1. (CLRS 13.1-5) Show that the longest simple path from a node $x$ in a red-black tree to a descendant leaf has length at most twice that of the shortest simple path from node $x$ to a descendant leaf.

2. (CLRS 13.1-6) What is the largest possible number of internal nodes in a red-black tree with black-height $k$? What is the smallest possible number?

---

1Collaboration is allowed, even encouraged, provided that the names of the collaborators are listed along with the solutions. Students must write up the solutions on their own.
3. (CLRS 13-2) The join operation takes two dynamic sets $S_1$ and $S_2$ and an element $x$ such that for any $x_1 \in S_1$ and $x_2 \in S_2$, we have $key[x_1] \leq key[x] \leq key[x_2]$. It returns a set $S = S_1 \cup \{x\} \cup S_2$. In this problem, we investigate how to implement the join operation on red-black trees.

(a) Given a red-black tree $T$, we store its black-height as the field $bh[T]$. Argue that this field can be maintained by RB-INSERT and RB-DELETE without requiring extra storage in the tree and without increasing the asymptotic running times. Show while descending through $T$, we can determine the black-height of each node we visit in $O(1)$ time per node visited.

We wish to implement the operation RB-JOIN($T_1, x, T_2$) which destroys $T_1$ and $T_2$ and returns a red-black tree $T = T_1 \cup \{x\} \cup T_2$. Let $n$ be the total number of nodes in $T_1$ and $T_2$.

(b) Assume without loss of generality that $bh[T_1] \geq bh[T_2]$. Describe an $O(\log n)$ time algorithm that finds a black node $y$ in $T_1$ with the largest key from among those nodes whose black-height is $bh[T_2]$.

(c) Let $T_y$ be the subtree rooted at $y$. Describe how $T_y$ can be replaced by $T_y \cup \{x\} \cup T_2$ in $O(1)$ time without destroying the binary-search-tree property.

(d) What color should we make $x$ so that red-black properties 1, 2, and 4 are maintained? Describe how property 3 can be enforced in $O(\log n)$ time.

(e) Argue that the running time of RB-JOIN is $O(\log n)$.