MA/CSSE 473 – Design and Analysis of Algorithms

Homework 14 (49 points total) plus extra-credit opportunity updated for summer 2012

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. **All problem numbers in this assignment are the same in editions 3 and 4.**

9.4.6 (linear time Huffman code algorithm)
10.2.2a (maximum flow example)
10.2.5 (maximum flow algorithm for tree)
10.2.6a (prove equation 10.9)
10.4.3 (stable marriage example)
10.4.6 (unique stable marriage solution)
10.4.10 (roommate problem)

Problems to write up and turn in:

1. (5) 9.4.4 (maximal Huffman codeword length) Once you have figured out the answer, describe a set of probabilities (or frequencies) that make that maximum happen.

2. (10) 9.4.10 (card guessing)

3. (10) Not for summer [Added question]
   Answer the questions from class(below) about the induction proof of the correctness of Kruskal's algorithm. See details below.

4. (6) 10.2.1 (sources and sinks with negative weights)

5. (5) 10.2.3a (Maximum flow solution unique?) Explain your answers.

6. (5) 10.2.4a (Maximum flow single source and sink)

7. (8) 10.4.1 (stable marriage example)

8. (4) 10.4.2 (stable marriage check algorithm)

9. (6) 10.4.5 (time efficiency of stable marriage Algorithm)

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 (#3, not for summer):

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.