CS 5633 Analysis of Algorithms – Spring 16

Exam 2

NAME:

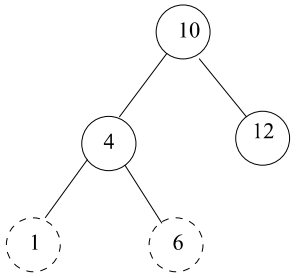
- This exam is closed-book and closed-notes, and electronic devices such as calculators or computers are not allowed. You are allowed to use a cheat sheet (half a single-sided letter paper).
- Please try to write legibly – if I cannot read it you may not get credit.
- **Do not waste time** – if you cannot solve a question immediately, skip it and return to it later.

1) Red-Black Trees		19
2) B-Trees		18
3) Range Trees		18
4) Sorting		15
5) Dynamic Programming		30
		100

1 Red-Black Trees (19 Points)

1. What is the maximum number of red nodes that can be in a red-black tree with black-height h? Express the answer as a function of h.

2. Consider the following red-black tree where the solid nodes are black and the dotted nodes are red (nil leaf nodes not shown in this figure):



Suppose we insert the following three nodes into the tree in order. Show the resulting tree after each insertion.

(a) Insert 5

(b) Insert 5.5

(c) Insert 3

2 B-Trees (18 Points)

Consider making the following sequence of inserts into an initially-empty B-tree with $t = 2$.

$$5, 9, 13, 17, 12, 10, 11, 19, 14$$

Show the tree before and after each split (i.e. you do not need to draw it after each insertion if the insertion does not force a split).

3 Range Trees (18 Points)

Consider a range tree on the 16 following 1D points:

$$6, 7, 9, 10, 11, 15, 17, 18, 22, 23, 27, 29, 30, 31, 34, 36$$

1. Draw the 1D range tree.
2. Show how the search works on your tree when searching for the points in the interval $[8, 13]$ (show any split nodes and search paths).

4 Sorting Runtimes (15 Points)

Suppose we want to choose a sorting algorithm with the best worst-case running time possible for the assumed input. What sorting algorithm would you use? What would the running time of the algorithm be? Justify your answer.

1. An array of n binary numbers.

2. An array of n credit card numbers.

3. An ranking of n candidates for a job.

5 Dynamic Programming (30 Points)

Suppose you want to go hiking on the Pacific Crest Trail, a long trail running from Mexico to Canada. It will take many days to complete the hike, and you will need to pay to stay at hostels each night. There are n hostels on the trail, and it costs c_i dollars to stay at hostel h_i. You want to minimize the amount of money that it costs to complete the trail, but you cannot hike more than 20 miles per day. So we want to compute a set of hostels that we will stay at to minimize the sum of their costs so that no two hostels are more than 20 miles apart. You can assume that hostels are spaced so that a feasible solution always exists.

1. Consider a brute force algorithm that considers every possible subset of hostels. What is the running time of this algorithm?

2. Let $a[i]$ denote the cost of an optimal solution considering only the first i hostels. Give a recursive definition for $a[i]$.

3. Give a dynamic programming algorithm based on your recursive defintion. You can use top-down or bottom-up.