MA U34605 Quiz 02 w/e 23/10/20

Answer any 3 questions. Submit them using the `submit-work` program as pdfs, either handwritten and scanned, or typeset. They should be submitted before 1pm on Tuesday 27 October. All questions carry 20 marks.

(1) As in the first quiz, there is some ambiguity in the word 'sorted.' How does this impact on the use of search trees? There are reasons to consider this rather different from the earlier question about binary search.

Answer

The difference is between 'nondecreasing' and 'strictly increasing.' Usually the 'keys' in binary search trees are connected with other information, so it is usual that different nodes have different keys. However, searching on a key would still work in that if a key is stored in the tree then a node is returned with that key, but which node is returned is a matter of chance.

---

(2) Sorting usually allows for the same key (or item) to have repetitions in the input, and in the output. A method is *stable* if, when a key is repeated in the output, it is repeated in the same order as in the input.

Is mergesort, defined by the code presented in the notes, stable? Give reasons.

Answer

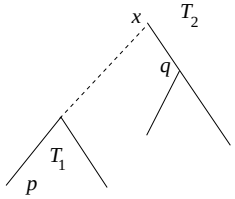
Yes. It depends on the way merge is written and used. The way it is used, `x[]` and `y` are part of the same array, and `x` comes before `y`.

The condition `threeway(x[i],y[j])>0` means that `x[i]` is strictly greater than `y[j]`, and `y[j]` is 'taken.' Otherwise `x[i] <= y[j]` and `x[i]` is 'taken.' This means that relative order is preserved where the values are equal.

---

(3) (Splay trees). Let T_1 be a tree with p nodes, and T_2 a tree with q nodes, with root x, where x has no left child. Join the trees by making T_1 the left subtree of x. Calculate the change in potential.

Answer



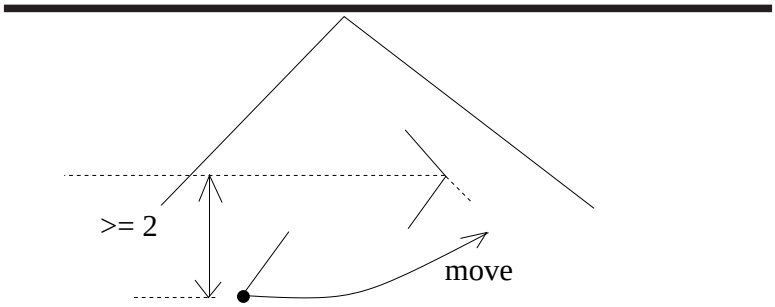
The only node whose rank is changed after the change is x. Before the change it was $\log_2(q/n)$. After, it is $\log_2((p+q)/n)$. The change in potential is $\log_2((p+q)/q)$.

(4) Here is one way — there might be better ways — to show that the IPL of a binary tree is $\Omega(n \log n)$. Note: n is always the number of nodes in the tree.

It is enough to assume that there exists a natural number d such that (i) all leaves are at depths d or $d+1$, and (ii) there are 2^d nodes at level d.

Prove that if T is any binary tree in which there is a node u with fewer than 2 children, and there is a leaf v such that depth(v) - depth(u) > 1, then the tree can be altered, while retaining the number of nodes, to get another tree with smaller IPL.

Answer



The diagram shows what to do. A leaf node is illstrated, and a node with 0 or 1 children whose depth is at least 2 less. The leaf can be moved, reducing its depth by at least 1, and leaving the depths of other nodes unchanged.

Keep doing this as long as possible. Suppose that $d+1$ is the maximum depth of all nodes in the tree. Every node at depth $\leq d-1$ has two children, since one cannot continue moving nodes as discussed. There are $2^{d+1}-1$ nodes at depth $\leq d$. There are at most 2^{d+1} at depth $d+1$.

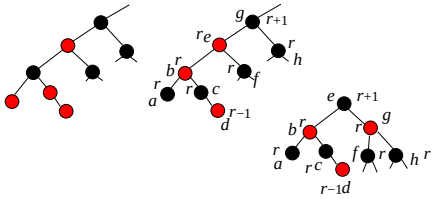
A crude lower bound on the IPL is $d \times 2^d$. Now $2^{d+2}-1 \geq n$, so $2^d \geq (1+\log_2 n)/4$ and $d \geq \log_2(n+1)-2$, so the ipl is at least

$$\left(\frac{1+\log_2 n}{4}\right)(\log_2(n+1)-2)$$

which is $\Omega(n \log n)$.

(5) Fix the double red illustrated, assuming that the whole tree fulfils the red-black conditions everywhere else.

In your answer you should label the nodes $a \ldots h$ (in inorder), and show their ranks, given that the lowest red node has rank $r-1$.


g
r+1
r e
r h
b r
r f
r a
r c
d r−1
e
r+1
b r
r g
r a
r c
f r
r h
r−1 d