22-8-2016

Programació avançada i estructura de dades

Curs 2015/16

LS31289 Jordi Rubió Jornet

LS30652 Gabriel Cammany

WordCount

Pràctica Segon Semestre

Índex

[Disseny dels modes de recompte 2](#_Toc459801961)

[Resultats 3](#_Toc459801962)

[Conclusions 5](#_Toc459801963)

[Bibliografia 6](#_Toc459801964)

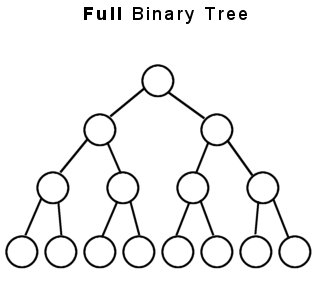
# Disseny dels modes de recompte

Exposeu el disseny de cadascun dels 4 modes de recompte que heu implementat. Concretament, dibuixeu les estructures de dades dissenyades i doneu el seu diagrama de classes. Expliqueu també el funcionament i doneu el cost computacional del procés de recompte i obtenció de les 2 modalitats de resultats per a cadascun dels 4 modes de recompte

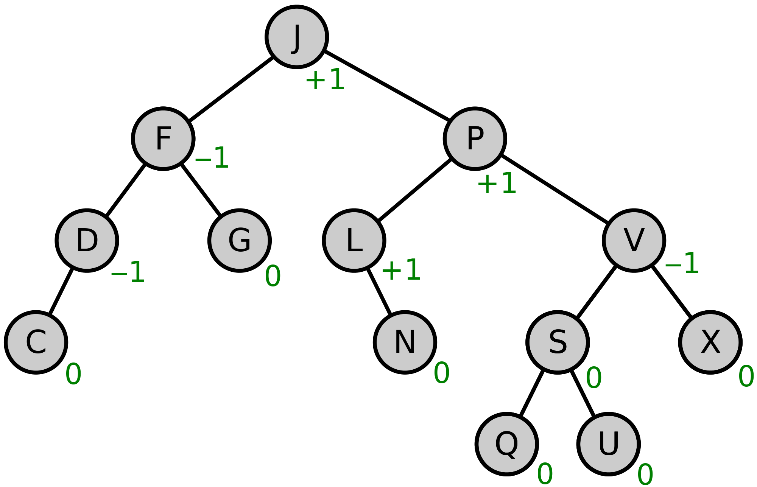
Default:

Per comparar amb els altres 4 modes de recompte, hem fet servir un Array per la busqueda.

Arbre Binary:



Arbre AVL



Taula HASH

A les taules d’adreçament obert, també anomenades hash tancat, es disposa d’un vector amb tantes posicions com valors de dispersió, el coneixem perquè al selecció del fitxer

El hash tancat ubica un parell < k , v > dins una posició de l’espai d’adreces. Si s’intenta col·locar la clau k dins la taula i la posició associada ja conté un altre parell, es produeix una col·lisió. Llavors la estratègia de redispersió genera una seqüència d’adreces alternatives (que enumerarem funció rehash() per la clau k dins la taula hash.

Un cop trobada la paraula amb la seva key, augmentem el

Resultats

Una coses que ens vam donar compte, que la funció hash tenia que ser de la major qualitat possible, si fem un promig entre els 3 documents que hem analitzat (de diferents tamanys), ens trobem que amb una funció bàsica hem tingut un promig de 1166460670 col·lisions. Això significa que hem tingut que fer 1166460670 la funció rehash, cosa que fa que incrementi el temps de execució fins a 51,6 segons de promig.

Si fem una funció senzilla, però mes lògica, com mirar el valor asci, la cosa canvia molt.

Les col·lisions baixen fins a 3593 (324557 vegades menys) y el temps es redueix a menys d’un segon.

Un cop analitzat aquest problema, vam cercar molts algoritmes hash orientats a cadenes de caràcter (y no a la seguretat) y vam trobar una llista de algoritmes, a continuació deixem la gràfica del anàlisis de les mateixes:

Com podem veure, la funció JSHash es la que millor ens funciona de manera absoluta amb uns temps de 68,4 ms y 1710 col·lisions de promig.

Adjuntarem els fitxers CSV y Excel que em obtingut en el desenvolupament de les proves.

Exposeu l’estudi i l’anàlisi que heu fet dels modes de recompte i de visualització de resultats segons les indicacions explicades abans. Cal incloure el temps d’execució i l’ús de memòria dels 36 experiments (12 per a cadascun dels 3 fitxer). Utilitzeu taules i gràfics.Dedicació

Planificació i disseny: Crear/replantejar les diferents estructures de dades que ens semblàvem mes correctes per el desenvolupament de la practica. Hi vam dedicar unes 5 hores.

Codificació de les estructures: Un cop teníem ja la idea clara, ens vam posar a picar el codi, dels quals podríem dir que vam tardar 8 hores, ja que vam haver de implementar tot el que havíem pensat, i després comprovar que fos correcte.

Experimentació: Aquesta part ha sigut la que mes ens ha costat a nivell de horari, ja que em experimentat amb varies funcions hash y en molts documents, així com combinació d’estructures. Hi vam dedicar unes 10 hores

Documentació: Finalment la creació d’aquest document i dels comentaris, ha sigut de unes 3h.

Conclusions

Expliqueu les conclusions que heu assolit amb la realització de la pràctica

Bibliografia

Indiqueu els recursos consultats. Seguiu la norma ISO 690:2010

http://griho2.udl.es/josepma/eines/hash/hash1.html