Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I

Examen final - 2024/2025 - Primera parte
Recuperación del segundo examen parcial

21 de enero de 2025

Nombre:

El examen final consta de dos partes, cada una con una duración de dos horas. La primera parte está destinada a aquellos estudiantes que han solicitado intentar mejorar la calificación obtenida en cualquiera de los exámenes parciales, renunciando a la misma. Este segundo ejercicio de la primera parte está dirigido a quienes buscan mejorar la nota del segundo examen parcial, y tiene una duración de 40 minutos.

La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Escribe tu solución empleando el lenguaje C++, y no olvides los costes que se piden. 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 2 — 1,5 puntos

Estamos utilizando un árbol AVL para realizar una implementación del Tipo Abstracto de Datos **Diccionario**. Contamos con 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 energia;
      int altura;
      Nodo * izquierdo;
      Nodo * derecho;
      ...
   };
   Nodo * raiz;
   ...
 public:
   ...
};
```

El atributo `clave` determina dónde se sitúan los nodos en el árbol. El atributo `energia` es útil para el videojuego que estamos desarrollando y siempre tiene un valor mayor o igual que cero. El atributo `altura` es propio de los árboles AVL y permite conocer la altura del subárbol cuya raíz es ese nodo. Los atributos `izquierdo`, `derecho` y `raiz` tienen el significado habitual.

En los siguientes apartados, tu solución debe ser eficiente. Además, se valorará que no siga recorriendo el árbol innecesariamente.

a) **[1 punto]** Sin utilizar ningún bucle y sin emplear ninguna otra clase, añade a la clase `Diccionario` un método

```
bool evaluarCapacidad(int p, int capacidad) const
```

que devuelva *true* si, y solo si, el valor de `capacidad` es mayor que la suma de los valores del atributo energia de todos los nodos que están a profundidad menor o igual que `p`. Realiza esto asumiendo que el árbol no está vacío y que tanto `p` como `capacidad` son mayores o iguales que 0.

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.

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 coincide con la cantidad de nodos del árbol). No es necesario justificar las respuestas. En el cálculo del coste espacial, no consideres el espacio ocupado por 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	$O($ $)$	$O($ $)$