MA/CSSE 473 – Design and Analysis of Algorithms

Homework 14 (37 points total)

These are to be turned in as hard copy. You can write solutions out by hand, or write them on your computer and print them. If there are multiple pages, please staple them together.

When a problem is given by number, it is from the textbook. 1.1.2 means “problem 2 from section 1.1” .

Problems for enlightenment/practice/review (not to turn in, but you should think about them):

How many of them you need to do serious work on depends on you and your background. I do not want to make everyone do one of them for the sake of the (possibly) few who need it. You can hopefully figure out which ones you need to do.

8.3.5 (Root of Optimal tree)
8.3.2 (Time and space efficiency of optimal BST calculation)
8.3.8 (n^2 algorithm for optimalBST. Not for the faint of heart!)
------ For the frequencies of the Day 33 class example (AEIOU), find he optimal tree if we consider only successful searches (set all q_i to 0)
------ For the frequencies of the Day 33 class example (AEIOU), find he optimal tree if we consider only unsuccessful searches (set all p_i to 0)
9.1.1 (Greedy change-making not optimal)
9.1.5 (greedy bridge crossing)

Problems to write up and turn in:

1. (5) 8.3.4 (Sum for optimalBST in constant time).
2. (10) 8.3.6 (optimalBST--successful search only--if all probabilities equal)
3. (10) 9.1.3 (Greedy job scheduling)
4. (6) 9.1.7b (Prim example) Start with node a. Whenever you have a choice because edge weights are equal, choose the vertex that is closest to the beginning of the alphabet. Then everyone should get the same answer, making it easier for us to check your work.
5. (6) 9.2.1b (Krushkal example) Whenever you have a choice because edge weights are equal, choose the edge whose vertices are closest to the beginning of the alphabet. Then everyone should get the same answer, making it easier for us to check your work.

Looking ahead to HW 15

It will include problem 8.3.10b. In addition, you can do 8.3.10c for 50 points extra credit. If you do it, you should not just design the algorithm, but also implement it, run it and show the output for a few interesting cases. If you plan to do this, you may want to start soon.

Don't forget the Boyer-Moore demonstration implementation problem

Described in Day 28 PowerPoint Slides