Universidad de Granada

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIA

ETSII Informática y de Telecomunicación, C/ Periodista Daniel Saucedo Aranda s/n- 18071- Granada (España)

Estructuras de Datos
Curso 2017-2018. Convocatoria de Enero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas

1. (0.5 puntos) Razonar la verdad o falsedad de las siguientes afirmaciones:

   (a) El orden en que las hojas se listan en los recorridos preorden, inorden y postorden de un árbol binario es el mismo en los tres casos.

   (b) Dado un árbol binario cuyas etiquetas están organizadas como un árbol binario de búsqueda, puedo recuperarlo a partir de su preorden.

   (c) Dado un árbol binario cuyas etiquetas están organizadas como un árbol parcialmente ordenado, puedo recuperarlo a partir de su postorden.

   (d) Es correcto en un esquema de hashing cerrado el uso como función hash de la función h(k)= [k + random(M)] % M, M primo y con random(M) una función que devuelve un número entero aleatorio entre 0 y M-1

   (e) Es correcto en un esquema de hashing doble el uso como función hash secundaria de la función h_0 (x)=[(B-1)-(x % B)] % B con B primo

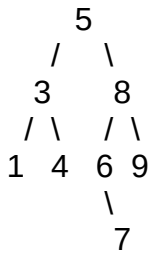
2. (1.5 puntos) Supongamos que tenemos una clase **Liga** que almacena los resultados de enfrentamientos en una liga de baloncesto:

```
struct enfrentamiento{
   unsigned char eq1,eq2; // códigos de los equipos enfrentados
   unsigned int puntos_eq1, puntos_eq2; //puntos por cada equipo
};
class liga{
 private:
   list<enfrentamiento> res;
...
};
```

- Implementa un método que dado un código de equipo obtenga el número de enfrentamientos que ha ganado.
- Implementa la clase iterator dentro de la clase Liga que permita recorrer los enfrentamientos en los que el resultado ha sido el empate. Implementar los métodos begin() y end().

3. (1 punto) Implementa una función **int orden (const list<int> &L);** que devuelva 1 si L está ordenada de forma ascendente de principio a fin, 2 si lo está de forma descendente y 0 si no está ordenada de ninguna forma.

4. (1 punto) Dado un árbol binario de búsqueda, implementa una función para imprimir las etiquetas de los nodos **en orden de mayor a menor profundidad**. Si tienen la misma profundidad pueden aparecer en cualquier orden. Ejemplo:



El resultado seria **7,1,4,6,9,3,8,5.**

5. (1 punto) Tenemos un contenedor de pares de elementos, {clave, bintree<int>} definida como:

```
template <typename T>
class contenedor {
    private:
       unordered_map<T, bintree<int> > datos;
    ……..
    ……..
}
```

Implementa un **iterador** que itere sobre los elementos que cumplan la propiedad de que la suma de los elementos del bintree<int> sea un número par. Debes implementar (aparte de las de la clase iterator) las funciones begin() y end().

6. (1 punto) Un "heap-doble" es una estructura jerárquica que tiene como propiedad fundamental que para cualquier nodo Z a profundidad **par** la clave almacenada en Z es **menor** que la del padre pero **mayor** que la del abuelo (cuando existen), y para cualquier nodo Z a profundidad **impar**, la clave almacenada en Z es **mayor** que la del padre pero **menor** que la del abuelo (cuando existen), siendo el árbol binario y estando las hojas empujadas a la izquierda. Diseña una función para insertar un nuevo nodo en la estructura y aplicarla a la construcción de un heap-doble con las claves {30, 25, 12, 16, 10, 15, 5, 18, 23, 32, 4, 17}.

Tiempo: 3 horas

Preguntas específicas para aquellos estudiantes sujetos a convocatoria adicional o

evaluación única final (4 puntos)

1. (2 puntos) Supongamos que dos personas (usuarios) se insertan en un laberinto de habitaciones con las siguientes características:

- Cada habitación del laberinto tiene una puerta por donde se entra y dos puertas posibles por la que se sale.
- Habrá habitaciones que conduzcan a la calle y por lo tanto finaliza el recorrido por el laberinto.
- Para pasar por una puerta se tira un dado y si sale un número par se sale por la puerta izquierda y si sale impar se sale por la derecha.
- A cada habitación solamente se accede por una única habitación.

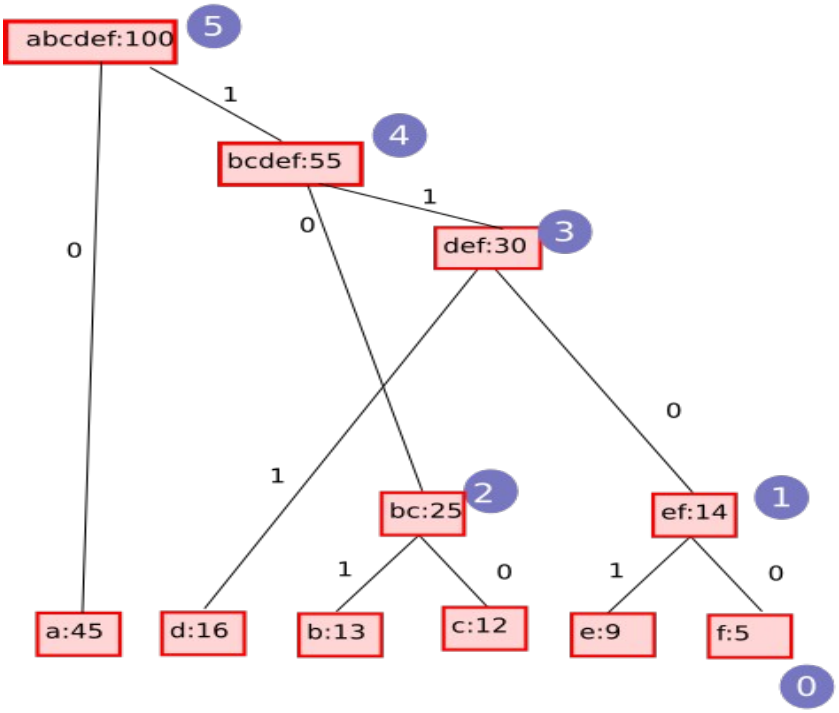
Se pide:
- Establece la representación para el laberinto y usuario
- Implementa la función que dado un laberinto y dos usuarios establezca de entre estos dos usuarios cual de ellos llega antes a la calle.

2.- (2 puntos) Implementa una función que obtenga la codificación Huffman de un conjunto de caracteres. En la siguiente imagen se puede ver un ejemplo de funcionamiento:
Los pasos para obtener el código Huffman de un conjunto de caracteres, con las frecuencias de aparición de cada carácter son:

1. Definir las parejas: frecuencia, carácter asociado.
2. Las parejas frecuencia-carácter insertarlas en una cola de prioridad. La más prioritaria debe ser aquella pareja de menor frecuencia y su carácter asociado.
3. Sacar los dos nodos más prioritarios de la cola. Sumar sus frecuencias y añadir al código de los caracteres correspondientes a 0 y a 1. Insertar un nuevo nodo en la cola con prioridad con frecuencia la suma de las frecuencias de los nodos, y caracteres asociados los sacados.

4. Si la cola está vacía terminar, en otro caso volver a sacar las dos nodos más prioritarios y repetir los pasos anteriores.

En la siguiente imagen se puede ver un ejemplo de codificación Huffman para los caracteres a, b,c,d,e,f, que tienen frecuencias 45,13,12,16,9,5 respectivamente. En el primer paso se introducen los caracteres en la cola de prioridad con sus frecuencias y se inicia el código a “”. A continuación se sacan los dos caracteres más prioritarios (que suman menor frecuencia) a uno se le asigna el código 0 (a todos los caracteres en el nodo) y a todos los caracteres en el otro nodo el código 1. Se repite este proceso hasta que la cola con prioridad esté vacía.
Una vez construido el árbol, hay que calcular la codificación de cada uno de los códigos.



Paso

Cola de Prioridad

0 — f:5 e:9 c:12 b:13 d:16 a:45

1 — c:12 b:13 ef:14 d:16 a:45

2 — ef:14 d:16 bc:25 a:45

3 — bc:25 def:30 a:45

4 — a:45 bcdef:55

5 — abdcef:100

Códigos

a="" c=""
d="" e=""
b="" f=""

a="" c=""
d="" e="1"
b="" f="0"

a="" c="0"
d="" e="1"
b="1" f="0"

a="" c="0"
d="1" e="01"
b="1" f="00"
a="" c="00"
d="11" e="101"
b="01" f="100"

a="0" c="100"
d="111" e="1101"
b="101" f="1100"

Tiempo: 1 hora