CS170, Fall 1994
Midterm #2
Professor Vazirani

There are 4 independent problems of equal weight. You need to discard one problem of your choice and do the other three. You will be graded on these three. (If you turn in something for all problems, only the first three will be graded)

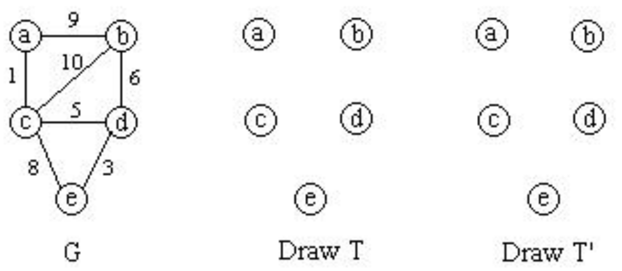
When using material from lecture, textbook or discussion session, you do not need to repeat it entirely. It is sufficient to refer to it clearly.

Please start each new problem on a new sheet of paper. Write your name on each sheet of paper. On the first page, write you ID number and which section you are enrolled in (AM or PM) *(1 point)*

Problem #1 - Dynamic Minimum Spanning Tree *(35 points)*

Let G = V(V,E) be an undirected graph, with a weight function w(e) &gt 0 for each edge e &lt E. Let T be a Minimum Spanning Tree (MST) of G for this weight function. We consider the problem which takes as input a minimum spanning tree T of G and a request "Decrease the weight of edge e_i down to $w'(e_i)$", where $w'(e_i)$ &lt $w(e_i)$, and outputs the new minimum spanning tree T' of G with this new wieght.

1.1 Example. *(5 points)* For example, consider the graph G below. Draw a minimum spanning tree T of G. If the request is "Change teh weight of edge {b, d} from 10 to 3", then draw the new MST T'.



1.2 Scheme *(15 points)* Give a high-level algorithmic scheme for constructing T' (no proof of correctness required). You may wish to consider separately the cases e_i &lt T and when e_i |&lt T.

1.3 Algorithm and running time. *(15 points)* Give an algorithm which, given Gand T represented by adjacency lists, the weight function in an array w, and a request (e_i, w') to decrease the weight of e_i from $w(e_i)$ to w', outputs tree T'. Argue that the running time of your algorithm is O(V).

Problem #2 - Course-scheduling in two lecture halls *(35 points)*

Suppose that we have a list of n classes, where class $c_i = (s_i, f_i)$ has starting time s_i and finish time f_i. We have two lecture halls, and the problem is 1) to decide whether it is possible to assign each class to a lecture hall (hall[i] = j if class ci is assigned to lecture hall j, where j = 0 or j = 1) so that classes which have been assigned to the same lecture hall do not overlap, and 2) to give an assignment when it is possible.

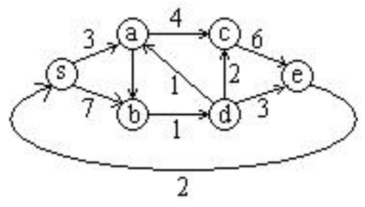
2.1 Scheme. *(12 points)* Propose a greedy high-level algorithmic scheme for this problem.

2.2 Correctness. *(11 points)* Prove by induction that the algorithm always finds an assignment when there is one. Be sure to state your inductive statement clearly.

2.3 Algorithm and running time. *(12 points)* Give a detailed algorithm implementing your scheme. What is the running time ? Justify briefly.

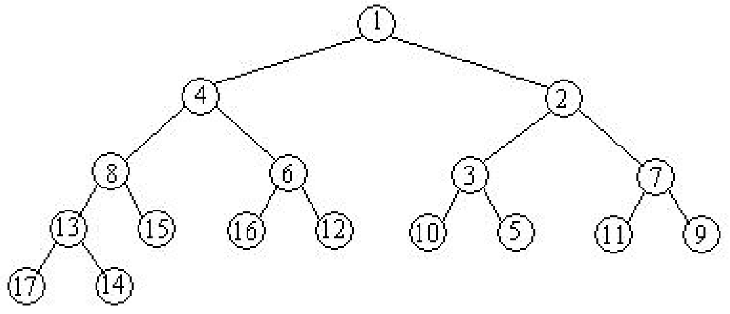
Problem #3 - Shortest path tree *(35 points)*

Given a weighted strongly connected directed graph G = (V, E) with positive edge weights, and a vertex s of G, we say that a subgraph T = (G, E') of G is a **shortest-path tree** rooted at s if the following three conditions hold:
1) The undirected version of T is a tree,
2) For any vertext w, there is a unique path from s to w in T, and
3) the weight of that path is equal to the smallest weight of any path from s to w in G.
We consider the problem which takes as input a graph G = (V, E) and a subgraph T = (V, E') of G, both represented by adjacency lists, an array w of edges weights, and a vertext s of G, and which decides whether T is a shortest-path tree rooted at s. Our goal is an O(V + E) algorithm.
3.1 Example. *(5 points)* Draw a shortest-path tree of the graph G below.
3.2 Scheme. *(10 points)* How would you decide whether conditions 1 and 2 above are satisfied? (No proof required).
3.3 Scheme. *(10 points)* How would you decide whether condition 3 above is satisfied? (No proof required).
3.4 Algorithm and running time. *(10 points)* Design an algorithm for deciding whether T is a shortest-path tree rooted at v. Prove that its running time is O(V + E).



Problem #4 - Selection in a heap *(35 points)*

Suppose you are given a binary heap A containing n elements, with the minimum at the root. you are asked to find the kth smallest element in the heap, where k can be any number between 1 and n.
4.1 Naive scheme. *(5 points)* Give an O(k * log n) algorithm for solving this problem. Justify.
4.2 Idea for scheme. *(10 points)* Give an O(1) algorithm for finding the 4th smallest element. Instead of writing the algorithm formally, just explain how you would proceed in the example below. (Do not prove that the running time is O(1))
4.3 Scheme. *(10 points)* Give an algorithmic scheme for finding the kth smallest element.
4.4 Implementation. *(10 points)* Implement your algorithm. What is the running time? Justify. (We expect you to find either an O(k^2) running time, or, with an improved implementation, an O(k * log k) running time).


1
4
2
8
6
3
7
13
15
16
12
10
5
11
9
17
14