University of Washington

Department of Computer Science and Engineering

CSE 373

Problem Set 7 Solution

1. (3 points) Dijkstra's Simulation

V	known	D_v	P_v
A	1	0	-
B	0	5	A
C	0	3	A
D	0	-	-
E	0	-	-
F	0	-	-
G	0	-	-

V	known	D_v	P_v
A	1	0	-
B	0	5	A
C	1	3	A
D	0	10	C
E	0	10	C
F	0	-	-
G	0	-	-

V	known	D_v	P_v
A	1	0	-
B	1	5	A
C	1	3	A
D	0	10	C
E	0	8	B
F	0	-	-
G	0	6	B

V	known	D_v	P_v
A	1	0	-
B	1	5	A
C	1	3	A
D	0	10	C
E	0	7	G
F	0	-	-
G	1	6	B

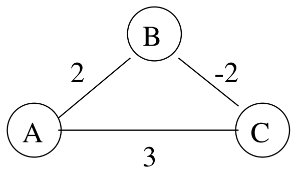
V	known	D_v	P_v
A	1	0	-
B	1	5	A
C	1	3	A
D	0	9	E
E	1	7	G
F	0	8	E
G	1	6	B

V	known	D_v	P_v
A	1	0	-
B	1	5	A
C	1	3	A
D	0	9	E
E	1	7	G
F	1	8	E
G	1	6	B

V	known	D_v	P_v
A	1	0	-
B	1	5	A
C	1	3	A
D	1	9	E
E	1	7	G
F	1	8	E
G	1	6	B

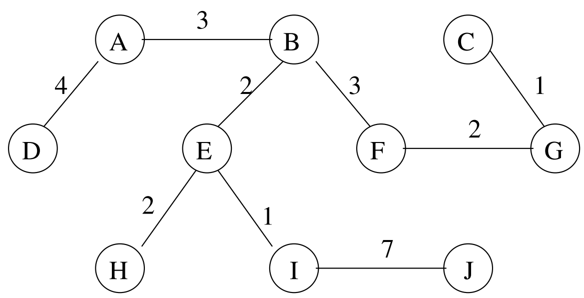
2. (2 points) Dijkstra's Counterexample

   Consider the below with vertex A as the start vertex.



3. (2 points) Find the MCST

   (a) One possible spanning tree is given below.



   (b) This tree is not unique; for example, we can remove edge BF and insert FI instead.

4. (2 points) Negative Edges and MCST's

   Both algorithms work with negative-weight edges. The proof makes no assumptions about the edge weights except that they can be sorted.

5. (1 point) Maximum-cost Spanning Trees

   Finding a maximum-cost spanning tree is virtually identical to finding a minimum-cost spanning tree. Replace "min" with "max" and "less than" with "greater than" everywhere in the algorithm. (Or, if you can't change the algorithm code, preprocess the graph by replacing each edge weight with its negative.)

6. (6 points) Pick-Up Sticks

   (a) Once one has taken the initial geometry and converted into *above* relationships, the problem is trivially converted into a topological sort. Create a digraph as follows: create a vertex for each stick. For each relation *above(u, v)*, add a directed edge from u to v; this represents the fact that u must be picked up before v. One can input this graph into the computer by using a standard edge-list representation.

   (b) Topological sort.

   (c) No modifications are necessary.