

CMSC 451: Homework 2, Spring 2004

Due at the beginning of class on February 24.

Warning: some of the problems require thought - do not wait until the last day to start working on them!

If you cannot come up with algorithms that run in the required time, then provide (correct) slower algorithms for partial credit. Write your answers using *pseudo-code* in the same style as the textbook. These make the algorithm description precise, and easy to read (as opposed to code in C or some other language).

1. We are given an undirected graph $G = (V, E)$ with a weight $w(u, v)$ for each edge (u, v) ; importantly, each given $w(u, v)$ is **non-negative**. Define a minimum spanning tree (MST) of G as usual. In addition, define the *energy* of each edge (u, v) to be $(w(u, v))^2$, i.e., the square of the weight of the edge. Now define a *minimum-energy spanning tree* (MEST) of G as a spanning tree, which minimizes the total energy of the edges used. Present an efficient algorithm for finding a spanning tree that is *both* an MST and an MEST; prove the correctness of your algorithm, and say what its time-complexity is. (The minimum-energy version is useful in wireless networking, as will be explained later.)
2. Consider the Activity Selection Problem from Section 16.1 of the textbook. Consider the following proposed algorithm for this problem. Let the difference $f_i - s_i$ be called the *length* of activity a_i . The algorithm repeatedly chooses an activity of smallest length, throws away all other activities that overlap with this activity, and repeats this until no activities are left. You can verify that the set of activities chosen by this greedy algorithm are indeed mutually compatible. Show that this algorithm **does not** always choose a maximum-sized subset of mutually compatible activities.
3. You are given a set $S = \{a_1, a_2, \dots, a_n\}$ of n tasks, where task a_i needs p_i units of processing time to complete, once it has started. You want to schedule these tasks on one computer; the computer can only process one task at a time. Given a schedule, let c_i be the *completion time* of task a_i : i.e., the time at which a_i completes processing. You wish to schedule the tasks in order to minimize the *total completion time* of the tasks: i.e., you want to minimize $\sum_{i=1}^n c_i$. For example, suppose $n = 2$, $p_1 = 3$, and $p_2 = 5$. If we schedule a_1 first and a_2 next, then $c_1 = 3$ and $c_2 = 8$. Otherwise, if we schedule a_2 first and a_1 next, then $c_1 = 8$ and $c_2 = 5$. Since $3 + 8 < 8 + 5$, the first schedule is the optimal schedule in this example.
 - (a) Suggest an algorithm to minimize the total completion time. Each task must run non-preemptively: i.e., once task a_i is started, it must run continuously for p_i units of time. (**Hint:** Consider small examples with $n = 2$, $n = 3$ etc., to get a hint about how an optimal schedule will look.)

(b) Prove that your algorithm indeed minimizes the total completion time.

4. **(Only for graduate students)** We are given a connected undirected graph $G = (V, E)$ with a weight $w(u, v)$ given for each edge (u, v) , such that all the given weights are *distinct* (i.e., no two are the same). Consider the following algorithm for finding a minimum spanning tree (MST) in G . Initialize a graph H to be the graph G (as we proceed, we will delete some edges from H , as seen below). We first sort the edges in *decreasing* order of weight, and consider the edges in this decreasing-weight order. When considering an edge (u, v) , we test if deleting this edge will still keep the graph H connected; if so, we delete this edge from H . (Note that the test performed, and the deletion, is on H , and not on G .)

After considering all the edges, we output the edges in the remaining graph H , as an MST.

(a) Prove that the final set of edges that is output, is indeed a spanning tree.

(b) Is the output indeed an MST? Justify your answer.