Graphs Practice Problems - Solutions

P1. Graph representation – space calculation. Solve the Student Self Study problem from Graphs presentation: Given an undirected graph with 10 million vertices, where each vertex has exactly 20 neighbors, what is the space (in Bytes) needed for each of these representations: Adjacency List, Adjacency Matrix ?

Assume these sizes: memory address 8B, integer 8B, char 1B .

Assume:

- a node in the adjacency list uses an int for the neighbor and a pointer for the next node.
- the Adjacency matrix can be implemented as a 2D matrix of bits ()

(You should be able to solve such problems for other sizes, e.g. 4B for an int, or using a 2D of int for the adjacency table.)

See Slides

P2. Give one application for each graph algorithm that we studied: DFS, BFS, topological sorting, MST (Minimum Spanning Tree), SPST (Shortest Path Spanning Tree), all-pairs shortest paths.

DFS used in topological sorting

BFS, SPS, see pb 3 below,

Topological sorting – order to take courses so prerequisites are completed before each course.

MST – smallest total cost to build roads to connect all cities without any cycle.

P3. a) Know usage of other data structures in graph algorithms. For each of the given data structures, list at least one algorithm that uses it, and explain what for (how does it benefit from using that data structure): list, queue, priority queue.

b) What graph algorithms discussed in class used recursion?

Review lectures or slides to find the answers

P4. What graph algorithm would you use to find:

a) cheapest flight – **SPST (Dijkstra)**
b) flight with fewest stops - **BFS**

P5. The SPST(G,0) algorithm generated the arrays d and p below (they are listed as a table to 'reuse' the indexes).

Indices	0	1	2	3	4	5	6	7
d	0	10	13	14	11	16	18	15
p	-1	0	4	2	1	2	7	3

What is the shortest distance between 0 and 5? Give vertices on the shortest path from 0 to 5:

Distance: **16** path: 0 -> **1 -> 4 -> 2** -> 5

P6. In class we looked at the "Telephone Network" application discussed in the link below. What are the changes/adaptation done Dijkstra's algorithm in order to solve this problem?
http://www.csl.mtu.edu/cs2321/www/newLectures/30_More_Dijkstra.htm

- What is the definition of the "distance" between 2 vertices in this case?
- What is the distance for the source vertex? (BW(v) in their notation)
- What type of heap is used?
- What is the condition used to decide if the distance for a particular vertex (BW[z]) should be updated?

See webpage

P7. Be able to continue the work for MST and Dijkstra. E.g. Given the adjacency list, and the table that shows dist/parent after a few iterations, execute one more iteration.

In this adjacency list, the first value in a node is the destination vertex and the second value is the weight of that edge. E.g. the first line gives edge (0,1) with weight 23 and edge (0,2) with weight 45.

0: (1,23), (2,45) ,
1: (0,23), (2,11), (4,63)
2: (0,45), (1,11), (3,7), (4,80)
3: (0,50), (2,7)
4: (1,63), (2,80)

Index	0	1	2	3	4
dist	23	63	11	inf	0
				7	

Assume vertices 1 and 4 are in the MST and the distance array, dist, updated after both vertices were selected is shown above. Which vertex will be picked next? 2
Update the table. (Show only the modified cells) See last row in the above table.

In Canvas: give the pairs (index, updatedDistance) in increasing order of the indices and with pairs separated by a comma. Do not put any spaces, e.g. (0,5),(1,25),(4,37) ***(3,50)***

P8. If you are given only the finish order for the vertices of an acyclic graph traversed with Depth-First Search (DFS), CAN you list the graph vertices in **topological order**? Assume there were no cycles.
In the table below, finish for vertex 5 is 1, meaning 1 finished first, and finish of vertex 6 is 4, meaning that 6 was the 4-th vertex to finish.

If Yes, **list the vertices** in topological order.

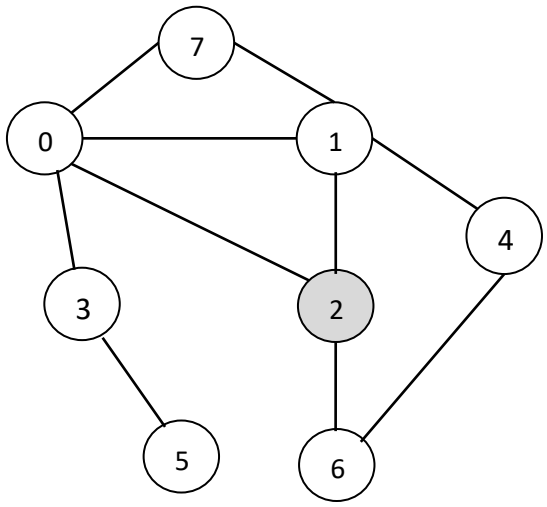
Yes: 2, 3, 0, 6, 4, 1, 5

If No, justify **briefly**.

Vertex	0	1	2	3	4	5	6
Finish	5	2	7	6	3	1	4

P9. Traverse the graph below in **Breadth-First** order**, starting from vertex 2**. Whenever you examine neighbors of a vertex, process them in **increasing order** of vertex number. Half the points will be lost if you visit the neighbors in a different order.

Fill in the table below (the vertices, edges and the distance).



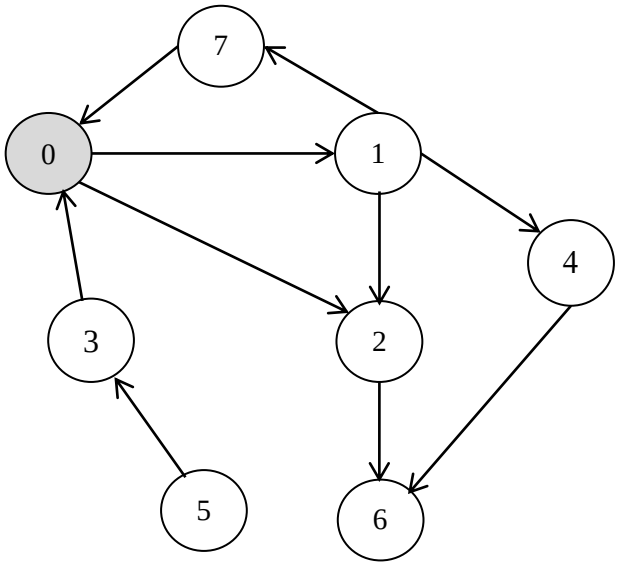
Queue: 2, 0, 1, 6, 3, 7, 4, 5

Order from first to last edges added to the BFS tree	Vertex, v, (in order visited by BFS)	Edge: u, v (edge that adds v to the BFS tree)	Distance (as number of edges) from the source
1st	2	-	0
2nd	0	(2,0)	1
…	1	(2,1)	1
	6	(2,6)	1
	3	(0,3)	2
	7	(0,7)	2
	4	(1,4)	2
	5	(3,5)	3

P10. Traverse this graph with Depth First Search (DFS) (all DFS, not just one DFS-Visit).

a) If a vertex has more than one neighbors **you must visit them in increasing order** of vertex number**,** for example from node 1 you would visit the neighbors in order: 2,4,7. Half the points will be lost if you visit the neighbors in a different order.

c) Label the edges as one of these 3 categories: **T** (tree), **B** (backward), **C/F**.



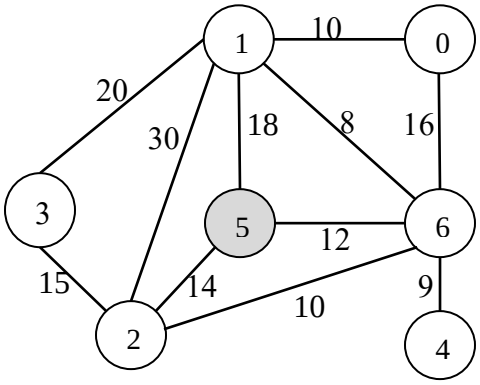
Order from first to last	Visited vertex	Pred
1st	**0**	-
2nd	1	0
…	2	1
	6	2
	4	1
	7	1
	3	-
	5	-

Tree edges: (0,1),(1,2),(2,6),(1,4),(1,7)

Backward edges: (7,0)

C/F: (0,2),(3,0),(5,3),(4,6)

P11. Run **Prim's algorithm** on this graph, to produce the minimum spanning tree (MST) **starting from vertex 5**. **Fill out the table below with edges** in the order in which they are selected by the algorithm and added to the MST. For each edge list the vertices and its weight, e.g. (2,5,14)



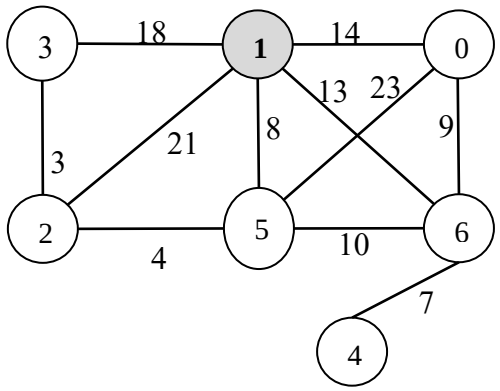
Order from first to last	Edge: u, v, weight
1st	5,6,12
2nd	6,1,8
…	6,4,9
	1,0,10
	6,2,10
	2,3,15

Fill in the dist/parent table as in class. **Show if the node is in the MST tree or not.** The third piece of information (MST) for each vertex v is either N or T, where N means the vertex is not in the MST and T means it is in the MST. E.g. see that in the first row we have T for vertex 5 meaning 5 is in the MST and all the other vertices were NOT in the MST at that point. The second row show that 6 is now in the MST.

In case of a tie, select the smaller vertex. E.g. below, when both 0 and 2 had distance 10, vertex 0 was selected.

Index	0	1	2	3	4	5	6	Vertex added to MST
d/p/MST	Inf/-1/N	Inf/-1/N	Inf/-1/N	Inf/-1/N	Inf/-1/N	0/-1/T	Inf/-1/N	5
		18/5/N	14/5/N				12/5/T	6
	16/6/N	8/6/T	10/6/N		9/6/N			1
	10/1/N			20/1/N	9/6/T			4
	10/1/T							0
			10/6/T					2
				15/2/T				3

P12. (10 points) Run Dijkstra's algorithm for Single Source Shortest Path (SPST) **starting from vertex 1** in the graph below. Process neighbors of a vertex in **increasing order** of the vertex value**.** Print the vertices and their distance from vertex 1, in the order in which they are added to the tree. (Remember that it is looking for the shortest path from vertex 1 to any other vertex.)



Order from first to last vertex added to the SPST tree	Vertex, v, (in order added to the SPST)	Distance from vertex 1 to vertex v.	Pred
1st	**1**	0	-
2nd	5	8	1
…	2	12	5
	6	13	1
	0	14	1
	3	15	2
	4	20	6

Fill in the dist/parent table as in class**. Show if the node is in the SPST tree or not.** The third piece of information (SPST) for each vertex v is either N or T, where N means the vertex is not in the SPST and T means it is in the SPST.

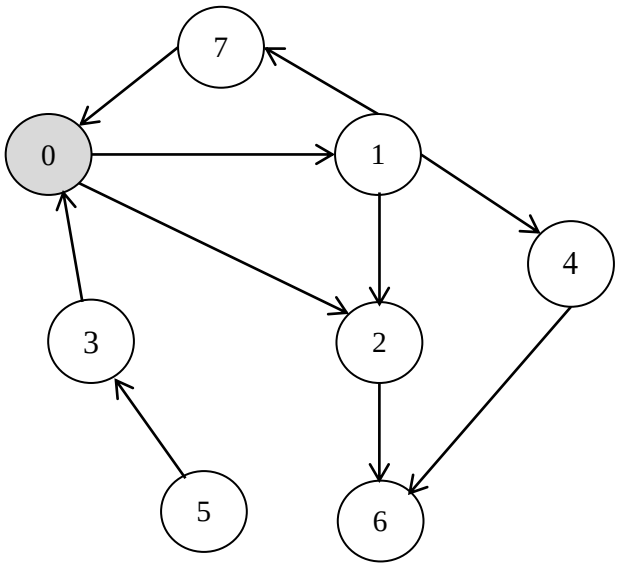
Index	0	1	2	3	4	5	6	Vertex added to SPST
d/p/SPST	Inf/-1/N	0/-1/T	Inf/-1/N	Inf/-1/N	Inf/-1/N	Inf/-1/N	Inf/-1/N	1
	14/1/N		21/1/N	18/1/N		8/1/T	13/1/N	5
			12/5/T					2
				15/2/N			13/1/T	6
	14/1/T				20/6/N			0
				15/2/T				3
					20/6/T			4

Extra

P13. This is the same as problem 10, but includes start and finish times. Traverse this graph with Depth First Search (DFS) (start at vertex 0) and:

a) Fill in the **start** and **finish** times, **predecessor node**, pred, in the table below. Visit neighbors in **increasing order,** for example from node 1 you would visit the neighbors in order: 2,4,7.

c) Label the edges as one of these 3 categories: **T** (tree), **B** (backward), **C/F**.



Order from first to last	Visited vertex	Start	Finish	Pred
1st	**0**	1	12	-
2nd	1	2	11	0
…	2	3	6	1
	6	4	5	2
	4	7	8	1
	7	9	10	1
	3	13	14	-
	5	15	16	-

Tree edges: (0,1),(1,2),(2,6),(1,4),(1,7)

Backward edges: (7,0)

C/F: (0,2),(3,0),(5,3),(4,6)