MATH 3330 — Applied Graph Theory
Assignment 5 – Solutions

1. Do problem 4.3.7 of the text (find a shortest-path tree for the given graph).

   Here is the tree, in parent notation, and the distance from each vertex to s

s	a	b	c	d	e	f	g
-	c	c	s	e	c	c	f
0	7	9	5	14	13	10	12

   Show your work

   Below are shown all frontier edges in order they are discovered. The first vertex listed is the endpoint that is in the tree. In the second column, the value of $w(e) + dist(u)$ is shown for each frontier edge e, where u is the endpoint of e in V_T. In the third column, the order in which edges are added to E_T is shown. A mark $-$ means that the edge was not a frontier edge anymore when it was considered. 2 points.

Edge	$w(e) + dist(u)$	Order added
sa	7	-
sb	9	-
sc	5	1
ca	7	2
cb	9	3
ce	13	6
cf	10	4
af	11	-
be	15	
bd	15	
fe	13	-
fg	12	5
gd	19	
ge	16	
ed	14	7

2. Show that, when the Dijkstra algorithm is applied to a graph where all edges have the same weight, the result is a BFS tree.

Assume without loss of generality that all edges have weight 1. Both Dijkstra's and BFS are based on the same essential tree-growing algorithm. The only difference is in choosing the next frontier edge to add. For Dijkstra's the criterium is: add the frontier edge $e = uv$, where u is the vertex already in V_T, that minimizes $w(e) + dist(u)$, where $w(e)$ is the weight of edge e, and $dist(u)$ is the distance from u to the root. Since all edge weights are 1, $dist(u)$ equals the graph distance from u to the root to u, and $w(e) = 1$ is the same for all edges. Therefore, the frontier edge added is one that minimizes the graph distance of its endpoint to the root. This also holds for bfs: vertices that are closer to the root are added first to the queue, and its adjacent edges therefore have preference. Since both algorithms in this case have the same criterium for selecting frontier edges, they will give the same tree. 2 points. Note: full marks only if your answer includes some argument about the selection criterium.

3. Problem 4.2.15. Characterize the graphs for which the BFS and DFS trees are identical, no matter what the "tie-breaking" priority or the starting vertex are.

   Trees. Reason: Consider the DFS tree. If there are any edges in the tree that are not part of the graph then they must be "shortcuts" between a vertex and one of its ancestors. But if this same tree is also a BFS tree, then the path through the tree is a shortest path. So no shortcuts are possible. Therefore, there are no edges except those in the tree, and thus the graph itself is a tree. 1 point.

4. (Variation of problem 4.3.9) Similarly to the problem from assignment 3, where MST was applied to finding the most reliable spanning tree, we can use the shortest path principle to find the most reliable connection in a network to a given vertex. Precisely, given a graph (network), and failure probabilities p_{ij}, find the most reliable path between two given nodes s and t.

   The most reliable path is that for which the probability of non-failure is highest. That is, the path for which the product, over all edges ij of the path, of $(1-p_{ij})$ is highest. By taking the logarithm in the negative, we can see that this is the path for which the sum of $-\log(1-p_{ij})$ is lowest. Thus by taking $-\log(1-p_{ij})$ as weights of the edges, we can turn this into a shortest path problem.

Show how to modify Dijkstra's algorithm to solve this problem (give details and justification). Illustrate your method on the graph given in 4.3.9.

First of all, $dist(u)$ *is now the negative log of the reliability of the path from the root to* u *(the product of* $1 - p_{ij}$ *over all edges of the path). Let* $rel(u)$ *denote the relibility of the path from the root to* u *(so* $dist(u) = -\log rel(u)$*). The criterium of Dijkstra's for choosing the next frontier edge to be added is to add the edge that minimizes* $w(e) + dist(u)$*. Considering again that the weigths are negative logarithms, this is equivalent to choosing the frontier edge* uv*, where* u *is the endpoint that is part of the tree, for which* $(1 - p_{uv}) \cdot rel(u)$ *is maximized. 2 points*

5. Find a dfs-tree of the graph shown in 2.4.9. Compute the df numbers and "low" numbers (see class notes and Section 4.4) for each vertex, and use this to find all cut vertices of the graph.

   Your answer will depend on the vertex you started with. This answer is for root a*, and used alphabetic ordering to resolve ties.*

Vertex	a	b	c	d	s	t	u	v	x	y	z
Parent	-	a	d	c	z	s	c	u	v	x	x
Df-number	0	1	3	2	9	10	4	5	6	7	8
low-number	0	0	1	0	9	10	1	5	6	5	8

   The root a *has only one child, so is not a cut vertex. The vertices that have df-number equal to the low number are* a, s, t, v, x, z*. The cut vertices are precisely the parents of these vertices in the df tree, which are* z, s, u, v, x*. 1 point*

6. True or false: the diameter of a graph is the maximum depth of a dfs tree of the graph.

   False. Diameter is about shortest paths. Counterexample: the complete graph. Any dfs tree will be a path, but diameter is one. 1 point

7. Follow the link given on the course Web page labelled "wire routing". The given applet finds a path from a start square to finish square in a grid with blockages. The graph it represents is as follows: the squares are its vertices, and edges correspond to adjacent squares. The applet

uses one of the tree-growing algorithms discussed in class. Which one? Explain your answer.

BFS. At each iteration, the algorithm adds all frontier edges. This means it uses a queue-type structure, and ends up with a bfs tree. Note that it therefore also finds the shortest path from start to finish square. 1 point