PS3

June 18, 2025

Problem 3-1

(a)

The requirement of the data structure is:
- insertion in sublinear time
- finding minimum and maximum in sublinear time

The data structure which fits these requirements are the AVL tree, which allows insertion, finding the minimum item and maximum item in $O(\log(N))$, where N is the number of items.

(b)

It will take $O(\log(N))$ time to insert a key in the range index.

(c)

It will take $O(\log(N))$ time find the minimum key in the range index. (You keep going from the root to the leaves in the left child nodes until you reach a node which doesn't have a left child node).

(d)

It will take $O(\log(N))$ time find the maximum key in the range index. (You keep going from the root to the leaves in the right child nodes until you reach a node which doesn't have a right child node).

(e)

Rank(l) counts all the keys for which $x \leq l$. The number of nodes between (including) l and h when both l and h exist as an index

$$\text{Count}(l, h) = \text{Rank}(h) - \text{Rank}(l) + 1$$

Since rank doesn't count the node itself but when evaluating $\text{Rank}(h) - \text{Rank}(l)$ it does count the the node with the key l but not the node with key h.

(f)

If doesn't l doesnt't exist in the index but h does, the answer changes to:

$$\text{Count}\,(l,h) = \text{Rank}\,(h) - \text{Rank}\,(l)$$

(g)

If l exists in the index but h doesn't, we have:

$$\text{Count}\,(l,h) = \text{Rank}\,(h) - \text{Rank}\,(l) + 1$$

(h)

If neither l or h exist in the index we have:

$$\text{Count}\,(l,h) = \text{Rank}\,(h) - \text{Rank}\,(l)$$

(i)

In order to respond to Rank () queries in sublinear time, we augment an extra field to each node $node.\gamma$ which means the number of nodes in the subtree rooted at $node$.

(j)

The aumentation requires $O\,(\log\,(N))$ bits of storage per node, since it requires $\log\,(N)$ bits to encode the possible values of Rank () numbers $1, 2, \ldots, N$.

(k)

$N_4.\gamma = 1$ since N_4 is a leaf, the subtree rooted at N_4 has a single node.

(l)

$N_3.\gamma = 3$

(m)

$N_2.\gamma = 6$

(n)

$N_1.\gamma = 10$

(o)

The functions which need to be modified are only: ROTATE-LEFT, ROTATE-RIGHT and REBALANCE.

INSERT AND DELETE apply rotate left and right which already will modify $node.\gamma$, and we are using a AVL tree so there is no method HEAPIFY for this data structure.

(p)

The running time of Rank () is $O(\log(N))$, using the γ atribute. As a result, Count () is also $O(\log(N))$.

(q)

LCA most likely means: lowest common ancestor. LCA pseudo code return the node closest to the root which is between l and h.

(r)

The running time of LCA (l, h) is $O(\log(N))$ as it basically performs a (conditional) search going from the root down to the leafs.

(s)

The running time is $O(\log(N) + O(L))$, since what Node-List () does is either appends a key to the list or ignores a branch of the tree, while advancing from the lowest common ansector of l and h down to the leas. The number of branches ignored scale as the height of the tree and the number of keys added equals the list size, L.

(t)

Since the running times of LCA and NODE-LIST are $O(\log(N))$ and $O(\log(N) + O(L))$ the total running time of LIST is $O(\log(N) + O(L))$.

Problem 3-2

(a)

Running circuit2.py with tests/10grid_s.in as input. The method with the highest CPU usage is: "intersects"

(b)

"intersects" method is called 187509314 times.

(c)

The x-coordinates of points of interest in the input are both end points of horizontal and vertical wires. Crossing points are not points of interest but the output of the method.

(d)

When the sweep line hits the x-coordinate of the left endpoint of a horizontal wire it adds the wire to the range index.

(e)

When the sweep line hits the x coordinate of the right endpoint of a horizontal wire, the wire is removed from the range index.

(f)

When the sweep line hits the mid-point of a horizontal wire nothing happens.

(g)

When the sweep line hits the x coordinate of the vertical wire a range index query is performed.

(h)

The good invariant for sweep-line algorithm holds all the horizontal wires stabbed by the sweep line

(i)

When a wire is added to the range index, the corresponding key is the y coordinate of the wire's midpoint.

(j)

The method with the highest CPU usage is "count"

(k)

"count" is called $20,000$ times.

(l)

The Python codes appear in the files AVL_index_range.py and circuit2.py.