LABORATORI 5 - Persistent Tree

Estructures de dades - Curs 2018-19

Marc Junyent

Tasca 1

La primera tasca es basa en la implementació d'una Linked Binary Search Tree inmutable utilitzant nodes enllaçats. Per a la implementació d'aquesta classe és necessaria la interfaç BinarySearchTree, amb les operacions de la qual són: isEmpty, containsKey, get, put i remove, les quals haurem d'implementar de manera que l'arbre no sigui modificat.

Dins de la classe LinkedBinarySearchTree hi haura la classe Node, que estarà formada per una clau(K), un valor(V) i els respectius apuntadors a altres nodes de l'arbre(left i right).

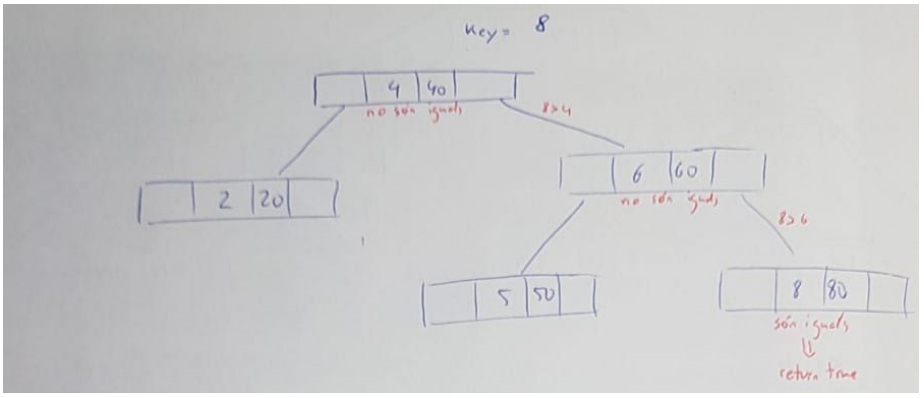
Els mètodes de la interfaç BinnarySearchTree a implementar són:

- isEmpty():

Retorna true en cas que l'arbre estigui buit, és a dir, que l'arrel de l'arbre corresponent sigui igual a null.

- containsKey(K key):

Funció que retorna un booleà i que cerca la clau corresponent dins de l'arbre.
La funció recorrerà recursivament l'arbre en busca de la clau, en cas de no trobar-la retorna fals.
Si el node que li passem per paràmetre és null retorna fals, en cas contrari, comprova si el node corresponent té la clau que estem cercant, en cas afirmatiu retornem true i si no cridem recursivament la funció passant-li com a paràmetre el fill esquerre del node en cas que la clau sigui més petita o el fill dret en cas que la clau sigui més gran.

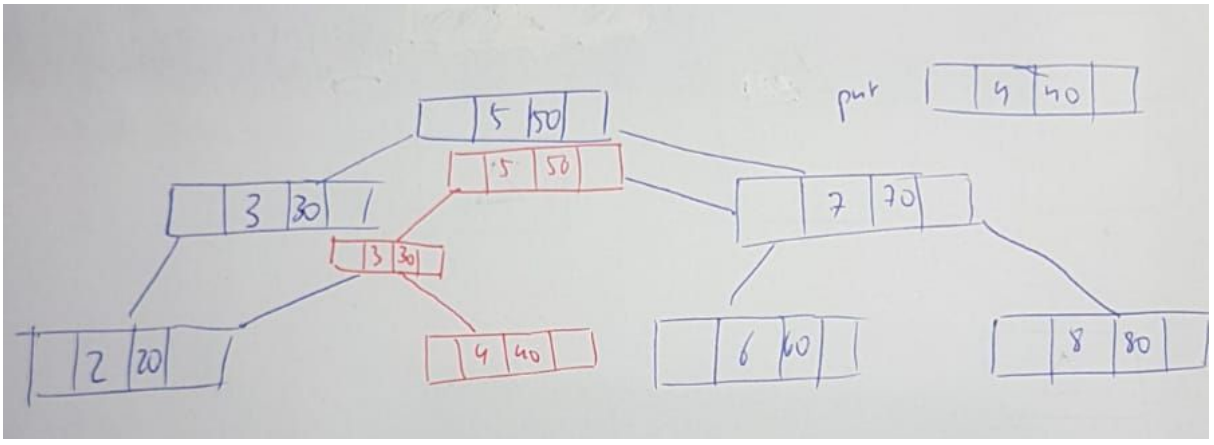


- get(K key):

Aquesta funció té un funcionament semblant a containsKey, també actua recursivament sobre l'arbre.
Si l'arbre no conté key retornarà null, en cas contrari recorrerà recursivament l'arbre a partir de l'arrel.
Quan la clau del node sigui igual a la key, retornarà el valor del node corresponent, si no farà la crida de la funció get passant com a paràmetre el node esquerra o dret del node depenent de si es tracta d'una clau més petita o més gran.

- put(K key, V value)

Funció que afegeix un node nou amb la clau i el valor corresponents. Si key o value valen null es retornarà una excepció, si no s'haurà de retornar un arbre nou amb el node afegit.
Si l'arbre és buit retornarà un arbre on l'arrel sigui el node nou, en cas contrari s'afegirà o s'actualitzarà segons el valor de key.
Si el valor key és igual a algun dels nodes ja existents s'actualitzarà el valor, en cas contrari s'anirà recorrent l'arbre segons el valor de la clau. Es crearà un node nou per a cada un que passi fins a arribar a la posició corresponent. Per a enllaçar els corresponents nodes nous s'utilitzarà una funció recursiva que retorni el node corresponent després de ser enllaçat amb el seu pare.

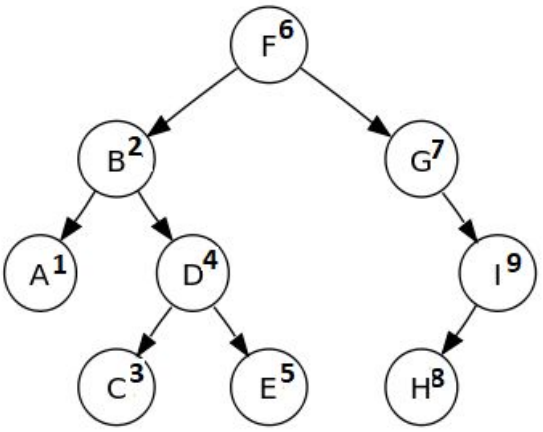


- Remove(K key):

Eliminarà el node que correspon amb la clau. Si la clau és null retornara una excepció i si l'arbre corresponent no conté aquesta clau retornarà el mateix arbre(this).
A l'hora de recorrer l'arbre, quan trobem el node corresponent amb la clau que estem buscant s'haurà de tenir en compte quants fills te.
Si te els dos fills, buscarem el node més gran del subarbre esquerra, ja que serà el substitut del node eliminat. També guardarem el node al qual apuntarà el substitut (RemovedNode).
Si te el fill esquerre o el dret, retornarem el node.left o el node.right, depenent de quin dels dos cas sigui.
Si la clau no es igual al node actual, seguirem cercant fent una crida recursiva de la funcio RemoveNode, que passarà com a parametre el fill esquerra o el dret depenent del valor de key.

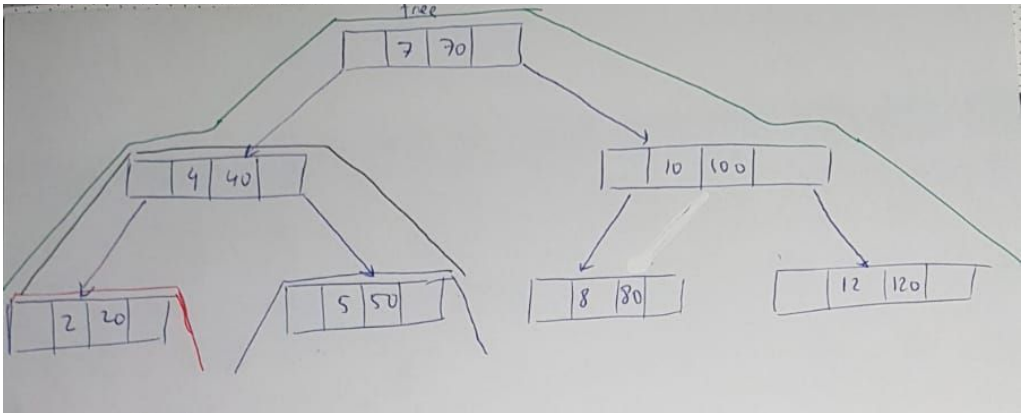
Tasca 2

La tasca 2 consisteix a implementar el recorregut inOrden de l'arbre de cerca per a guardar l'ordre dels nodes de més petit a més gran dins d'un array. L'ordre a seguir és el següent:



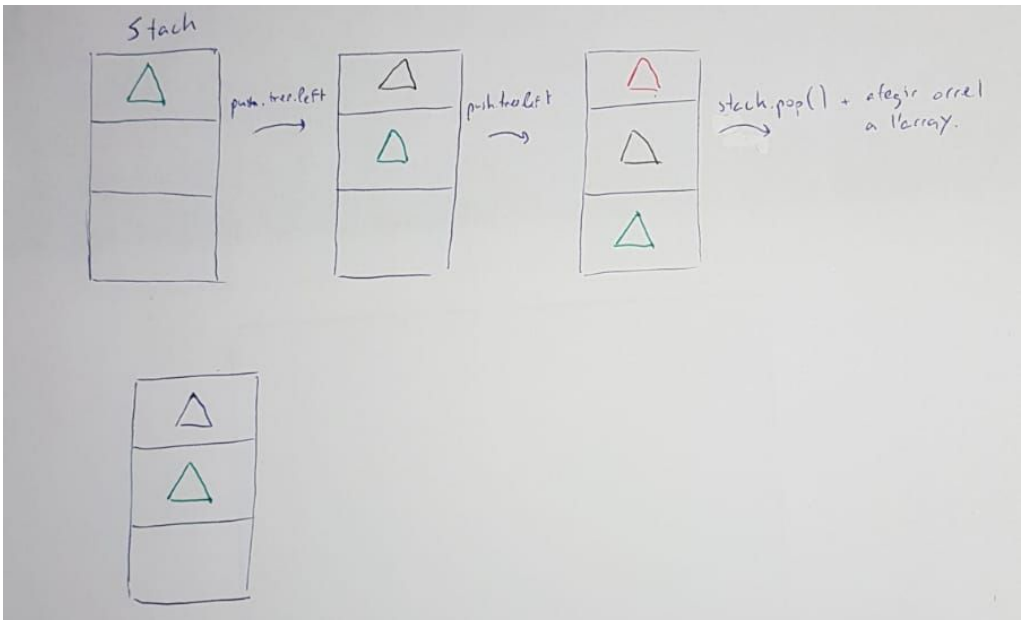
Per a implementar aquest mètode necessitarem un array on guardar les parelles clau i valor a Pair<S,T>, també serà necessari una Stack on anirem guardant tots els subArbres per a després afegir les arrels de cada un a l'array que retornarem.

Suposant aquest arbre:



Es recorrerà l'arbre des de l'arrel afeing tots els subarbres de l'esquerra, un cop afegit l'ultim arbre es farà un pop i s'afegirà la root a l'array, després es recorrerà les branques de la dreta seguint els mateixos passos.

Així doncs, la stack inicialment quedaria així:



Un cop recorregut l'arbre i havent afegit els subarbres a la stack i havent fet els pops corresponents, l'array que retornarem quedarà ordenat segons el valor de la clau dels nodes.