Analys datatyper LinkedList, BST, AVL och Splay-träd

Grupp 10 Sebastian Ljunggren, Max Witt

February 11, 2012

Jämför vi AVL och Splay, vilka båda genomför någon form av optimering så kan vi se att Splayträdet gör bäst ifrån sig när exakt samma element hämtas flera gånger (exempelvis text 2 och 4), eller när datan läggs till på ett sådant sätt att det egentligen blir en länkad lista (varje nod får bara ett barn) och elementen anropas i turordning (text 3). Text 4 och 5 innehåller i princip samma data, men sorterad på två olika sätt. Då samma element återkommer flera gånger på rad i text 4 kommer splay-funktionen göra skillnad en gång per unikt ord. I text 5 däremot splayas trädet verkligen vid varje ord (även upprepningarna). Teoretiskt borde skillnaden vara n mot 5n splay-rotationer mellan text 4 och 5.

AVL gör också ganska bra ifrån sig tack vare att trädet fylls upp och balanseras. Även om AVL gör fler jämförelser än Splay i de flesta fall så tar Splay proportionellt längre tid på sig. I detta läget tar Splay ungefär hälften av tiden för AVL, men gör bara 10% av jämförelserna om man tittar på text 3 exempelvis. Hade compareTo() tagit längre tid hade vi märkt en signifikant skillnad mellan AVL och Splay, men i nuläget är själva splay-funktionen lika dyr som de extra jämförelserna.

Det är tydligt i vår mätdata att BSTs stora svaghet är försorterad indata. Första elementet som läggs till förblir rotelement även om det egentligen skulle varit ett löv längst till vänster i det optimerade fallet. Kort och gott blir BST en länkad lista för text 3, 4 och 5 men med tidsmässigt dyrare metoder än den simpla länkade listan.

Text	1	2	3	4	5
SLC	26	6	277	21	19
BST	3	2	310	23	22
AVL	3	2	5	2	3
Splay	3	2	3	1	3

Table 1: Tidsåtgång. Värden i ms

Text	1	2	3	4	5
SLC	2,21 M	492 k	17,8 M	1,65 M	1,66 M
BST	74,9 k	46,0 k	17,8 M	1,65 M	1,66 M
AVL	52,6 k	31,2 k	112 k	41,0 k	41,1 k
Splay	47,1 k	13,6 k	13,8 k	5,49 k	17,5 k

Table 2: Antal jämförelser