Fall 2006 **CMSC 451: Homework 5** William Gasarch

Due Wedensday Oct 18, at Beginning of class

1. (0 points). Write your name clearly. Staple your HW. READ Chap 4 on Huffman Codes and Chapter 5.1-5.5.

2. (20 points) The Huffman Code maps elements of Σ to elements of $\{0,1\}^*$. We want to look at codes that map elements of Σ to elements of $\{0,1,2\}^*$.

   (a) Give an algorithm that will, given an alphabet Σ and frequencies p_σ for every $\sigma \in \Sigma$, the optimal prefix free code mapping to $\{0,1,2\}^*$.

   (b) Use your algorithm on the following input: $\Sigma = \{a,b,c,d,e,f,g,h,i,j\}$.

$$
\begin{aligned}
p_a &= 0.04 \\
p_b &= 0.08 \\
p_c &= 0.1 \\
p_d &= 0.14 \\
p_e &= 0.13 \\
p_f &= 0.09 \\
p_g &= 0.12 \\
p_h &= 0.11 \\
p_i &= 0.02 \\
p_j &= 0.17
\end{aligned}
$$

   (c) Use the algorithm from class and in the book that produces prefix-free codes with output $\{0,1\}^*$ on the same data.

   (d) Find the average length encoding in both cases. Which one is better?

3. (20 points) Recall that if $a_1, \ldots, a_n$ is a list then an INVERSION in the list is a pair (i,j) such that $i < j$ but $a_i > a_j$. Dr. Am objects to this and defines a new term: an AM-INVERSION is a pair (i,j) such that $i < j$ but $a_i > 3a_j$. Give an algorithm that finds the number of AM-inversions in $O(n \log n)$ steps. (No proof required.)

TURN OVER

4. (20 points) Consider the following problem. You are given n objects. The ONLY operation you can perform on them is, given two of them, tell if they are THE SAME or DIFFERENT. You want to determine if there exists $n/2$ objects that are the same. You COULD do this with $O(n^2)$ calls to the operation (ask for each pair SAME or DIFF). Devise an algorithm that takes $O(n \log n)$ calls to that OPERATION. The algorithm should be clear and you need to do an analysis to prove its $O(n \log n)$.

5. (20 points) We define the cost of the path DIFFERENTLY then usual. If

$$(v_1, v_2), (v_2, v_3), \ldots, (v_{a-1}, v_a)$$

is a path then we define the COST of this path to be the SUM of the two MOST EXPENSIVE weights.

For example, in the above, if

$$(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6)$$

is a path, and $w(v_1, v_2) = 4$, $w(v_2, v_3) = 8$, $w(v_3, v_4) = 2$, $w(v_4, v_5) = 3$, $v(v_5, v_6) = 5$ then the cost of the path is $8 + 5 = 13$ which is the sum of the two most expensive edges.

Devise an algorithm that will, given a weighted graph $G = (V, E)$ and a node $s \in V$, output, for all v the shortest distance of s to v and the path that yields that distance. (So the output is a set of $n - 1$ costs and paths, one for each $v \in V - \{s\}$.) The algorithm should be a variant of Dijistra's algorithm. No proof of correctness is required.

6. (20 points) We want to find MST for graphs where the weights are all positive integers bounded above by a number C. Give an algorithm for this that runs in time $O(CV + E)$. You need not prove that it produces an MST but you do need to show that its run time is $O(CV + E)$.