Tipo arbol (ab)

constr:

arbol nulo:--->arbol

plantar: α.arbol.arbol--->arbol

proye:

hi: arbol--> arbol

hd: arbol--> arbol

raiz: arbol--> α

altura: arbol-->n

hi(anulo())--> error

hd(anulo())--> error

raiz(anulo())--> error

hi(plantar(α1,t1,t2))= t1

hd(plantar(α1,t1,t2))= t1

raiz(plantar(α1,t1,t2))= α1

altura(plantar(α,t1,t2))=1+max(altura(t1),altura(t2))

recorrido de arbol

Un árbol binario es una estructura de datos conformada de un nodo principal nombrado raíz del cual se derivan 0..2 nodos. Un árbol puede implementar distintos tipos de recorridos:

* *preOrden.*Recorre el árbol en el orden de raíz, izquierda, derecha
* *enOrden.* El orden es izquierda, raíz, derecha
* *postOrden.* Los nodos son recorridos en el orden de izquierda, derecha, raíz

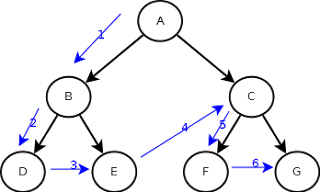


Fig. Recorrido preorden

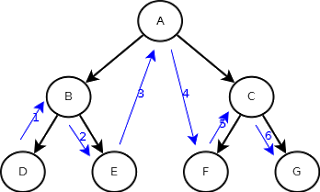


Fig. Recorrido enorden

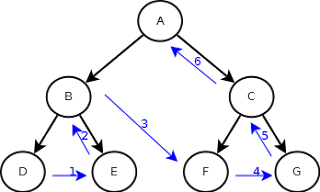


Fig. Recorrido postorden

recorrido: arbol--> lista

preorden (t)= add(raiz(t),concat(preorden(hi(t)),preorden(hd(t)))

inorden (t)= concat(inorden(hi(t)),add(raiz(t),inorden(hd(t)))

postorden(t)=concat(postorden(hi(t)),concat(postorden(hd(t)),add(raiz(t),listanva()))

abb(arbol bin de busq):

abb es un a bin de buq tal que: raiz(t)>raiz(hi(t))

raiz(t)<raiz(hd(t))

hi(t) es un abb y hd(t) es un abb

ej: tengo 7,8,6,1,10,5,18,9

7

/ \

6 8

/ \

1 10

\ / \

5 9 18

/

2

\

3

buscar: α.abb-->bool (si esta el elemento )

buscar(α1,t) t es abb(verifica si ESTA el elemento α1 a encontrar)

raiz(t)=α1 true

buscar(α1,t)=

raiz(t)>α1 buscar(α1,hi(t))

raiz(t)<α1 buscar(α1,hd(t))

t=anulo() false

recorrido inorden -->lista ordenada

1,2,3,5,7,8,9,10,18

en caso de casi ordenado x ej;1,2,3,10,11,12,4,5

se puede dar

1

\

2

\

3 y asi donde la altura es aprox a n

nivel de comparaciones → log2(n)

arbol completo: si alt(hi)=alt(hd) y es completo(hi(t)) y es completo (hd(t))

es decir se tiene 2^n-1 elementos

arbol semicompleto: completo(1) o altura(hi(t))=altura(hd(t)) +1 y el hd(t) sea completo y el hi sea semicompleto(2) o altura (hi(t))=alt(hd(t)) y hi(t) es completo y hd(t) es semicompleto (3)

completo semicompleto

. .

/ \ /

. . .

arbol equilibrado: si la |alt(hi(t))-alt(hd(t))|<=1 (el valor abs de eso es igual o menor a 1)

y equilibrado(hi(t)) y equilibrado(hd(t))