COMP 271 Design and Analysis of Algorithms
2005 Spring Semester
Written Assignment 2
Distributed: March 31, 2005
Due: April 19, 2005 ***at start of class***

Note: Please follow the guidelines on doing your own work and avoiding plagiarism on the class home page.

2nd Note: Doing this assignment *before* the midterm is a good excercise!

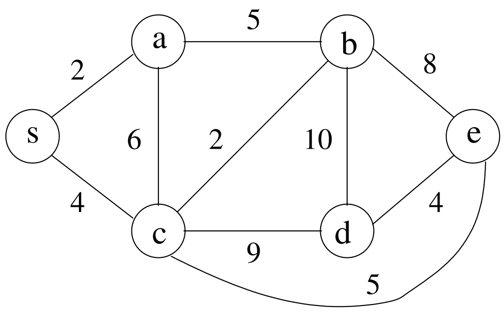
Problem 1. (MST) Given a connected and undirected graph $G = (V, E)$ such that the edge weights are all distinct. Let T be a MST of G. Let (x, y) be an edge of G which is **NOT** in T. Suppose we *decrease* the weight of (x, y) to a new value k, making a new graph G' so that the edge weights of G' are still all distinct.

a. Under what circumstances will T be a MST of G'? Explain your answer.

b. (i) Under what circumstances will T not be a MST of G'? Explain your answer.
   (ii) Suppose the weight of (x, y) has been decreased to a value such that T is not a MST of G'. Describe how we can use T to construct a MST T' of G'. You also need to show that the tree constructed is a MST of G'.

Hint: You may use the fact that the heaviest edge in any cycle of G will not be included in any MST.

Problem 2. (Kruskal's Algorithm) Trace the executions of Kruskal's Algorithm on the following graph. Use vertex s as the root vertex, and show every step of the algorithm (similar to CLRS p. 568).



Problem 3. (Dijkstra's Algorithm) Consider the MTR graph in Hong Kong, i.e., the nodes are the MTR stations, two nodes u and v are connected by two directed edges (u, v) and (v, u) iff the two MTR station are connected by a train. The edges have weights according to the approximate travelling times. For example 1.5 minutes between stations along the Island Line. We add one more node HKUST with edges to at least Choi Hung and Hang Hau (weights according to the time of the minibus ride). Assume, we solve the SSSP with source node HKUST using Dijkstra's algorithm.

a. Label the nodes by $1, 2, 3, \ldots$ in the order the algorithm dequeues them from the priority queue.

b. Which number is assigned to Central?

c. What is the length of the shortest path to Tung Chung?

Problem 4. (0-1 Knapsack) In this question, you are required to solve the 0-1 Knapsack problem for two knapsacks. You are given a set of n objects. The weights of the objects are $w_1, w_2, \ldots, w_n$, and the values of the objects are $v_1, v_2, \ldots, v_n$. You are given two knapsacks each of integer weight capacity C. If an object is taken, it may be placed in one knapsack or the other, but not both. All weights and values are positive integers. Design an $O(nC^2)$ dynamic programming algorithm that determines the maximum value of objects that can be placed into the two knapsacks. Your algorithm should also determine the contents of each knapsack. Justify the correctness and running time of your algorithm.

Problem 5. (Chain Matrix Multiplication) Let matrices A_1, A_2, A_3, A_4, A_5 have dimensions $8 \times 3, 3 \times 2, 2 \times 19, 19 \times 18$, and 18×17, respectively.
Use the dynamic programming algorithm shown in class to find the least number of scalar multiplications needed to calculate the product $A_1 A_2 A_3 A_4 A_5$. You should show the completely filled in table and also the final way of parenthesizing the product.

Problem 6. (Greedy vs. Dynamic Programming) Consider a country whose coins are minted with positive denominations $D = \{d_1, d_2, \ldots, d_k\}$. The *coin changing problem* is defind as follows: Given an integer n find the $minimum$ number of coins from D that add up to n. You may assume that $d_1 = 1$ so that it is always possible to find some set that adds up to n.

For example, if $D = \{1, 5, 8\}$ and $n = 27$, the best way of making change uses 5 coins, i.e., $\{1, 5, 5, 8, 8\}$.

a. The greedy algorithm for making change repeatedly uses the biggest coin smaller than the amount to be changed until it is zero. Show that the greedy algorithm does not always give the minimum number of coins in a country whose denominations are $\{1, 6, 10\}$.

b. Give an $O(nk)$ time algorithm that correctly determines the minimum number of coins needed to make change of n units using denominations $\{d_1, d_2, \ldots, d_k\}$. Justify the correctness and running time of your algorithm.