7, 9, 3, 6, 6
c) 13, 13, 13, 13, 13, 13, 13, 11, 11
d) 1, 2, 3, 4, 5, 6, 7
e) 7, 6, 5, 4, 3, 2, 1

EJERCICIO 5 (C)

La implementación de una cola de prioridad mediante un heap permite encontrar el elemento mínimo en tiempo constante y borrar e insertar un elemento en O(log(n)). Explicar como modificar la estructura de heap y las operaciones de cola de prioridad para proveer la implementación de una cola de prioridad que tiene las siguientes características:

1. La estructura se puede construir en tiempo O(n)
2. Un elemento se puede insertar en tiempo O(log(n)).
3. El máximo y el mínimo se pueden borrar en tiempo O(log(n)).
4. El máximo o el mínimo se pueden encontrar en tiempo constante.

Sugerencia: ordenar los elementos de modo que los nodos en los niveles pares tengan valor mayor que sus descendientes y los nodos en los niveles impares tengan valores menores que los de cualquiera de sus descendientes. Esta estructura se denomina comunmente min-max-heap.

EJERCICIO 6 (C)

Una variante de la estructura descripta en el Ejercicio 5, es el llamado "deap" (double ended heap). Un deap es como un heap, excepto que no hay ningún elemento en la raíz. El sub árbol izquierdo de la raíz es un heap ordenado con el mínimo elemento en la raíz y cada nodo tiene clave menor o igual que la de sus hijos. El sub árbol derecho de la raíz es un heap con el máximo elemento en la raíz y cada nodo tiene clave mayor o igual que la de sus hijos. Cada hoja del heap izquierdo es menor que la correspondiente hoja en el heap derecho, donde el "correspondiente" es el que está en la misma posición en el heap, si existe y sino su padre. Muestre que hay algoritmos para implementar las mismas operaciones que en el Ejercicio 5, con las mismas restricciones de tiempo.

EJERCICIO 7 (C)

Se pretende agregar a las operaciones de cola de prioridad del ejercicio 2, la siguiente:

$$\texttt{DecrementarTodas} : \texttt{colaP} \times \texttt{int} \rightarrow \texttt{colaP}$$

que disminuye en cierto valor las claves de todos los elementos del heap. Discuta si es posible implementar esta operación en O(1) peor caso y en caso afirmativo qué impacto tiene sobre el resto de las operaciones.