Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I
Evaluación continua - 2024/2025

8 de noviembre de 2024

Nombre:

La duración de esta prueba es de 40 minutos. Consta de un solo ejercicio con dos apartados. La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Al finalizar entrega tu solución junto con el enunciado (no es necesario que entregues todas las hojas). Pon tu nombre en cada hoja que entregues.

Ejercicio — 1,5 puntos

Estamos utilizando un árbol AVL para realizar una implementación del Tipo Abstracto de Datos **Diccionario**. Tenemos los siguientes atributos, y no puedes añadir otros atributos en `Diccionario` ni en `Diccionario::Nodo` (los puntos suspensivos corresponden a posibles declaraciones de métodos):

```
class Diccionario {
   struct Nodo {
      float clave;
      int equipo;
      int altura;
      Nodo * izquierdo;
      Nodo * derecho;
      ...
   };
   Nodo * raiz;
   ...
 public:
   ...
};
```

El atributo `clave` es el que determina dónde se sitúan los nodos en el árbol. El atributo `equipo` es de utilidad en el videojuego que estamos desarrollando. En un diccionario no puede haber dos elementos con la misma clave, pero sí que puede haber varios elementos con el mismo equipo. El atributo `altura` es el propio de los árboles AVL para tener calculada en cada nodo la altura del subárbol cuya raíz es ese nodo. Los atributos `izquierdo`, `derecho` y `raiz` tienen el significado habitual.

En cada uno de los siguientes dos apartados, implementa en lenguaje C++ el método que se pide. En ambos apartados, se valorará que la solución no siga recorriendo nodos cuando ya no es necesario.

a) **[1 punto]** Sin utilizar ningún bucle, y sin utilizar ninguna otra clase, implementa un método:

```
int minimaProfundidadEquipo(int e) const
```

que devuelva la mínima profundidad a la que se encuentra un nodo que tenga en el atributo `equipo` el valor `e`. Si no existe ningun nodo que cumpla eso, devolverá -1 (si el árbol está vacío, también).

No confundas profundidad con altura. La profundidad de un nodo mide la distancia desde la raíz hasta ese nodo. Si existen, la raíz está a profundidad 0, los hijos de la raíz están a profundidad 1, los nietos de la raíz están a profundidad 2, etcétera. En la siguiente página se proporciona un ejemplo.

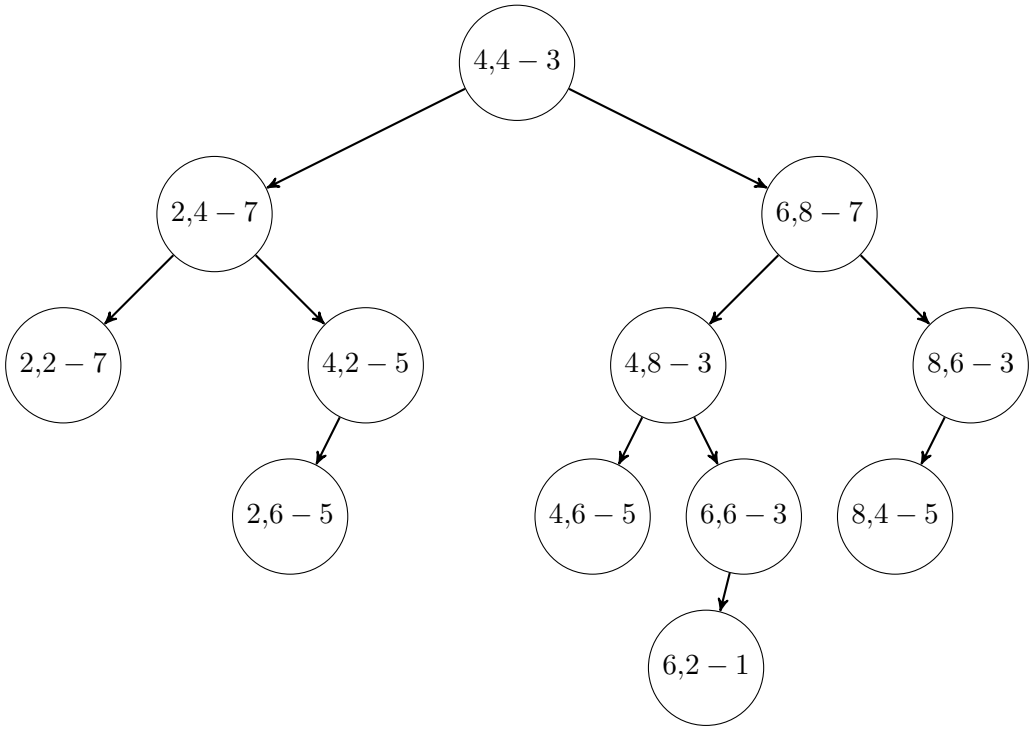
Si haces uso de otras funciones, debes implementarlas también sin utilizar ningún bucle ni otra clase.

b) **[0,5 puntos]** Implementa de nuevo el método del apartado anterior, esta vez sin utilizar recursión (puedes utilizar bucles). Si haces uso de otras funciones, debes implementarlas también sin utilizar recursión. Si lo necesitas, solamente en este apartado, puedes hacer uso en C++ de las clases `stack`, `queue` y/o `priority_queue`, sin tener que implementarlas. Si no recuerdas los nombres de sus operaciones básicas, puedes ponerlos en castellano.

Indica cuáles son los siguientes costes de tus dos soluciones en función de n, siendo n la talla del diccionario (que conincide con la cantidad de nodos del árbol). No es necesario que lo justifiques. En el coste espacial, se pide no tener en cuenta lo que ocupa en memoria el propio árbol.

	Apartado a	**Apartado b**
Coste temporal en el peor caso	$O($ $)$	$O($ $)$
Coste espacial en el peor caso sin contar el árbol	$O($ $)$	$O($ $)$

Ejemplo En cada nodo del siguiente árbol AVL, el número a la izquierda del guión separador es el valor del atributo `clave` y el número a la derecha del guión es el valor del atributo `equipo`:



Con ese árbol:

- el resultado de `minimaProfundidadEquipo(1)` sería 4, que es la profundidad del nodo con clave 6,2;
- el resultado de `minimaProfundidadEquipo(3)` sería 0, que es la profundidad del nodo raíz;
- el resultado de `minimaProfundidadEquipo(5)` sería 2, que es la profundidad del nodo con clave 4,2;
- el resultado de `minimaProfundidadEquipo(7)` sería 1, que es la profundidad de los hijos de la raíz;
- el resultado de `minimaProfundidadEquipo(9)` sería -1.