Head-Banging Session 5
Fall 2008 CS 473: Algorithms

Problem 1. **[Saving with Union-Find]**
Show the Union-Find forest that results from the following sequence of operations on the ten singleton elements $a, b, \ldots, j$. Do this exercise *two* ways: with and without path compression. On which steps does path compression save running time?

UNION(FIND(a), FIND(e))
UNION(FIND(c), FIND(b))
UNION(FIND(b), FIND(h))
FIND(e)
UNION(FIND(f), FIND(e))
UNION(FIND(j), FIND(g))
UNION(FIND(g), FIND(b))
UNION(FIND(i), FIND(c))
FIND(j)
UNION(FIND(e), FIND(d))
UNION(FIND(a), FIND(c))
FIND(g)

Problem 2. **[Practice With Huffman]**
The following table contains Huffman code for the given nine-letter alphabet.

Letter — Code Word
O — 1111100
D — 1111101
M — 11110
R — 11101
S — 11100
A — 100
I — 110
T — 101
E — 0

Build the Huffman coding tree that corresponds to the above code. Is the given encoding an optimal encoding? If not, how could you improve it? What are possible frequencies for the letters in this alphabet that would result in the optimal tree that you have drawn?

Problem 3. **[Properties of Huffman Codes]**
Prove that in the Huffman coding algorithm if some character occurs with frequency greater than $\frac{2}{5}$, there exists a codeword of length one. Also prove that if all characters occur with frequency less than $\frac{1}{3}$ then, there is no codeword of length one. (Assume that all frequencies add up to 1.)

Problem 4. **[Codeword Length]**
SC Decoders manufactures hardware to decode Huffman-encoded game files in an XBox when you play them over a network. Its decoding hardware stores an entire codeword in a register when it is looking for a match. The uncompressed game files contain characters from an alphabet of size n. What is the minimum required register width (number of bits) that can support all Huffman codes for n characters, no matter what the frequencies $f_1, f_2, ..., f_n$ are? What relationship between the frequencies $f_1, f_2, ..., f_n$ causes the worst case (longest codeword) to occur?

Problem 5. **[Bottlenecks]**
The bottleneck of a spanning tree is the edge that has the greatest cost. The bottleneck of a graph is the smallest of all the bottlenecks that exist for each possible spanning tree that could be could be built. (In other words, we examine each spanning tree that can be built and for each one find the edge that is the worst bottleneck among all of them.) Give an algorithm that finds the bottleneck in a graph. The running time should be at most $O((\mid V \mid + \mid E \mid) \log(\mid V \mid + \mid E \mid))$.

Problem 6. **[Reciprocal Edge Weights]**
Suppose we are given a connected, undirected graph G with positive costs on the edges. Now we construct a new graph G', which is the same as G except if the cost of an edge e in G is c_e, the cost of the edge in G' is $1/c_e$. Is the Minimum Spanning Tree of G a Maximum Spanning Tree of G'? Prove or disprove.

Problem 7. **[MST Properties]**
Let T be an MST of a graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H. (A subgraph $H = (V_H, E_H)$ of a graph $G = (V_G, E_G)$ has $V_H \subset V_G$ and $E_H \subset E_G$.)