CMPS 102: Homework #4

Due on Tuesday, April 28st, 2015

John Allard 1437547

Problem 1

Suppose Dijkstra's algorithm is run on the following graph, starting at node A.

DPV41.PNG

1. Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.

   The table below shows each iteration as a row, with later iterations being below earlier iterations. The numbers in each table section represent the shortest distance knows from the source (vertex A) to the vertex corresponding to that column during the iteration given by the row number. The distances to A are always zero because A is the source. For brevity, I use a '.' instead of an ∞ to represent infinite distances.

d(x)	a	b	c	d	e	f	g	h
	0	.	.	.	.	.	.	.
	0	1	.	.	4	8	.	.
	0	1	3	.	4	7	7	.
	0	1	3	4	4	7	5	4
	0	1	3	4	4	7	5	4
	0	1	3	4	4	7	5	4
	0	1	3	4	4	6	5	4
	0	1	3	4	4	6	5	4

2. Show the final shortest-path tree.

Problem 1 continued on next page...

Problem 2

Consider the following graph

1. What is the cost of it's minimum spanning tree?
   The total cost of the minimum spanning tree is 14, it consists of the following edges :
   $$[(A, E, 1), (E, B, 2), (E, F, 1), (F, G, 3), (G, C, 4), (G, H, 3)]$$
2. How many minimum spanning trees does it have?
   By Caley's theorem, there are n^{n-2} spanning trees, given that $n = 8$, the number of spanning trees is $8^6 = 262144$.
3. Suppose Kruskals algortithm is run on this graph. In what order are the edges added to the MST? For each edge in this sequence give a cut that justifies its addition.

   Below is a table that answers the question above. It shows the steps of the algorithm going down the table, with the groups of vertices being shown in the other columns. Notice they all start out as singletons before being formed into a single tree. This example is odd in that no other components of size greater than one were created, all iterations consisted of adding a singleton to the current tree, which does not have to happen with Kruskal's algorithm.

iteration	A	B	C	D	E	F	G	H
1	AE	B	C	D		F	G	H
2	AEF	B	C	D			G	H
3	AEFB		C	D			G	H
4	AEFBG		C	D				H
5	AEFBGH		C	D				
6	AEFBGHC			D				
7	AEFBGHCD							

Problem 3

Often there are multiple shortest paths between two nodes of a graph. How do you need to modify Dijkstras algorithm so that it finds the number of shortest paths

Problem 4

Recall that the correctness proof of Dijkstras algorithm assumes that the weights of the edges are non-negative. Give a small example digraph with some negative edges where Dijkstras algorithm does not find the shortest path.

Problem 5

KT, problem 17, p 197.)
Consider the following variation on the Interval Scheduling Problem.You have a processor that can operate 24 hours a day, every day. People submit requests to run *daily jobs* on the processor. Each such job comes with a *start time* and an *end time*; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. (Note that certain jobs can begin before midnight and end after midnight; this makes for a type of situation different from what we saw in the Interval Scheduling Problem.)

Given a list of n such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Provide an algorithm to do this with a running time that is polynomial in n. You may assume for simplicity that no two jobs have the same start or end times.

Example. Consider the following four jobs, specified by (*start-time, end- time*) pairs.

(6 PM, 6 AM), (9 PM, 4 AM), (3 AM, 2 PM), (1 PM, 7 PM)

The optimal solution would be to pick the two jobs (9 P.M., 4 A.M.) and (1 P.M., 7 P.M.), which can be scheduled without overlapping.

Problem 6

KT, Problem 2, p. 189