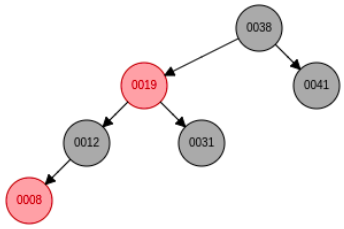
Assignment 4

Jixuan Ruan PB20000188

October 14, 2022

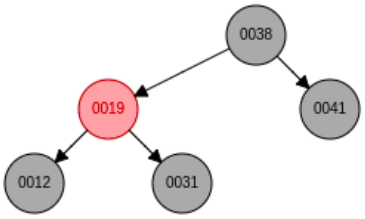
1. Assumption: Let's consider the condition when y.key is neither the smallest key greater than x.key nor the greatest key smaller than x.key.
   1) If the x is the left child of y, then $x.key < y.key$. Since y.key is not the smallest key greater than x.key, we can expect that there a node(here we assume it to be z) whose key is greater than x.key and smaller than y.key at the same time. Because $x.key < z.key$, x is in the left subtree of z or z is in the right subtree of x. We can see that x is a leaf, so x has no subtree. Meanwhile, since $y.key > z.key$, z is in the left subtree of y or y is in the right subtree of z. Since the x is the left child of y, then the conflict shows.
   2) If the x is the right child of y, then $x.key > y.key$. Since y.key is not the greatest key smaller than x.key, we can expect that there a node(here we assume it to be z) whose key is smaller than x.key and greater than y.key at the same time. Because $x.key > z.key$, x is in the right subtree of z or z is in the left subtree of x. We can see that x is a leaf, so x has no subtree. Meanwhile, since $y.key < z.key$, z is in the right subtree of y or y is in the left subtree of x. Since the x is the left child of y, then the conflict shows.
   So the previous assumption is wrong in both circumstances.
   Consequently, y.key is either the smallest key greater than x.key or the greatest key smaller than x.key.

2. (a) The picture below shows the final black-red tree.

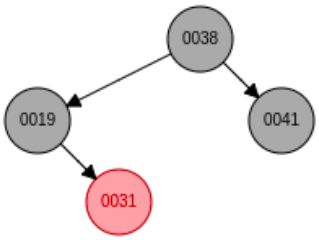


(a) red-black tree

(b) The pictures beside show the red-black trees after deleting 8 12 19 in order.



(b) delete 8



(c) delete 12

(d) delete 19

3. (a) We can see that the biggest overlap points are actually in intervals whose boundary are marked by the endpoint of several intervals participating in the tree generation or just the endpoints of some intervals, so there must exist an endpoint in these biggest overlap points.

   (b) Underlying Data Structure:
   Choose a red-black tree in which each node v_i contains the endpoints of the intervals.
   Additional Information:
   Each v_i has one $p[v_i]$. If the v_i is the left endpoint of the interval, then $p[v_i]$ = 1, else $p[v_i]$ = -1. Also, each v_i has one $w[v_i]$ which means $\sum p[v_j](v_j$ is in the tree rooted by $v_i)$.
   Maintaining the Information:
   1)INTERVAL-INSERT:
   Using the insert method in red-black tree, and add the $p[v_i]$ to all the nodes' $w[v_i]$ which $v_i$has passed through.
   2)INTERVAL-DELETE:
   Using the delete method in red-black tree, and reduce the $p[v_i]$ to all the nodes' $w[v_i]$ which v_i has passed through.
   3)FIND-POM:

$$m[v_i] = max(w[lchild(v_i)] + p[v_i] + m[rchild(v_i)], w[lchild(v_i)] + p[v_i], m[lchild(v_i)]) \quad (1)$$

   Then sort the $m[v_i]$ and find the max ones. We can easily generate the biggest overlap intervals by their value and $p[v_i]$.
   Developing Operations:
   1) INTERVAL-INSERT
   2)INTERVAL-DELETE
   3)FIND-POM