University of Nevada, Las Vegas Computer Science 477/677 Spring 2015

Assignment 5: Due March 5, 2015

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. But the handwriting on this document must be your own. You may attach extra sheets, using a stapler.

"Designing" an algorithm does not mean writing code. In fact, explicit hand-written pseudo-code can be hard to grade (and you don't want to upset the grader, do you?) You should describe your algorithm in English, although you can use **small** bits of hand-written pseudo-code to clarify, as needed.

Of course, you may actually wish to encode and then execute your algorithm. This does not relieve you of the obligation to completely explain your algorithm in your own handwriting.

1. Describe an algorithm for finding the longest non-decreasing subsequence of a given sequence of n integers. For example, the longest non-decreasing subsequence of 5,1,7,3,6,2,6,8,2 is 1,3,6,6,8.

   Your algorithm should take $O(n \log n)$ time, where we assume that it takes $O(1)$ time to compare two integers.

   Test your algorithm by finding the longest non-decreasing subsequence of the sequence below, where $n = 100$. (There could be more than one best solution.)

   54, 60, 72, 86, 62, 88, 57, 88, 46, 67, 69, 44, 49, 36, 96, 13, 104, 36, 47, 57, 89, 92, 63, 59, 68, 52, 105, 97, 21, 48, 23, 80, 18, 53, 100, 113, 95, 33, 106, 107, 117, 68, 100, 102, 68, 75, 101, 91, 35, 97, 88, 84, 40, 80, 121, 103, 80, 39, 70, 77, 56, 49, 108, 105, 77, 54, 89, 47, 35, 67, 96, 50, 97, 59, 98, 76, 132, 129, 107, 84, 75, 102, 84, 132, 111, 52, 49, 140, 110, 110, 112, 63, 67, 82, 59, 123, 79, 109, 85, 82.

2. You are given two sorted lists of integers, one of length n, the other of length $n+1$. Assume that the integers are unique. Describe an efficient algorithm which finds the median of the union of those lists.

   For example, if $n = 5$ and the two lists are 2,10,12,14,15 and 1,3,7,8,9,12 the median item is 9.

   Assume it takes $O(1)$ time to compare two integers. What is the asymptotic time complexity of your algorithm?

3. The *edit distance* between two strings of symbols is defined to be the number of editing changes needed to change one string into the other. Each editing change can do one of three things: replace a symbol by another symbol, delete a symbol, or insert a symbol. For example, the edit distance between "apresheation" and "appreciation" is 4.

   Give an efficient algorithm for finding the edit distance between two strings.[1]

[1] This problem is important for determining relationships between proteins, DNA, and RNA, for instance.

4. You are given a weighted dag (dag = directed acyclic graph) G of n vertices and m arcs (some people use the word "arc" instead of "edge" for directed graphs). Describe an $O(n+m)$-time algorithm for finding the least weight path from s to t. (In a directed graph, we assume that "path" means "directed path.")

5. You are given a weighted dag (dag = directed acyclic graph) G of n vertices and m arcs (some people use the word "arc" instead of "edge" for directed graphs). You need to find two paths from s to t, such that every vertex of G belongs to at least one of the two paths. That might not be possible: if the vertices of G are $\{s, x, y, z, t\}$ and the arcs are $\{(s, x), (s, y), (s, z), (x, t), (y, t), (z, t)\}$, then no two paths can cover all three of the middle vertices: at least one of $\{x, y, z\}$ will not be covered.

   Design an algorithm for finding two paths which cover all vertices of G, such that the sum of the weights of those two paths is minimum. Your algorithm should halt with the message, "Impossible," if no such two paths exist.

   Assume that G is given to you as an array of out-neighbor lists. Your algorithm may need to contain subroutines, for example, to topologically sort the vertices.

   I know it seems incredible, but there is an $O(n + m)$-time algorithm for this problem.[2] I'll be happy if you can find one that takes quadratic time.

[2] Of course, I might be wrong about this. Senile amnesia, you know.