MA/CSSE 473 – Design and Analysis of Algorithms

Homework 14 (90 points total) plus extra-credit opportunity

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 optimal BST. 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. (15) 8.3.10b (how many different parenthesizations?) . Prove youtr answer by mathematical induction.

2. (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.

3. (5) 9.1.8 (Prim prior connectivity check?)

4. (10) 9.1.11 (change value of an item in a min-heap)

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.

6. (8) 9.2.2 (Kruskal TF questions) Briefly explain your answers.

7. (5) 9.2.8 (efficiency of *find* in union-by-size)

8. (8) 9.4.1 (Huffman code for specific data)
   (a) 4 points. When there is a choice due to a tie, place the one that appears first in the problem statement's character list "on the left" in the tree. (b) 2 points. (c) 2 points.

9. (12) 9.4.3 (Huffman TF questions) (a) 5 points (b) 7 points

10. (5) 9.4.4 Once you have figured out the answer, describe a set of probabilities (or frequencies) that make the maximum happen.

11. (10) [Added question] Answer the questions from class about the induction proof of the correctness of Kruskal's algorithm. See details below.

Extra credit: You can do 8.3.10c for 30 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.

Added problem on Kruskal's algorithm:

The questions

(a) How do we know that v was already part of some connected component of G'?

Does the addition of e to C satisfy the hypothesis of the lemma? For each statement below, explain why it is true.

(b) G' is a subgraph of some MST for G:

(c) C is a connected component of G':

(d) e connects a vertex in C to an vertex in G – C:

(e) e satisfies the minimum-weight condition of the lemma:

The algorithm:

- To find a MST for a connected undirectedG:
  - Start with a graph G' containing all of the n vertices of G and no edges.
  - for i = 1 to n – 1:
    - Among all of G's edges that can be added without creating a cycle, add one (call it e) that has minimal weight.

The property we are trying to prove: Before every loop execution, G' is a subgraph of some MST of G.

Proof is (of course) by induction on i.
BASE CASE: When I = 1, G' consists of only vertices of G. Since all vertices must be part of any MST for G, G is a subgraph of every MST.

INDUCTION STEP. Assume that G' is a subgraph of an MST for G. Choose e according to the above algorithm. Show that G' $\cup\{e\}$ is a sungraph of an MST of G.

The Lemma we want to use: Let G be a weighted connected graph with a MST T; let G′ be any subgraph of T, and let C be any connected component of G′. If we add to C an edge $e=(v,w)$ that has minimum-weight among all of the edges that have one vertex in C and the other vertex not in C, then G has an MST that contains the union of G′ and e.

In order to be able to use the lemma, we have to pick a connected component of C, and show that it satisfies the conditions of the lemma. We let v be one of the vertices of e, and let C be the connected component of G' that contains v.

Within this context, answer the 5 questions above.