Sean Blanchard
Comp 354

Problem Set 3-2

1a) data structure used for the range index?
AVL Trees - This will allow you to find the minimum and maximum quickly as it is a self-balancing binary search tree.

1b) How much time will it take to insert a key in the range index?

The time required is O(log n) for lookup, plus a maximum of O(log n) retracting levels on the way back to the root, so the operation can be completed in O(log n) time.

1c) How much time will it take to find the minimum key in the range index?

O(log N) - NOTES

1d) How much time will it take to find the maximum key in the range index.

O(log N) - AVL Tree

1e) Assuming $l < h$, and both l and h exist in the index, COUNT(l, h) is

The key is to count all the keys. ~~If~~ both l and h exist then $l \leq x \leq h$.
if we assume $l < h$, and h indicates all key $x \leq h$, and l ~~[illegible]~~ indicates all key $x \leq l$

This shows that ~~it must~~ be $h - l$,
hence ~~RANK(h) - (RANK(l) - 1)~~
$=$ RANK(h) - RANK(l) + 1

1f) Assume $l < h$, h exist, l DNE, COUNT(l, h)
It is like what is stated above, but since l is not included, thus $x < l$ instead of $x \leq l$.

HENCE:
RANK(h) - RANK(l)

1g) Assume $l < h$, l exist, h DNE, COUNT(l, h)
RANK(h) - RANK(l) + 1

1h) Assume $l < h$, l, h DNE, COUNT(l, h)
RANK(h) - RANK(l)

1i) The meaning of node.y is
4. The number of nodes in the subtree rooted @ node.

1j) How many extra bits of ~~storage~~ per node does the augmentation above require?

Since the class supports optimized RANK() out of box, To store numbers or nodes you need Θ(log N)-bits.

1k) $N_{4.7}$

1 since it is at the lowest part of ~~the~~ tree

1l) $N_3.y$

3 * $1 + N_4 + N_5 = 3$ Whereas $N_7 = 1$

1m) $N_2.y$

6 $1 + N_4 + N_5 + N_3 + N_7 + N_6 = 6$

1n) $N_1.y$ is

10 $1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 + N_9 + N_{10} = 10$

1o) Which functions need to be modified to update y?

Rebalance; The node has to be updated when Rotate left, searching the height of the tree, Rotate right; thus these 3 need to be modified

1p) What is the run time of COUNT() based on RANK().

$\Theta(\log N)$

1q) LCA most likely stand for

Lowest common ancestor of a Binary tree "The lowest common ancestor is defined between two nodes, p & q, as the lowest node in T that has both p & q as descendants.

1r) The running time of LCA(l,in) for trees used by the range index is

I thought the answer would be $\Theta(n)$ because you have to maybe visit every node in the tree as worst case, but don't see that as answer so it is probably $\Theta(\log N)$.

1s) Assuming that ADD-KEY runs in O(1) time, and that LIST returns a list of L keys, the running time of the NODE-LIST call @ line 3 of LIST is.

List returns a list of L keys, there are 3 checks, checking left, right and middle of subtree.

→ O(L) since we know it must return a list of keys, and since the complexity of the LCA is log(log N), ~~we will have the~~ same complexity, thus $O(L) + O(\log N)$

1t) the running time of LIST.

Same as above, we have to traverse the tree @ O(log N) time and return list of keys $O(L) = O(L) + O(\log N)$

1w) Prove that LCA is correct.
To check if LCA(l, h) returns the least-common ancestors, we must "run" the program to test. It needs to return the value of the smallest sub-tree. It shows us that both the l and h will be greater than the node returned by the LCA.

Problem 3-2-2

2a) What is the name of the method w/ the highest CPU usage?

insects has the highest CPU usage

2b) How many times is the method called?

187,590,314 times

2c) The x coordinates of points of interest in the input are (T/F)

1. the x coordinates of the left endpoints of horizontal wires

   True

2. the x coordinates of the right endpoints of horizontal wires.

   True

3. the x coordinates of midpoints of horizontal wires.

   False

4. the x coordinates where horizontal wires cross vertical wires

   False

5. the x coordinates of vertical wires

   False

2d) When the sweep line hits the x coordinate of the left endpoint of a horizontal wire.

Adds to the range index

2e) When the sweep line hits the x coordinate of the right endpoint of a horizontal line

Removed from the range of index

2f) When the sweep point hits the x coordinate of the midpoint of a horizontal wire

NOTHING HAPPENS - Algorithm has no check for midpoints

2g) When the sweep line hits the x coordinate of a vertical wire

Range Indexing

2h) What is a good invariant for the sweep-line algorithm

The range Index holds all the horizontal wires to the right of the sweep line

2i) When a wire is added to the range index, what is the corresponding key?

It has to be the y coordinate of the wire's midpoint, thus would return the true midpoint.