Assignment 3

1 (Exercise 23.1-6) Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counterexample.

2 In a city there are N houses, each of which is in need of a water supply. It costs W_i dollars to build a well at house i, and it costs C_{ij} to build a pipe in between houses i and j. A house can receive water if either there is a well built there or there is some path of pipes to a house with a well. Give an algorithm to find the minimum amount of money needed to supply every house with water.

3 (Exercise 23.2-8) Professor Toole proposes a new divide-and-conquer algorithm for computing minimum spanning tree, which goes as follows. Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1(V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edges in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

4 We describe here one simple way to merge two sorted lists: Compare the smallest numbers of the two lists, remove the smaller one of the two from the list it is in, and place it somewhere else as the first number of merged list. We now compare the smallest numbers of the two lists of remaining numbers and place the smaller one of the two as the second number of the merged list. This step can be repeated until the merged list is completely built up. Clearly, in the worst case it takes $n_1 + n_2 - 1$ comparisons to merge two sorted lists which have n_1 and n_2 numbers, respectively. Given m sorted lists, we can select two of them and merge these two

lists into one. We can then select two lists from the $m-1$ sorted lists and merge them into one. Repeating this step, we shall eventually end up with one merged list.

Let A_1, A_2, A_3, A_4 be four sorted lists with 73, 44, 100, 55 numbers, respectively.

(a) Determine an order to merge the four lists so that the total number of comparisons is minimum.

(b) Describe a general algorithm for determining an order in which m sorted lists A_1, A_2, ..., A_m are to be merged so that the total number of comparisons is minimum. Prove that your algorithm is correct.

5 (Exercise 16.3-4) Prove that if we order the characters in an alphabet so that their frequencies are monotonically decreasing, then there exists an optimal code whose codeword length are monotonically increasing.

6 Suppose that we have a set of n files, $f_1, f_2, \ldots, f_n$, of sizes $m_1, m_2, \ldots, m_n$. (Assume that the m_i's are all distinct.) We wish to store the files sequentially on a single tape, in some order, and retrieve each file exactly once, in the reverse order. The retrieval of a file involves rewinding the tape to the beginning and then scanning the files sequentially until the desired file is reached. The *cost* of retrieving a file is the sum of the sizes of the files scanned plus the size of the file retrieved. (Ignore the cost of rewinding the tape.) The *total cost* of retrieving all the files is the sum of the individual costs.

(i) Suppose that the files are stored in some order $f_{i_1}, f_{i_2}, \ldots, f_{i_n}$. Derive a formula for the total cost of retrieving the files, as a function of n and the m_{i_k}'s.

(ii) Describe briefly a **greedy strategy** to order the files on the tape so that the total cost is minimized, and prove that this strategy is indeed optimal.

7 Let V be a fixed set of n vertices. Consider a sequence of m undirected edges e_1, e_2, ..., e_m. For $1 \leq i \leq m$, let G_i denote the graph with vertex set V and edge set $E_i = \{e_1, ..., e_i\}$. Let c_i denote the number of connected components of G_i. Design an algorithm to compute c_i for all i. Your algorithm should be asymptotically as fast as possible. What is the running time

of your algorithm?

8 (Exercise 16.1-3) Suppose that we have a set of activities to schedule among a large number of lecture halls. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall. (Note: Each activity can be represented by an interval $[s_i, f_i)$.)