Problemes 3: Arbres

Estructura de Dades - Curs 2020/2021
Grau d'Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica

Objectiu

Aquests problemes estan enfocats en l'ús i la implementació del TADs arbres en particular els arbres de cerca binària (BST) i els arbres de cerca binària balancejats (AVL) i heaps. S'inclouen tan problemes conceptuals de la definició dels arbres, com d'implementació d'arbres enllaçats en C++ i d'anàlisi computacional del cost de cadascuna de les implementacions.

1. Dibuixa l'arbre binari BST tal i com quedaria si s'afegeixen en l'ordre especificat els següents elements en un arbre buit.

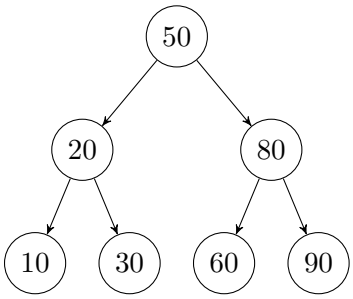
   Elements a inserir: 50, 20, 75, 98, 80, 31, 150, 39, 23, 11, 77

2. Dibuixa l'arbre AVL resultant d'inserir successivament els elements 41, 38, 31, 12, 19, 8 en un arbre AVL inicialment buit. Mostreu cada nova inserció i indiqueu si feu alguna rotació. Per a cada rotació indiqueu el tipus de rotació que es fa. Mostreu en l'arbre AVL l'alçada de cada node i el factor de balanç.

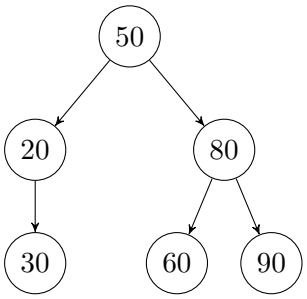
3. Dibuixa l'arbre AVL resultant d'inserir successivament els elements 7, 12 ,10, 16, 4, 2 en un arbre AVL inicialment buit. Mostreu cada nova inserció i indiqueu si feu alguna rotació. Per a cada rotació indiqueu el tipus de rotació que es fa. Mostreu al costat de cada node de l'arbre AVL el seu factor de balanç.

4. Construeix l'arbre binari que mostri el següent recorregut inordre 4,2,5,1,3,6 i el següent recorregut postordre 4,5,2,6,3,1.

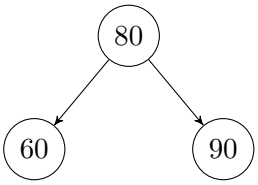
5. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters. Definiu i implementeu en C++ un mètode `subarbre()` que donat un número d'un node, retorni un altre arbre amb els nodes amb clau major que aquest node.



(a) Arbre inicial



(b) subtree(20)



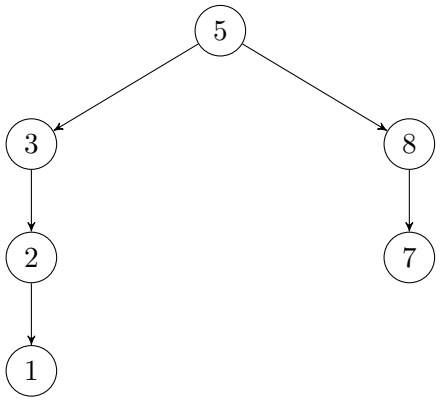
(c) subtree(60)

6. Dibuixa el max-heap resultant d'inserir els elements: 4, 9, 2, 5, 1, 10, 6 en un max-heap binari inicialment buit.

7. És l'array A amb valors (23, 17, 14, 6, 13, 10, 1, 5, 7, 12) un max-heap?

8. Dibuixa l'operació de `removeMax` en el heap A = < 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1 >. Dibuixa el heap i l'array A, amb tots els pasos i tots els canvis que es produeixen.

9. Definiu el mètode anomenat `paintTree` de la classe Tree (un arbre de cerca binària) que recorri en inordre l'arbre i imprimeixi per a cada node, el valor del node i el nivell en el que esta. Per exemple, donat l'arbre:



El que s'imprimirà per pantalla serà:

```
>> 1 4
>> 2 3
>> 3 2
>> 5 1
>> 7 3
>> 8 2
```

Important! en aquest exemple s'ha considerat que el nivell de l'arrel és 1. Es pot resoldre considerant que el nivell del node arrel és 0.

Definiu el cost del mètode `paintTree` que heu implementat.

10. Definiu el mètode anomenat `max()` de la classe Tree (un arbre de cerca binària) Que retorna el valor màxim de l'arbre binari. Suposeu que l'arbre és d'enters. Optimitzeu el codi al màxim.

11. Definiu el mètode anomenat `min()` de la classe Tree (un arbre de cerca binària) Que retorna el valor mínim de l'arbre binari. Suposeu que l'arbre és d'enters. Optimitzeu el codi al màxim.

12. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu de manera iterativa les funcions `void printInorder()`, `void printPreorder()` i `void printPostorder()`. Aquestes funcions recorren l'arbre de manera iterativa en inordre, postordre i preordre respectivament i mostren els valors dels nodes per pantalla (usant una pila auxiliar `STL::Stack`).

13. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu de manera iterativa la funció `void printInWeight()`. Aquesta funció recorre l'arbre en amplada i mostra els valors dels nodes per pantalla (usant una cua auxiliar `STL::Queue`).

14. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, definiu i implementeu la funció `int countChildren()const`. Aquesta funció compta el nombre de descendents en un arbre binari (amb el recorregut en postordre).

15. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu la funció `bool isPerfect()`. A partir de l'arrel d'un arbre, determina si l'arbre és perfecte. Retorna cert si és perfecte, fals en cas contrari.

16. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu la funció `bool equals(const BinarySearchTree<Element>& other)`. Aquesta funció a partir de dos arbres, els compara i retorna cert si són iguals, fals en cas contrari.

17. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu la funció `Tree( const Tree<Element>& other)`. Aquesta funció copia recursivament un arbre, creant-ne un altre d'igual amb un nou espai de memòria.

18. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu la funció `int countNumNodes()const`. Aquesta funció retorna el nombre de nodes de l'arbre.

19. Els arbres que contenen el mateix número de nodes poden variar en forma depenent en l'ordre en que els nodes han sigut inserits als arbres. Una manera de mesurar la forma de l'arbre és l'altura que és el nombre de nodes del camí més llarg des de l'arrel fins a un node fulla. Aquesta mesura és important perquè la quantitat de temps que pren en buscar un node a l'arbre. Donada la classe `Tree` (arbre de cerca binària) que emmagatzema elements enters, implementeu la funció `int getHeight()const` que permeti calcular l'alçada d'un arbre binari usant el recorregut en postordre i la següent funció recursiva de l'alçada d'un arbre al node p.

$$height(p) = \begin{cases} 0 & \text{if } p \text{ is null } (base\ case) \\ \max(height(p \rightarrow \text{left}), height(p \rightarrow \text{right})) + 1 & \text{if } p \text{ is not null } (recursive\ step) \end{cases}$$

20. Donat l'arbre més curt possible d'N diferents claus, desenvolupa l'orde de magnitud (cost computacional) del pitjor-cas de l'execució de les següents operacions del TAD Tree (arbre de cerca binària) i explica el teu raonament sobre la teva estimació.

- search O( )
- insert O( )
- remove O( )
- print O( )

21. Quins són les alçades dels arbres més curts i més llargs que es poden construir amb N claus diferents? Justifica i raona la teva resposta.