2015

Documentació de la Pràctica 4

ESTRUCTURA DE DADES

PAU SANCHEZ VALDIVIESO I ALBERT ESPÍN ROMÁN

| 14.05.2015

Índex

1. Avaluació de les implementacions d'arbres binaris i sistemes de cerca binària

En la present pràctica de l'assignatura hem implementat el tipus de dada abstracta arbre binari, *BTree*, aquella estructura formada per nodes connectats amb la particularitat que el màxim nombre de fills de cada node és dos, és a dir que es tracta d'un arbre de grau 2. Fent servir aquesta estructura hem implementat un cercador de paraules, la classe *BTreeWordFinder*, capaç de fer una mostra de totes les paraules d'un arxiu per ordre alfabètic tot indicant les aparicions al text (amb línia i posició en aquesta), a més de mostrar quines de les paraules d'un diccionari es troben al document esmentat tot indicant, de nou, les seves aparicions al text.

La implementació esmentada al paràgraf anterior, però, presenta un defecte d'optimització de cara a la cerca: la distribució dels nodes que representen les paraules del text arreu de l'arbre és regida només per un criteri de valor ASCII —donada una paraula en un node de l'arbre, les paraules amb lletres alfabèticament anteriors es col·loquen a l'esquerra d'aquesta, les alfabèticament posteriors a la dreta; per a la resta de caràcters ocorre el mateix cas general: major valor ASCII implica que la paraula es posicioni a l'esquerra, menor valor condueix cap a la dreta—; així, l'arbre pot quedar desnivellat, amb nodes que no són fulla sense fills o amb només un (imaginem, per exemple, el cas que la paraula "a" sigui l'arrel del node; totes les cadenes de caràcters diferents que comencin per lletres quedaran a la seva esquerra).

Si l'arbre binari, com al cas analitzat, no compleix que els nodes de tots els nivells menys l'últim tinguin cadascun dos nodes fills, en pocs casos particulars resultarà possible que la cerca d'un element en l'arbre es pugui produir amb complexitat O($\log_2 n$), on n és el nombre de nodes, ja que per definició això requereix arbres amb nodes de dos fills excepte als nivells de fulla, o, si es permet aproximar, almenys un cas molt proper a aquest.

L'arbre balancejat, implementat a la classe *BalancedBTree*, controla el procés d'inserció de nodes a l'arbre de tal manera que regula la distribució al menor desnivell per evitar que quedin buits —nodes sense dos fills a la part interior de l'arbre— i per assolir la màxima simetria tant en l'arbre complet com en el major nombre possible de subarbres analitzables a l'arbre global.

El cercador de paraules *BalancedBTreeWordFinder*, que fa servir un arbre binari balancejat, aconsegueix dur a terme una cerca binària òptima dins de l'arbre, on s'assoleix efectivament una complexitat de cerca O($\log_2 n$): podem observar la Figura 2 i la Figura 4 ubicades als Annexos i comprovar com efectivament això es compleix. Així, el fitxer *smallText* compta amb 199 paraules i 8 nivells en l'arbre balancejat, i tenim que $\log_2 199$=7'636624621 (que aproximat a l'alça és precisament 8); cal fer un màxim de 8 passos (en ocasions 7, podem concloure donat el nombre decimal), baixant nivells i comparant nodes, per comprovar si un valor està als nodes de l'arbre i on o no hi és, efectivament la complexitat és O($\log_2 n$). També es verifica aquest punt per al cas del *largeText*, que compta amb 10117 paraules i 14 nivells: $\log_2 10117$=13.30449393, que

s'aproxima a l'alça a 14, els nivells de l'arbre; de nou, la complexitat O($\log_2 n$) resulta evident.

Analitzant les diverses figures annexades, podem veure com el temps d'inserció donat un dels dos textos és significativament menor al temps de cerca amb el diccionari independentment del tipus d'arbre, això s'explica fàcilment: l'operació de cercar una vegada un element té una complexitat similar a la de fer una inserció —en ambdós casos es recorre l'arbre en cerca binària, tot i que en un cas es pugui inserir i a l'altre no—, i donat que en aquesta pràctica al cas d'inserir es treballa amb un text de moltes menys paraules que quan se cerca (*dictionary* conté centenars de milers de paraules; *largeText*, major que *smallText*, tan sols una desena de milers).

Podem veure, també, que el temps de construcció de l'estructura de l'arbre és superior, però no en gran magnitud, al cas de l'arbre binari balancejat; això s'explica perquè afegeix a les operacions involucrades ja en la inserció de l'arbre binari simple aquelles dedicades a comprovar si cal balancejar (per tant les implicades en comprovar el factor de balanç de diferents nodes i per extensió la profunditat dels seus descendents) i a realitzar les rotacions. Cal dir, però, que el fet que l'arbre estigui balancejat comporta que hi hagi menys nivells de profunditat a l'estructura, i en conseqüència redueix el nombre de passos previs a comprovar si cal balancejar, si bé aquestes comprovacions posteriors, porten, com hem indicat, a un augment del temps total de construcció de l'estructura.

Al cas de la cerca d'una paraula als nodes de l'arbre, però, hi ha un moderat guany de velocitat en l'arbre balancejat, justament perquè es redueixen els nivells en anul·lar la presència de nodes interns sense dos fills i amb espais buits, de manera que els passos per a fer una cerca es redueixen, ja que no cal baixar tant a través de l'estructura simplement perquè verticalment, per cerca binària, s'acaba de recórrer més ràpidament que si l'arbre no fos balancejat.

Com a conclusió podem indicar que l'arbre binari balancejat és menys òptim en inserció però més per a fer cerques que l'arbre binari simple, i, com la magnitud del que es guanya per a cercar és propera a la d'allò que es perd per a construir l'estructura, considerem que als contexts on realment és útil emprar un arbre binari balancejat per sobre de fer servir un arbre binari simple és en situacions en què s'insereixi —i, si és possible fer-ho, s'elimini— molt poc freqüentment sobre l'arbre, de manera que mantingui força constant la seva estructura, però que el volum de cerques que es produeixin sobre l'arbre sigui elevat, de manera que es compensi el temps addicional d'inserció respecte l'arbre binari simple amb un guany per cercar aprofitat en cerques contínues o almenys amb una certa freqüència; per arbres que fossin construïts per a acollir una única o molt poques cerques podria ser més eficient emprar arbres binaris que no es balancegessin.

2. Annexos

Temps de construcció de l'arbre (inserció de totes les paraules)	smallText	largeText
BTree	6.42390379846 ms	446.450143065 ms
BalancedBTree	9.26845631128 ms	537.267609803 ms

Figura 1

Temps de cerca de l'arbre	smallText	largeText
BTree	3173.88013781 ms	5876.53363644 ms
BalancedBTree	2498.44770656 ms	4185.30830673 ms

Figura 2

Profunditat de l'arbre	smallText	largeText
BTree	16 nivells	36 nivells
BalancedBTree	8 nivells	14 nivells

Figura 3

Text	Nombre de paraules
smallText	199
largeText	10117

Figura 4