CMPS 2200 Introduction to Algorithms – Fall 17

9/20/17

4. Homework

Due **9/27/17** at the beginning of class

Remember, you are allowed to turn in homeworks in groups of two. One writeup, with two names.

1. $1, 2, 3, \ldots, 15$ **(8 points)**
   Justify your answers shortly.
   (a) (2 points) Draw the binary search tree that results from inserting the numbers $1, 2, 3, \ldots, 15$ in that order. What is the total runtime for inserting the numbers $1, 2, 3, \ldots, n$ into a binary search tree in that order?
   (b) (2 points) Give an order for inserting the numbers $1, 2, 3, \ldots, 15$ into a binary search tree such that the result is a perfectly balanced complete binary tree.
   (c) (2 points) Draw the red-black tree that results from inserting the numbers $1, 2, 3, \ldots, 15$ in that order. What is the total runtime for inserting the numbers $1, 2, 3, \ldots, n$ into a red-black tree in that order?
   (d) (2 points) Draw the B-tree with minimum-degree k=2 that results from inserting the numbers $1, 2, 3, \ldots, 15$ in that order. What is the total runtime for inserting the numbers $1, 2, 3, \ldots, n$ into a B-tree with minimum-degree k in that order?

2. **Black-Height (6 points)**
   Write pseudocode for a function `int computeBH(RBnode root)` that takes the root node of a candidate red-black tree and returns the black-height of the tree if the tree is a valid red-black tree, or `-1` otherwise.
   - The class `RBnode` stores the `key`, the `color`, and references `left` and `right` to its two children.
   - You can assume all `null`'s are black and that each node's `color` is either `RED` or `BLACK`.
   - `int computeBH` should check the blackness of the root separately and then call a recursive function `int computeBH_rec(RBnode node)` that returns the black-height of the subtree rooted at `node` if red-black tree properties 4 and 5 are fulfilled, or `-1` otherwise.

   Analyze the runtime of your function.

3. **B-tree-search using binary search (4 points)**
   Consider changing B-TREE-SEARCH to use **binary search** instead of linear search on the key.
   (a) What is the number of disk accesses? Justify your answer.
   (b) Show that the CPU time is $O(\log n)$, which is independent of k.