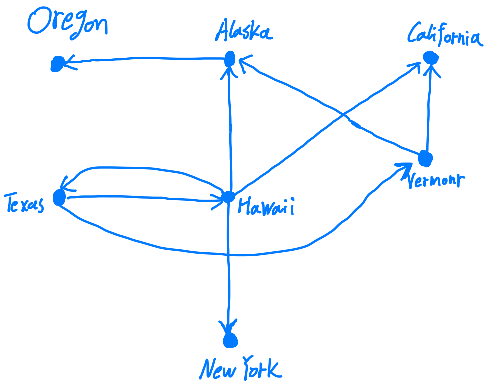
CMSC204

Kartchner

V(StateGraph) = {Oregon, Alaska, Texas, Hawaii, Vermont, NewYork, California}
E(StateGraph) = {(Alaska, Oregon), (Hawaii, Alaska), (Hawaii, Texas), (Texas, Hawaii), (Hawaii, California), (Hawaii, New York), (Texas, Vermont), (Vermont, California), (Vermont, Alaska)}

1. Draw the StateGraph



1. Describe the graph pictured above, using the formal graph notation.

   V(StateGraph) =

   E(StateGraph) =

2. a. Is there a path from Oregon to any other state in the graph? No

   b. Is there a path from Hawaii to every other state in the graph? Yes

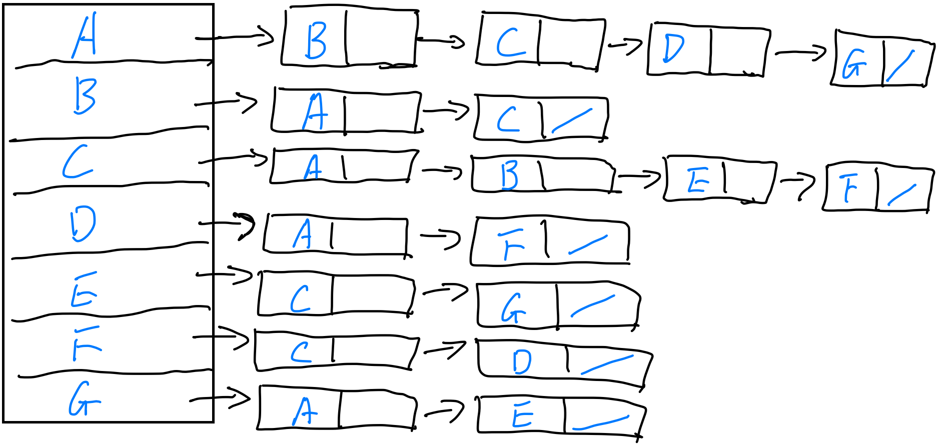
   c. From which state(s) in the graph is there a path to Hawaii? Texas

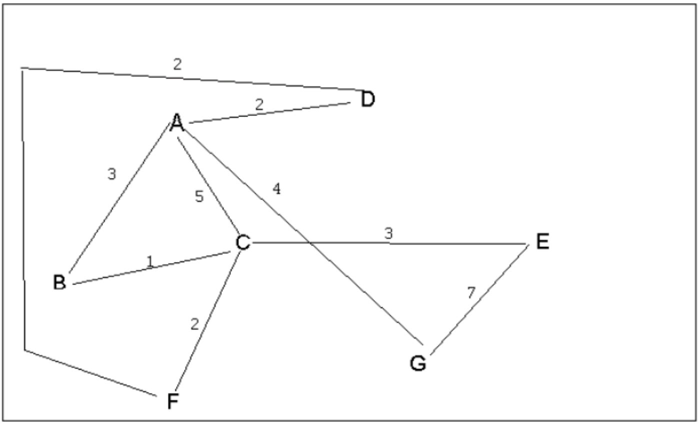
1. a. Show the adjacency matrix that would describe the edges in the graph. Store the vertices in alphabetical order

States

Alaska	0	0	0	0	1	0	0
California	0	0	0	0	0	0	0
Hawaii	1	1	0	1	0	1	0
New York	0	0	0	0	0	0	0
oregon	0	0	0	0	0	0	0
Texas	0	0	1	0	0	0	1
Vermont	1	1	0	0	0	0	0

3. b. Show the adjacency lists that would describe the edges in the graph



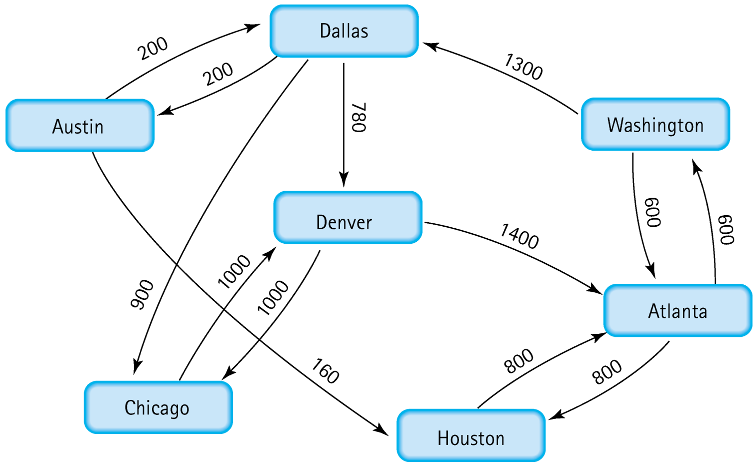


4 a. Which of the following lists the graph nodes in depth first order beginning with E?

A) E, G, F, C, D, B, A
B) G, A, E, C, B, F, D
[✓ C) E, G, A, D, F, C, B]
D) E, C, F, B, A, D, G

4 b. Which of the following lists the graph nodes in breadth first order beginning at F?

A) F, C, D, A, B, E, G
B) F, D, C, A, B, C, G
C) F, C, D, B, G, A, E
[✓ D) a, b, and c are all breadth first traversals]



5. Find the shortest distance from Atlanta to every other city

Atlanta to Washinton : 600

Atlanta to Houston : 800

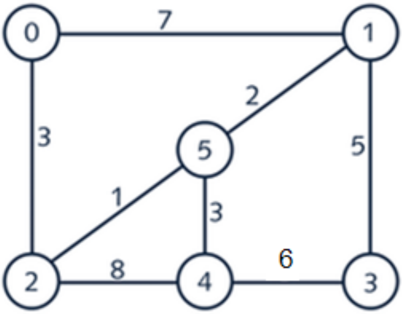
Atlanta to Dallas : 600+1300 = 1900

Atlanta to Denver : 600+1300+780 = 2680

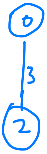
Atlanta to Austin : 600+1300+200 = 2100

Atlanta to Chicago : 600+1300+900 = 2800

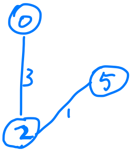
6. Find the minimal spanning tree using Prim's algorithm. Use 0 as the source vertex . Show the steps.



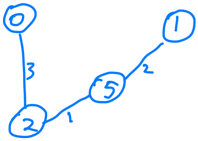
1:



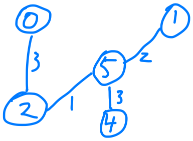
2:



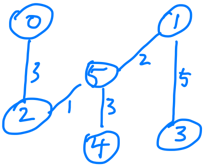
3:



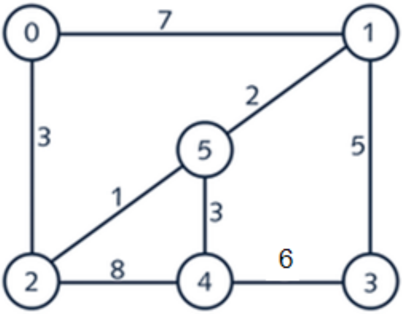
4:



5:

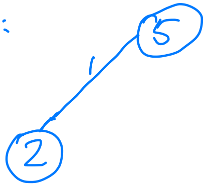


7. Find the minimal spanning tree using Kruskal's algorithm. Show the weights in order and the steps.



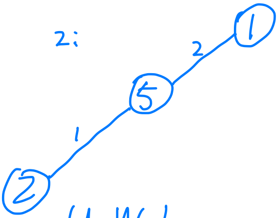
Weight_List= {1,2,3,4,5,6,7,8}

1:



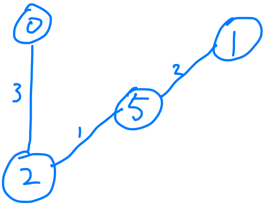
Updated Weight_List = {2,3,3,5,6,7,8}

2:



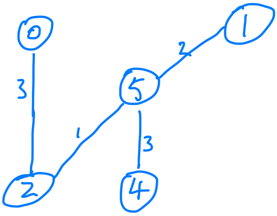
U.W_L = {3,3,5,6,7,8}

3:



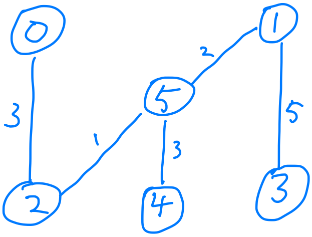
U.W_L = {3,5,6,7,8}

4:



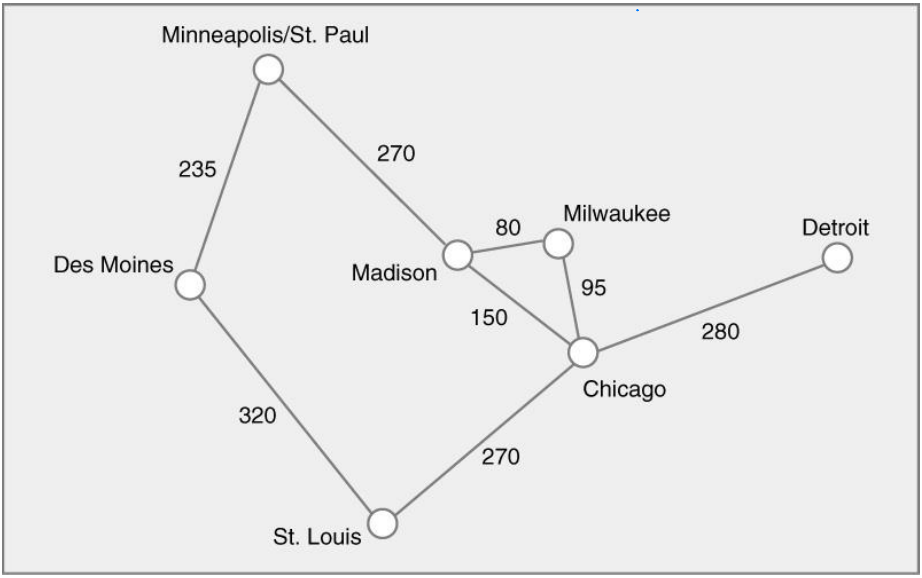
U.W_L = {5,6,7,8}

5:



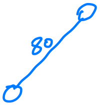
U.W_L = {6,7,8}

8. Find the minimal spanning tree using the algorithm you prefer. Use Minneapolis/St. Paul as the source vertex



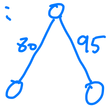
Weight_List = {80, 95, 150, 235, 270, 270, 280, 320}

1:



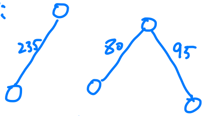
U.W.L
= {95, 150, 235, 270, 270, 280, 320}

2:



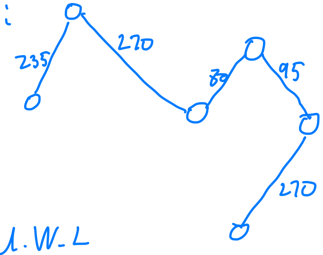
U.W.L
= {150, 235, 270 270, 280, 320}

3:



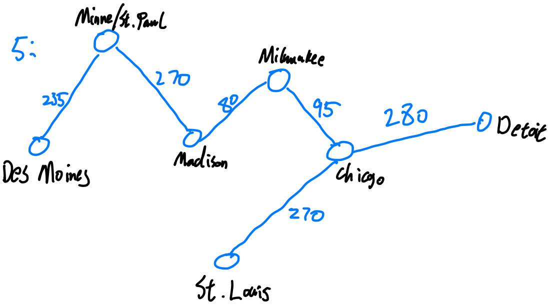
U.W.L
= {270, 270, 280, 320}

4:

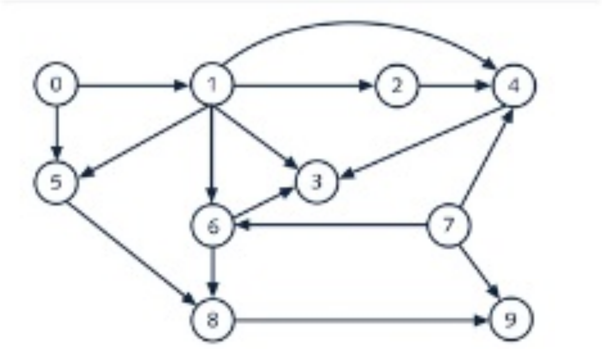


U.W.L
= {280, 320}

5:



9. List the nodes of the graph in a breadth first topological ordering. Show the steps using arrays predCount, topologicalOrder and a queue



1: Predcount[ ]

0	1	2	3	4	5	6	7	8	9
0 ✓	1	1	3	3	2	2	0 ✓	2	2

queue

0	7		

↑f

remove from f place in topological [ ]

0									

Successor node '0' are 1,5

Predcount[1] = Predcount[1]-1

Predcount[5] = Predcount[5]-1

2: Predcount[ ]

0	1	2	3	4	5	6	7	8	9
0 ✓	0	1	3	3	1	2	0	2	2
	decrement								

Queue

7	1				
↑ front					

t_sort

0	7								

Successors of node 7 are: 4,6,9

~~Succe~~ Predcount of node: 4,6,9.

3: Predcount[ ]

0	1	2	3	4	5	6	7	8	9
0	0	1	3	2 ✓	1	1 ✓	0	2	1 ✓

← decremented

queue :

1					
↑ front					

t_sort

0	7	1							

successors of node 1: 2,3,4,5,6

Decrement Predcount[ ] of node 2,3,4,5,6

4:

0	1	2 ✓	3	4	5 ✓	6 ✓	7	8	9
0	0	0	2	1	0	0	0	2	1
		↑	↑	↑	↑	↑			

Dercrement

queue

2	5	6			

t_Sort

Successor of node: 2 : 4

0	7	1	2						

5:

0	1	2	3	4	5	6	7	8	9
0	0	0	2	0	0	0	0	2	1
				↑					

Queue

5	6	4			
↑ front					

Successor of node: 8

t Sort

0	7	1	2	5					

6: Predcount[ ]   decrement Predcount[8]

0	1	2	3	4	5	6	7	8	9
0	0	0	2	0	0	0	0	1	1

Successor of node 6: 3,8

Queue

6	4				

t_Sort

0	7	1	2	5	6				

7: Predcount[ ]

0	1	2	3	4	5	6	7	8	9
0	0	0	1	0	0	0	0	0	1
			↑					↑	

Queue

4	8				

t_Sort

Succ of node 4: 3

0	7	1	2	5	6	4			

8: Predcount[3]

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	1

Queue

8	3				

t_Sort

Succ of node 8: 9

0	7	1	2	5	6	4	8		

9: Predcount

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

Queue

3	9				

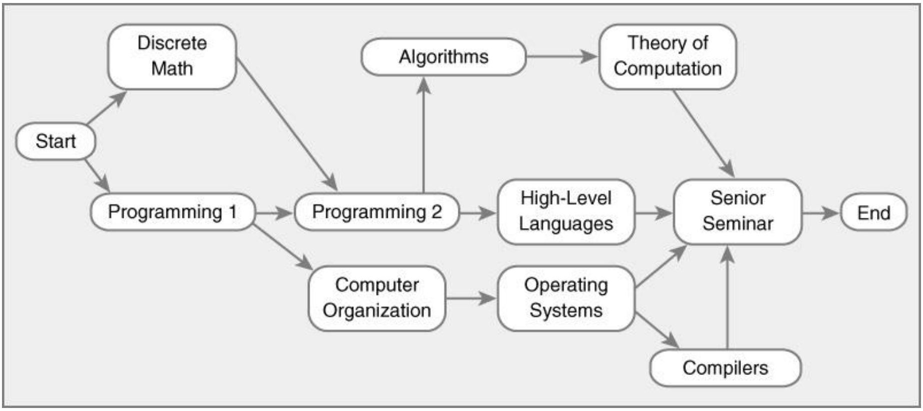
Final Sort

T_Sort

0	7	1	2	5	6	4	8	3	9

Seccesor of 3 and 9 all null.

10. List the nodes of the graph in a breadth first topological ordering.



Start

Discrete Math

Programming 1

Programming 2

Computer Organization

Algorithms

High-Level Languages

Operating Systems

Theory Computation

Senior Seminar

Compilers

End