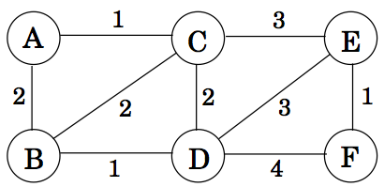
CS 415 Algorithm Analysis **Mid-term 2** **Fall 2014**

Practice problem Solutions (or hints)

1) Shown below is a weighted, undirected graph G = <V,E> on which Kruskal's algorithm is used to find the minimum spanning tree. Exhibit how the data structure for disjoint sets will change after each step of Kruskal's algorithm. (You can ignore path compression while performing FIND.)



initial: each vertex in a set by itself.
Iteration 1: choose the next smallest weight edge, namely (A, C) and perform FIND(A), FIND(C). Since FIND(A) = A and FIND(C) = C and they are different UNION is performed. The result is:
{A, C}, {B}, {D}, {E}, {F}.
Iteration 2: next smallest weight edge is (B,D). Since they belong to two different components, UNION is performed. The result is:
{A, C}, {B,D}, {E}, {F} etc.

I have not shown the sets in the form of trees. You should.

2) Exercise 5.5, 5.9, 5.16, 5.17

5.5: (a) spanning tree does not change. Proof: consider the sorted order of edge weights considered by Kruskal's algorithm. Adding 1 each edge weight does not change the sorted order so the same spanning tree will be produced by Kruskal's algorithm. Since Kruskal's algorithm always produces the minimum spanning tree, the claim follows.

(b) shortest path can change. Consider a very simple example:

V = {1, 2, 3}, E = {(1, 2), (2, 3), (1, 3)} where w(1,2) = 0.5, w(1,3) = 0.5 and w(2, 3) = 1.1. The shortest path from 1 to 3 is 1 ->2 ->3 of weight 1. But if all the weights are increased by 1, the new shortest path from 1 to 3 is 1->3 of weight 2.2. (The path 1->2->3 now weighs 3.)

5. 9: I have not provided any proof. You should. Some of the claims may require knowing Prim's algorithm which we did not cover so I have omitted them.

The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected and connected. Do not assume that edge weights are distinct unless this is specifically stated.

(a) If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

FALSE

(b) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.

TRUE

(c) Let e be any edge of minimum weight in G . Then e must be part of some MST.

TRUE

(d) If the lightest edge in a graph is unique, then it must be part of every MST.

TRUE

(e) If e is part of some MST of G , then it must be a lightest edge across some cut of G.

TRUE

(f) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

(g) The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.

FALSE.

(h) The shortest path between two nodes is necessarily part of some MST.

FALSE

(i) Prim's algorithm works correctly when there are negative edges.

(j) (For any $r > 0$, define an *r-path* to be a path whose edges all have weight $< r$.) If G contains an r-path from node s to t, then every MST of G must also contain an r-path from node s to node t.

5.16. Prove the following two properties of the Huffman encoding scheme.

(a) If some character occurs with frequency more than 2/5, then there is guaranteed to be a codeword of length 1.

(b) If all characters occur with frequency less than 1/3, then there is guaranteed to be no codeword of length 1.

You can show both claims by induction of the number of code symbols.

For (a), Try to show this for k = 4 as the base case. (For k <= 3, it is obvious.) The induction step is now easy. Look at what happens in the first step. It is easy to see that a character of frequency more than 2/5 will never get selected in the first step. (Why not?) If it does, then there are at least two more symbols each with higher frequency. This is a contradiction!

5.17. Under a Huffman encoding of n symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case.

Answer: n-1. It is left for you to construct a set of frequencies. (Make sure to create one for a general n, not for specific n.)

3) Which of the following are true of the longest path problem (from s to t for two specified vertices s and t) in a weighted, directed graph G?

   (a) there is an O(n+m) time algorithm for this problem.

   No. Only if it is a DAG we know how to compute it in O(n+m) time.

   (b) Changing the weight w to −w on each edge will change the problem to the shortest path problem.

   True.

   (c) Adding a fixed constant c to the weight of each edge will leave the shortest path unchanged.

   False. See above.

4) A greedy algorithm for the scheduling problem chooses the most profitable task first, remove all the tasks that overlap with the chosen task(s), and repeat the process. Give an example to show that this algorithm does not produce an optimal solution.

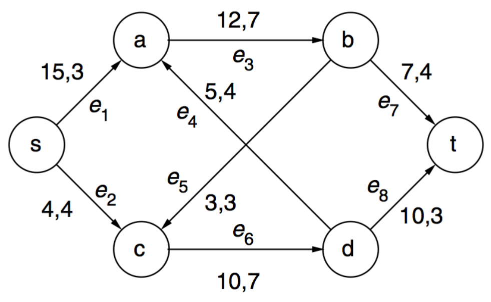
   Example: (1, 4) → 10, (3, 8)→ 15, (7, 10) → 10

5) I have a collection of mp3 files containing n songs of lengths t1, t2, ..., tn. I want to create a CD by recording these songs. But the time for which the CD plays is T which is less than the sum $t_1 + t_2 + ... + t_n$ so I have to leave out some songs. I want to minimize the wasted space on the CD. Formulate this problem as a knapsack problem. (Specifically, assume that there is a function *Knapsack(w[1:n], p[1:n], C)* that has been implemented – show how you can use this function to solve the given problem.)

Use the duration as both profit and weight and set the knapsack capacity as T and call the knapsack(t[1:n], t[1:n, T). The output from this call is the answer to the given problem.

6) Shown below is a graph G with source s = and terminal t = . Also shown is a current flow f. Construct the residual graph G_f. Apply Edmunds-Karp algorithm to find the augmenting path and use it to find an improved flow. Is the new flow optimal?

Left as exercise.



7)

Suppose you're consulting for a company that manufactures PC equipment and ships it to distributors all over the country. For each of the next n weeks, they have a projected *supply* s_i of equipment (measured in pounds), which has to be shipped by an air freight carrier.

Each week's supply can be carried by one of two air freight companies, A or B.

- Company A charges a fixed rate r per pound (so it costs $r \cdot s_i$ to ship a week's supply s_i).
- Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

A *schedule*, for the PC company, is a choice of air freight company (A or B) for each of the n weeks, with the restriction that company B, whenever it is chosen, must be chosen for blocks of four contiguous weeks at a time. The *cost* of the schedule is the total amount paid to company A and B, according to the description above.

Give a polynomial-time algorithm that takes a sequence of supply values $s_1, s_2, \ldots, s_n$ and returns a *schedule* of minimum cost.

Example. Suppose $r = 1$, $c = 10$, and the sequence of values is

$$11, 9, 9, 12, 12, 12, 12, 9, 9, 11.$$

Then the optimal schedule would be to choose company A for the first three weeks, then company B for a block of four consecutive weeks, and then company A for the final three weeks.

Hint: The problem can be represented as the shortest path problem in a directed acyclic graph.

8)

Fred has \$2200 to invest over the next five years. At the beginning of each year he can invest money in one- or two-year time deposits. The bank pays 8 percent interest on one-year time deposits and 17 percent (total) on two-year time deposits. In addition, West World Limited will offer three-year certificates at the beginning of the second year. These certificates will return 27 percent (total). If Fred reinvests his money available every year, formulate a linear program to show him how to maximize his total cash on hand at the end of the fifth year.

Hint: the problem can be formulated as a linear programming problem. We have not discussed the simplex algorithm yet so you can skip this.

9) Which of the following are true of Dijkstra's algorithm?

(i) it requires edge weights to be non-negative.

TRUE.

(ii) it requires edge weights to satisfy triangle inequality.

FALSE.

(iii) adding a constant to each edge will not change the shortest path between any two vertices.

FALSE. (See above.)

(iv) all pairs shortest path can be solved by making n calls to Dijkstra's algorithm.

TRUE.

10) A road map is represented as a weighted directed graph in which each node represents a town and the edges represent roads connecting towns. There are two weights on each edge - <w1, w2> where w1 is the time taken to drive on the road, and w2 is the time taken to travel by bicycle. You are also given two vertices s (home) and t (the campus). You are to design an algorithm that finds the shortest path from s to t in this graph using the following mode of transportation: you have a bicycle at home, you can start riding the bike to some town and then travel by bus the rest of the way to reach the campus. You can exercise the two extreme options as well – ride the bike all the way, or ride the bus the entire way. The goal is to minimize the total time taken to reach t from s. The problem is the following: by making a single call Dijkstra's algorithm (that solves the single source shortest path on a standard graph), solve this variation. Make sure you understand the problem clearly: the standard Dijkstra's algorithm takes as input a graph with one weight on each edge, but the problem at hand has two weights on each edge. You have to create a single new graph G' that uses both weights of G, but G' itself has only one weight on each edge. Further, you can only make a single call to Dijkstra's algorithm.

Create a new graph by making two copies of the given graph where the first part of the graph allows travel by incycle, and the second one by bus. Then make jump from vertex x of first copy to vertex x of second copy with edge of weight 0. Then run Dijkstra's algorithm on this graph.