Laboratorio 5

Persistent Tree

Estructura de Dades

2n Curs GEI
Grup 2

15/01/2021

Ton Llucià Senserrich
47125160T

Eduard Sales Jové
49539818A

Apartado 1: Implementación de los árboles binarios de búsqueda inmutables

has2Childs

Només retornarà true en cas de tenir dos fills, un únic fill o cap, es retornarà fals.

containsKeyRec i containsKey

La funció containsKeyRec recursiva que ens serveix per obtenir la referència a un node a partir de la seva clau, la funció està implementada de manera que compara si la clau és més gran o més petita que el node actual i fa una crida recursiva desplaçant-se cap a la dreta o l'esquerra respectivament.
La funció containsKey simplement fa la primera crida a la funció containsKeyRec i si retorna null voldrà dir que no podem trobar cap node amb aquesta clau i per tant retornem fals, en cas contrari retornarem true.

get ,getNode i getNodeRec

Tant get com getNode funcionen de la mateixa manera que containsKey, però get retornarà un objecte de tipus V i getNode un objecte de tipus Node.

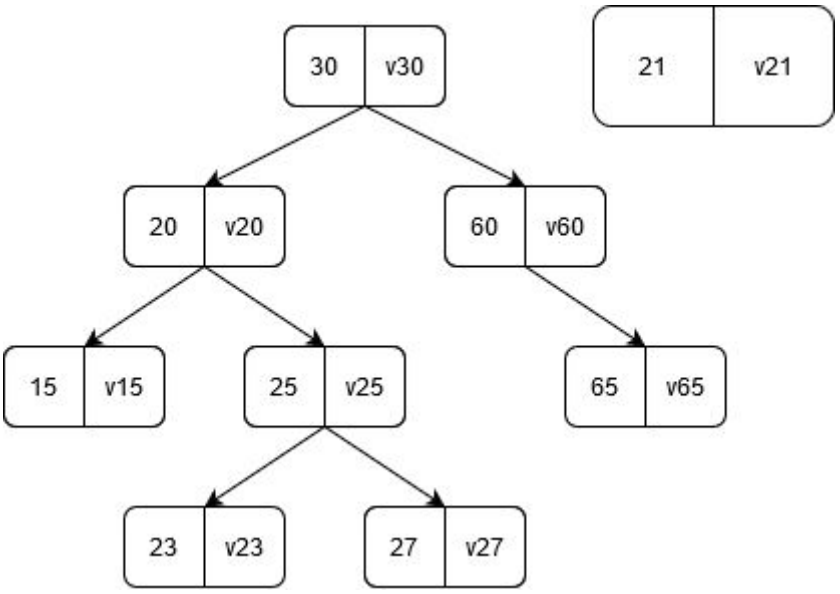
put

La funció put fa una crida a una funció recursiva anomenada putRec que conté els casos recursius, la idea és que cridarem a la funció putRec amb el node arrel de l'arbre, aquesta triarà cap a quin canto ens hem de desplaçar i anirà creant els nous nodes amb les referències actualitzades a partir del resultat de tornar a executar la funció potRec fins que arribem a la posició on s'ha d'afegir el nou valor, on crearem el nou node i retornarem la seva referència per tal que els pares corresponents adoptin les referències corresponents als seus fills i quedin modificades, un cop acabada la funció recursiva retornarem el nou arbre que tindrà com a arrel l'últim node retornat per putRec.

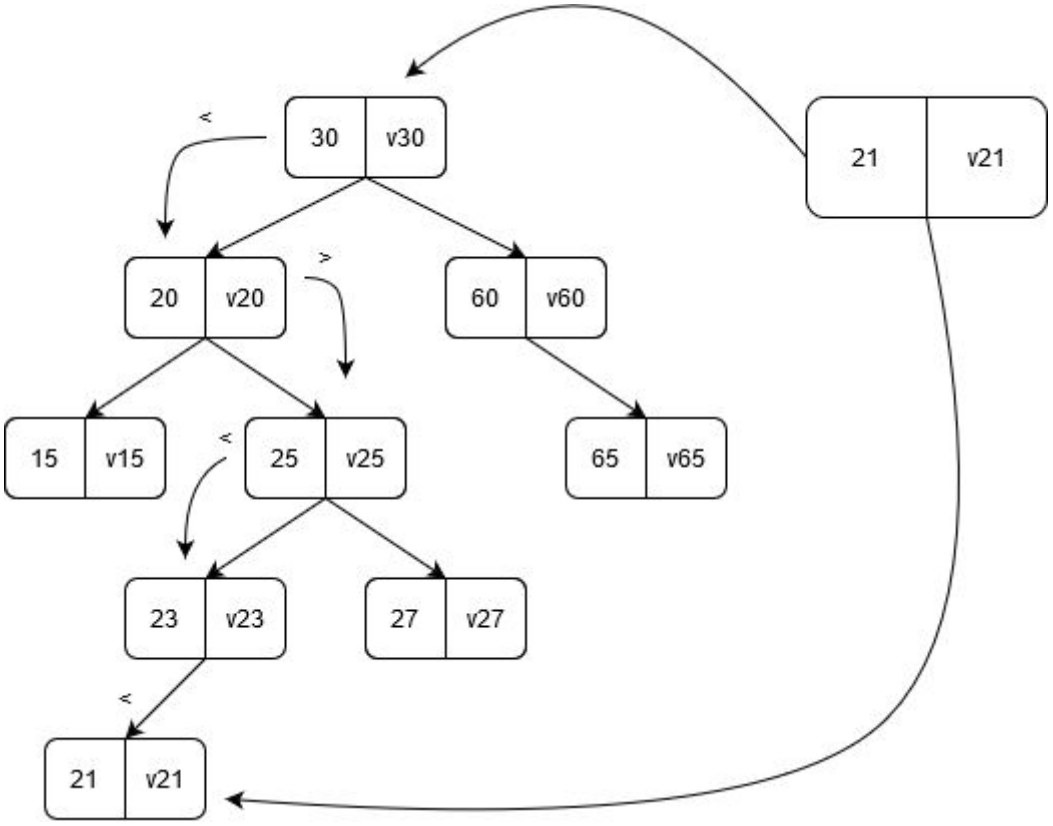
putRec

Per començar farem una comprovació per saber si el valor n és null, en aquest cas voldrà dir que ja hem arribat al final i podrem retornar el nou node.
El següent pas serà comparar l'objecte actual amb els que ja estan, si el valor de la key és més gran ens mourem cap a la dreta, en cas que sigui més petit cap a

l'esquerra, en cas que siguin iguals els dos objectes a comparar podrem actualitzar el valor, ja que significarà que el node ja existeix.



Tenim aquest arbre binary i volem afegir el 21

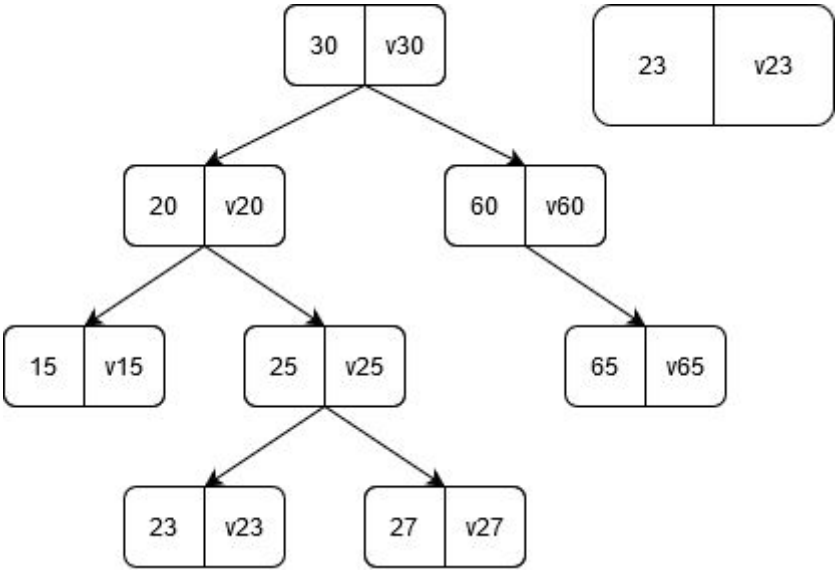


Comprovem si 21 és més gran o més petit que 30, és més petit per tant seguim comprovant amb el fill esquerre del node amb clau 30, repetim, mirem si 21 és més

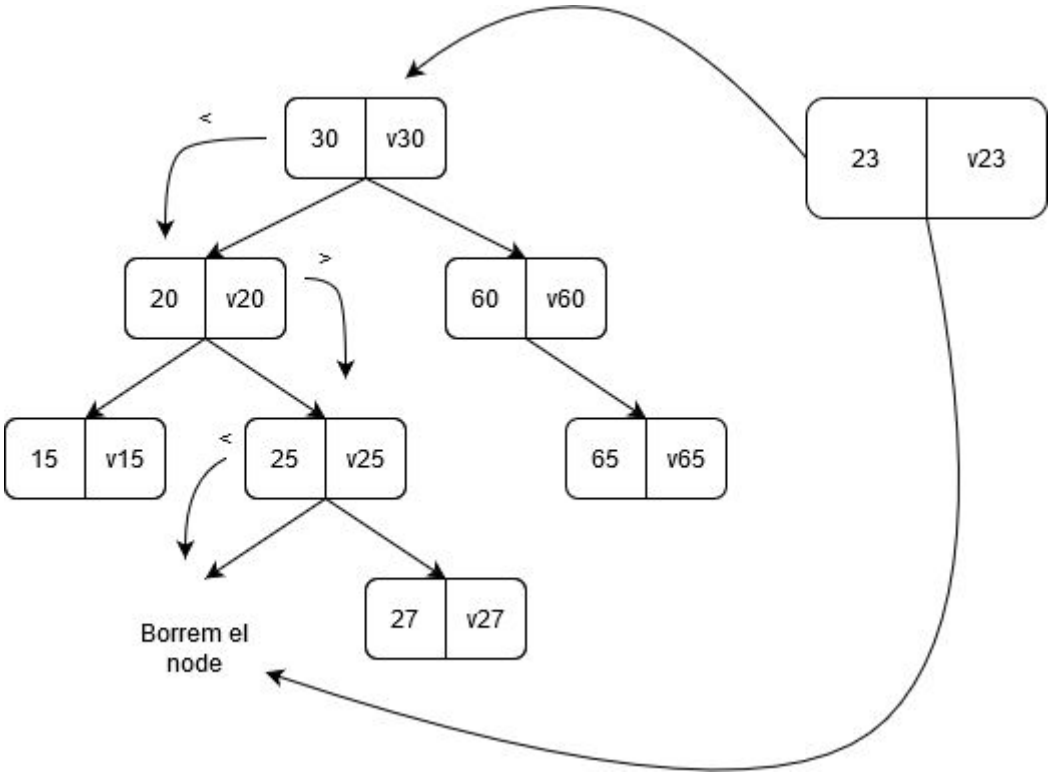
gran o més petit que 20, és més gran pel tant la següent mirarem amb el fill dret del node de clau 20.
Repetirem el procés fins que no quedin fills, pel tant haurem arribat a la posició correcta.

remRec i remove

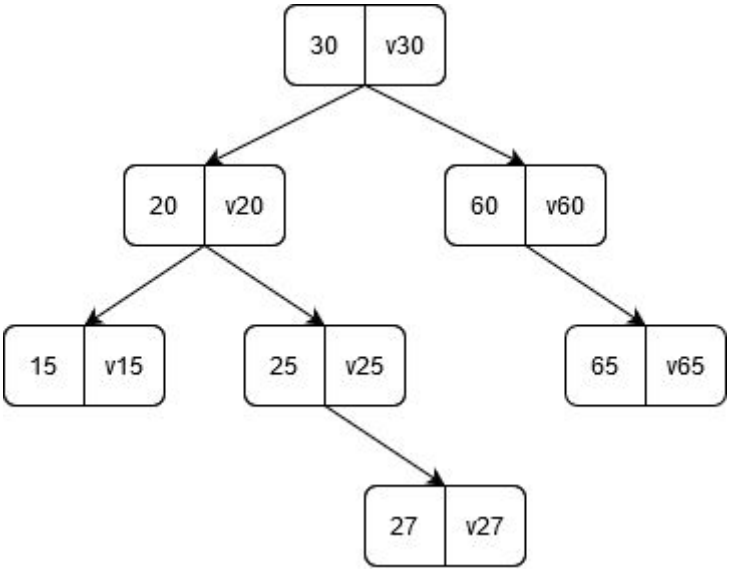
La funció remRec és una funció recursiva que mitjançant diverses crides recursives aconsegueix eliminar un node de l'arbre i reposicionar els nodes fills si és que en te, això ho aconseguim eliminant el node i després intentem reemplaçar-lo pel seu fill dret, i si no en te per l'esquerra, si té ambdós, el que no reemplaça al pare (sempre serà l'esquerre) s'afegeix a l'arbre mitjançant el mètode addNode.



En aquest gràfic veiem l'arbre sense modificar i el node que volem esborrar, el que té com a clau el nombre 23.



Compararem la clau del node que volem esborrar amb els de l'arbre binari, primer mirem que 23 és més petit que 30 així que repetim el procés pel fill esquerre, mirem que 23 és més gran 20 així que repetim el procés pel fill de la dreta, i repetim fins que ja no es compleixi cap condició, que serà el punt en el qual el node en el qual estem i el node que volem esborrar són el mateix i l'esborrem.



Finalment ens queda l'arbre sense el node.

Apartado 2: Recorridos sobre árboles binarios de búsqueda inmutables

En aquest apartat hem començat a fer la primera part, ja que era molt semblant al primer apartat, però en avançar, no hem estat capaços de passar a forma iterativa l'algoritme demanat.