1 Contents of testfile4.txt

zzzzz zzzzy zzzzx zzzzw zzzzv zzzzu zzzzt zzzzs zzzzr zzzzq zzzzp zzzzo zzzzn zzzzm zzzzl zzzzk zzzzj zzzzi zzzzh zzzzg zzzzf zzzze zzzzd yyyyz yyyyy yyyyx yyyyw yyyyv yyyyu yyyyt yyyys yyyyr yyyyq yyyyp yyyyo yyyyn yyyym yyyyl yyyyk yyyyj yyyyi yyyyh yyyyg yyyyf yyyye yyyyd xxxxz xxxxy xxxxx xxxxw xxxxv xxxxu xxxxt xxxxs xxxxr xxxxq xxxxp xxxxo xxxxn xxxxm xxxxl xxxxk xxxxj xxxxi xxxxh xxxxg xxxxf xxxxe xxxxd wwwwz wwwwy wwwwx wwwww wwwwv wwwwu wwwwt wwwws wwwwr wwwwq wwwwp wwwwo wwwwn wwwwm wwwwl wwwwk wwwwj wwwwi wwwwh wwwwg wwwwf wwwwe wwwwd vvvvz vvvvy vvvvx vvvvw vvvvv vvvvu vvvvt vvvvs vvvvr vvvvq vvvvp vvvvo vvvvn vvvvm vvvvl vvvvk vvvvj vvvvi vvvvh vvvvg vvvvf vvvve vvvvd uuuuz uuuuy uuuux uuuuw uuuuv uuuuu uuuut uuuus uuuur uuuuq uuuup uuuuo uuuun uuuum uuuul uuuuk uuuuj uuuui uuuuh uuuug uuuuf uuuue uuuud ttttz tttty ttttx ttttw ttttv ttttu ttttt tttts ttttr ttttq ttttp tttto ttttn ttttm ttttl ttttk ttttj tttti tttth ttttg ttttf tttte ttttd ssssz ssssy ssssx ssssw ssssv ssssu sssst sssss ssssr ssssq ssssp sssso ssssn ssssm ssssl ssssk ssssj ssssi ssssh ssssg ssssf sssse ssssd rrrrz rrrry rrrrx rrrrw rrrrv rrrru rrrrt rrrrs rrrrr rrrrq rrrrp rrrro rrrrn rrrrm rrrrl rrrrk rrrrj rrrri rrrrh rrrrg rrrrf rrrre rrrrd qqqqz qqqqy qqqqx qqqqw qqqqv qqqqu qqqqt qqqqs qqqqr qqqqq qqqqp qqqqo qqqqn qqqqm qqqql qqqqk qqqqj qqqqi qqqqh qqqqg qqqqf qqqqe qqqqd ppppz ppppy ppppx ppppw ppppv ppppu ppppt pppps ppppr ppppq ppppp ppppo ppppn ppppm ppppl ppppk ppppj ppppi pppph ppppg ppppf ppppe ppppd ooooz ooooy oooox oooow oooov oooou oooot oooos oooor ooooq oooop ooooo ooooon ooooom ooool ooook ooooj ooooi ooooh ooooog ooooof ooooe ooood nnnnz nnnny nnnnx nnnnw nnnnv nnnnu nnnnt nnnns nnnnr nnnnq nnnnp nnnno nnnnn nnnnm nnnnl nnnnk nnnnj nnnni nnnnh nnnng nnnnf nnnne nnnnd mmmmz mm-mmy mmmmx mmmmw mmmmv mmmmu mmmmt mmmms mm-mmr mmmmq mmmmp mmmmo mmmmn mmmmm mmmml mm-mmk mmmmj mmmmi mmmmh mmmmg mmmmf mmmme mm-mmd llllz lllly llllx llllw llllv llllu llllt lllls llllr llllq llllp llllo lllln llllm lllll llllk llllj lllli llllh llllg llllf lllle lllld kkkkz kkkky kkkkx kkkkw kkkkv kkkku kkkkt kkkks kkkkr kkkkq kkkkp kkkko kkkkn kkkkm kkkkl kkkkk kkkkj kkkki kkkkh kkkkg kkkkf kkkke kkkkd jjjjz jjjjy jjjjx jjjjw jjjjv jjjju jjjjt jjjjs jjjjr jjjjq jjjjp jjjjo jjjjn jjjjm jjjjl jjjjk jjjjj jjjji jjjjh jjjjg jjjjf jjjje jjjjd iiiiz iiiiy iiiix iiiiw iiiiv iiiiu iiiit iiiis iiiir iiiiq iiiip iiiio iiiin iiiim iiiil iiiik iiiij iiiii iiiih iiiig iiiif iiiie iiiid hhhhz hhhhy hhhhx hhhhw hhhhv hhhhu hhhht hhhhs hhhhr hhhhq hh-hhp hhhho hhhhn hhhhm hhhhl hhhhk hhhhj hhhhi hhhhh hhhhg hhhhf hhhhe hhhhd ggggz ggggy ggggx ggggw ggggv ggggu ggggt gggggs ggggr ggggq ggggp ggggo ggggn ggggm ggggl ggggk ggggj ggggi ggggh ggggg ggggf gggge ggggd ffffz ffffy ffffx ffffw ffffv ffffu fffft ffffs

ffffr ffffq ffffp ffffo ffffn ffffm ffffl ffffk ffffj ffffi ffffh ffffg fffff ffffe ffffd eeeez eeeey eeeex eeeew eeeev eeeeu eeeet eeees eeeer eeeeq eeeep eeeeo eeeen eeeem eeeel eeeek eeeej eeeei eeeeh eeeeg eeeef eeeee eeeed ddddz ddddy ddddx ddddw ddddv ddddu ddddt dddds ddddr ddddq ddddp ddddo ddddn ddddm ddddl ddddk ddddj ddddi ddddh ddddg ddddf dddde ddddd ant

2 Why it works

This testfile works because its contents are in reverse alphabetic order. This means that the element being inserted into the search tree is always lexicographically prior to previous elements and is inserted to the left of all existing nodes. In the unbalanced BST this results in a tree that is essentially equivalent to a singly linked list. Searching for the word "ant" demonstrates the difference in efficiency for worst case scenarios for a BST compared to an AVL Tree.

3 Numerical results

testfile1.txt - general statistics

Structure	Total Nodes	Average Node Depth	Single Rotations	Double Rotations
AVL	19	2.26316	6	2
BST	19	3.15789	n/a	n/a

testfile1.txt - search results

	AVL			BST		
Word	Left Links Followed	Right Links Followed	Total Links Followed	Left Links Followed	Right Links Followed	Total Links Followed
me	3	1	4	4	2	6
kind	1	0	1	3	1	4
problems	1	1	2	2	1	3
created	2	1	3	4	1	5

testfile2.txt - general statistics

Structure	Total Nodes	Average Node Depth	Single Rotations	Double Rotations
AVL	16	2.5	9	2
BST	16	6.0625	n/a	n/a

testfile2.txt - search results

Word	AVL			BST		
	Left Links Followed	Right Links Followed	Total Links Followed	Left Links Followed	Right Links Followed	Total Links Followed
caught	2	1	3	0	2	2
flying	1	1	2	0	5	5
flown	2	2	4	1	5	6
mauve	0	1	3	0	9	9

testfile3.txt - general statistics

Structure	Total Nodes	Average Node Depth	Single Rotations	Double Rotations
AVL	13	2.23077	5	0
BST	13	3.23077	n/a	n/a

testfile3.txt - search results

Word	AVL			BST		
	Left Links Followed	Right Links Followed	Total Links Followed	Left Links Followed	Right Links Followed	Total Links Followed
misty	2	1	3	3	2	5
cobwebs	1	0	1	1	0	0
clockwork	2	0	2	2	0	2
landscape	1	1	2	2	1	3

testfile4.txt - general statistics

Structure	Total Nodes	Average Node Depth	Single Rotations	Double Rotations
AVL	530	7.08868	520	2
BST	530	264.5	n/a	n/a

testfile4.txt - search results

Word	AVL			BST		
	Left Links Followed	Right Links Followed	Total Links Followed	Left Links Followed	Right Links Followed	Total Links Followed
ggggs	6	2	8	444	0	444
nnnni	3	4	7	293	0	293
uuuuo	6	2	8	126	0	126
ant	9	0	9	529	0	3

4 When are AVL trees preferable?

AVL trees are the better option when operations need to be reliably fast. This is not to imply that AVL trees are always faster than BSTs, but that they perform better in the worst case. In fact, due to the overhead created by rebalancing rotations in AVL trees, a BST will often be faster for a particular operation. Thus they are ideal when it is not critical that an operation is as fast as possible, but it is very important that any given operation will never take significantly longer than average. An example of when these types of goals might exist is when creating a user interface for an application. A page load or button press that takes much longer than normal would have a severe negative impact on the user experience.

Certain characteristics of the data being operated on and what is being used for can also influence the choice between BSTs and AVL trees. If the data is inserted into the tree in sorted order, a BST will simply mimic a linked list, provided none of the benefits of the tree data structure. If it is known that elements of the tree will be accessed with some uniformity then the reduced average path length of the AVL tree may be more important than if only a few key elements are being accessed repeatedly and can be stored near the root of the tree. Besides performance, the cost of implementation as discussed in the next section is also something to consider when deciding between the two tree types.

5 AVL implementation costs

Although they are generally faster, AVL trees are also much more difficult to implement. The addition of automatic balancing adds complexity to the code, including handling multiple different cases for inserting and removing nodes.

Additionally, to allow balancing to be done with relative speed, nodes must store additional information about their height in the tree. This means that AVL trees consume more memory than BSTs, an amount that grows in direct correspondance to the size of the tree.