ALGORITHMS, FALL 2018, HOMEWORK 6

Due Thursday, October 18 at 11:59pm
but there will be no penalty if you submit up to 48 hours later.

We can only guarantee that your work will be graded before the exam if you submit on time.
Worth 1% of the final grade.

1. In a fully balanced (symmetric) binary search tree (BST) with n nodes, the root has rank $\lceil \frac{n}{2} \rceil$. Given a BST that is a valid red-black tree, how extreme (e.g., how low) could the rank of the root be? Use Θ-notation for your answer, otherwise lower order terms may become annoying. Include a description or drawing of a tree that corresponds to your answer (in other words explain how you're getting your answer).

   The height in a red-black tree is at most $2 \log n$.

   It is easy to calculate the rank of the root because its just the size of each subtree + 1.

   The best-case scenario for a subtree is always a subtree of height $2 \log n - 1$ (because we can't go up 2 levels or the tree wouldn't be balanced).

   That means that the lowest rank the root can have is the size of that subtree + 1.

   The number of nodes in a tree of size $2 \log n - 1$ is $2^{2 \log n} + 1$

   This means that the lowest rank for the root node is $\Theta(\sqrt{n})$.