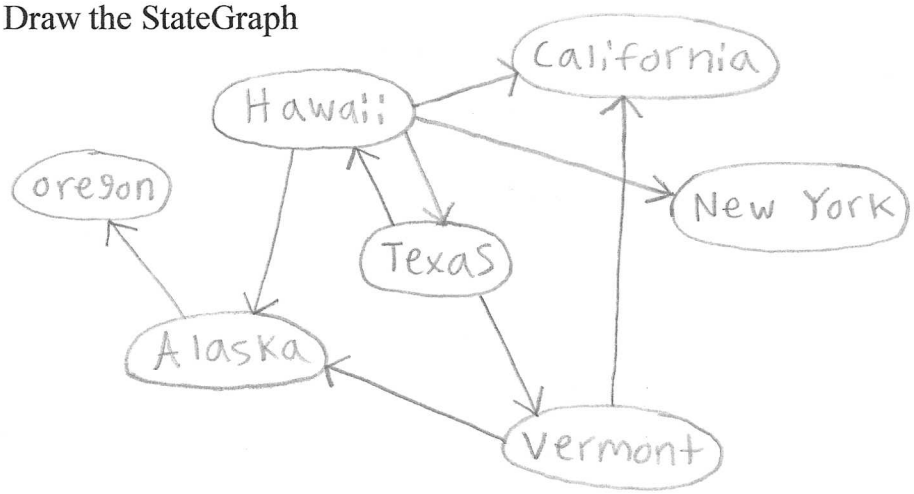
Kevin Villegas

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

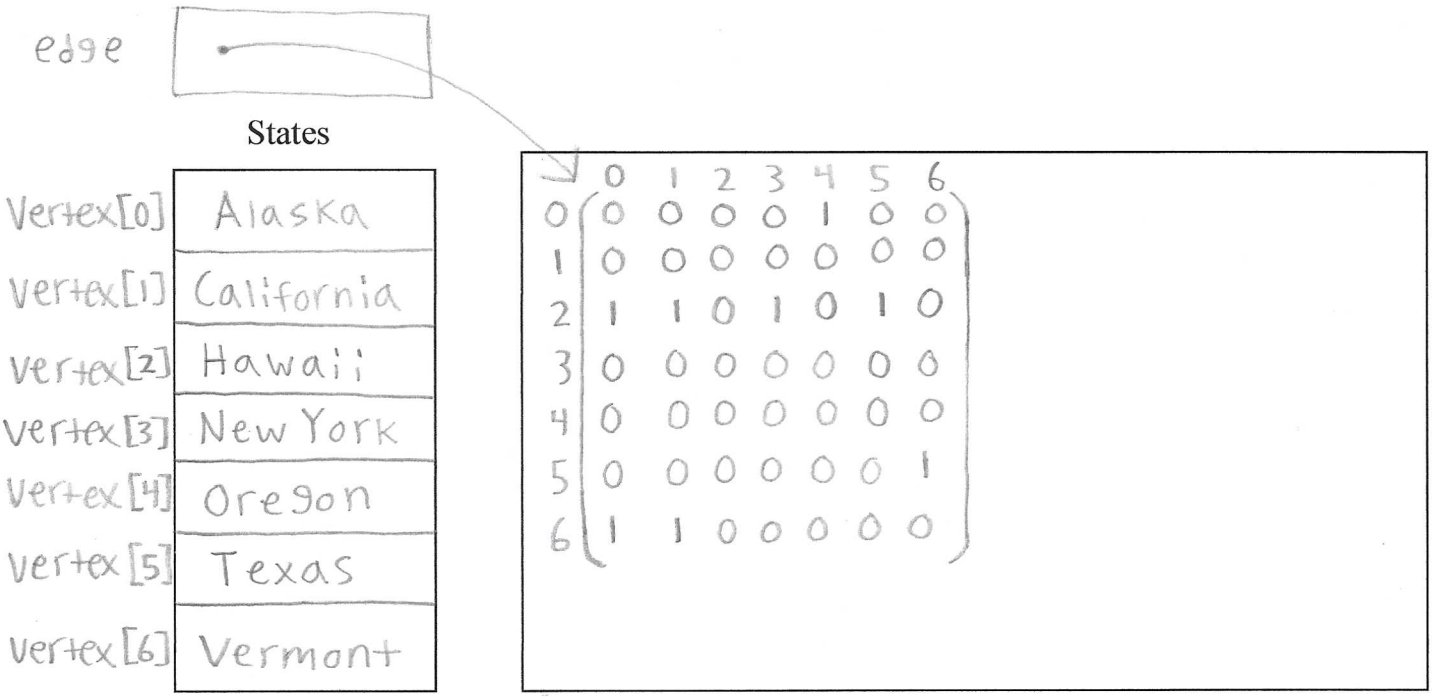
   b. Is there a path from Hawaii to every other state in the graph?

   Yes

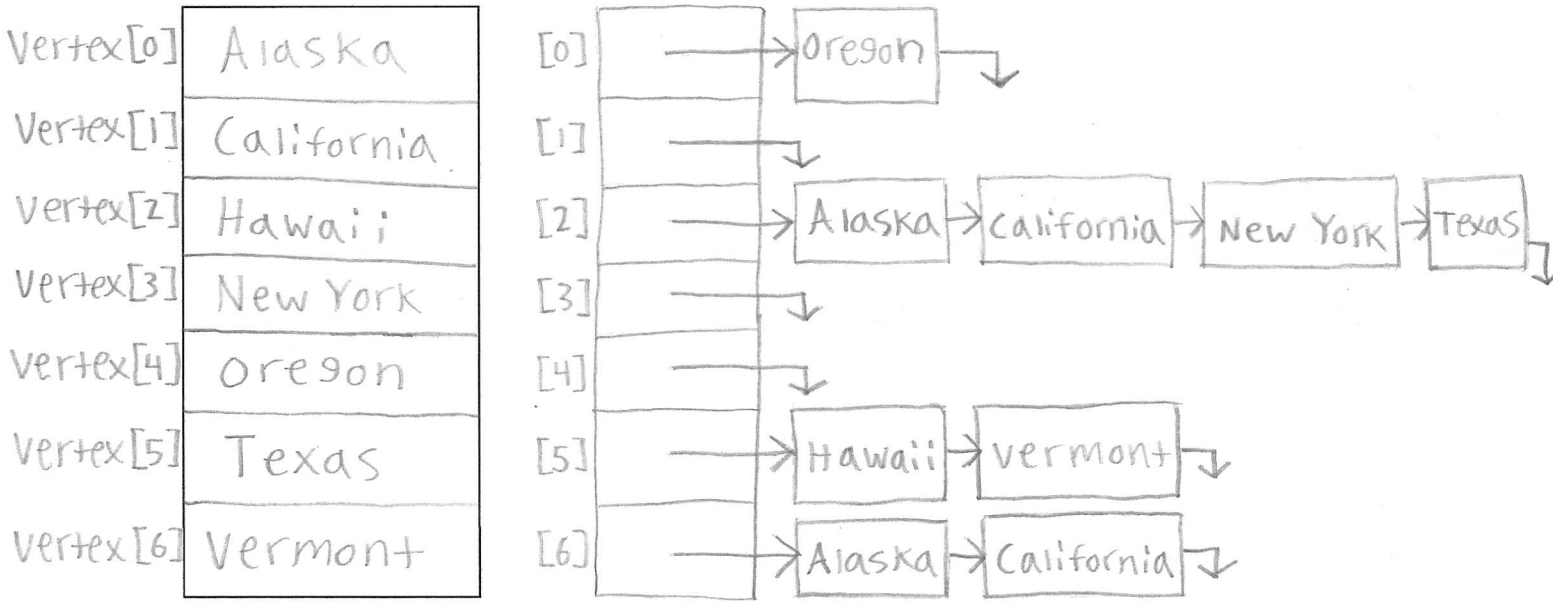
   c. From which state(s) in the graph is there a path to Hawaii?

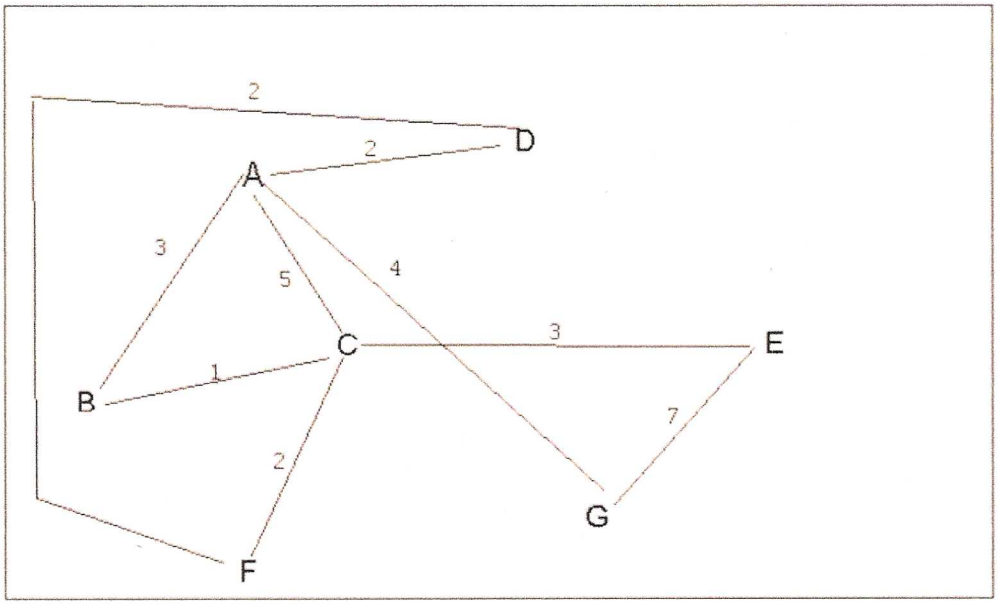
   Texas

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



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



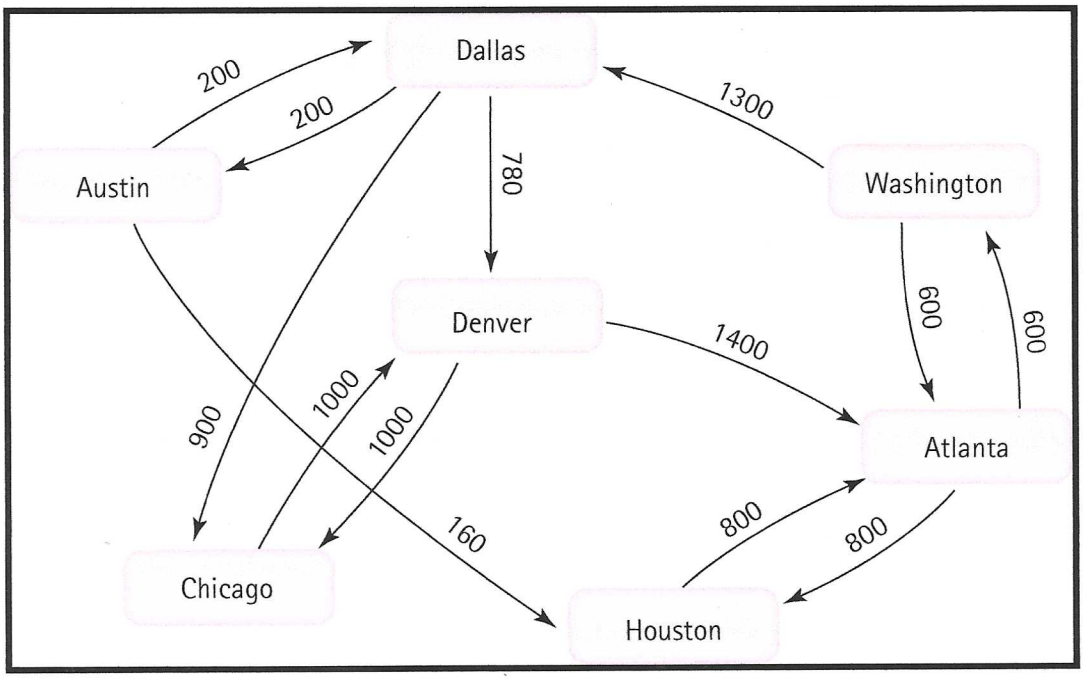


4 a. Which of the following lists the graph nodes in depth first order beginning with E?

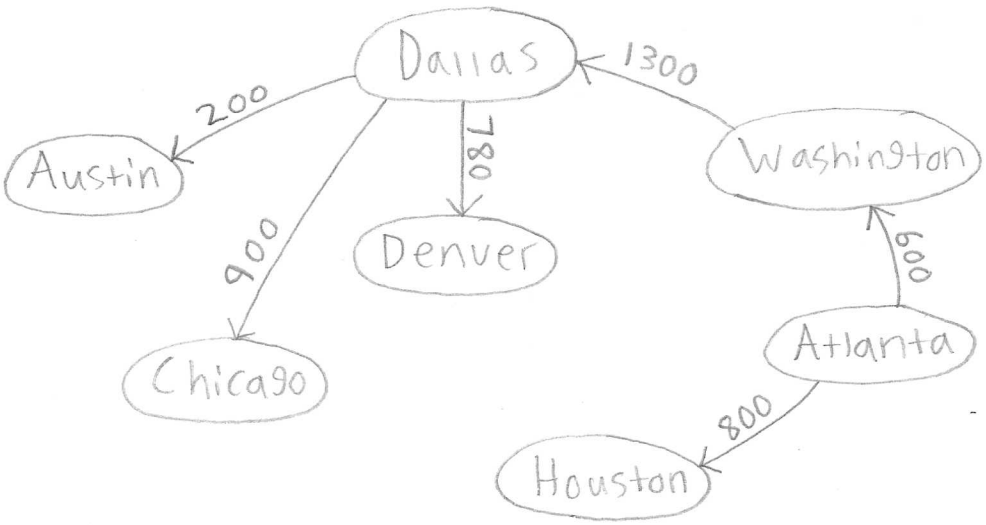
A) E, G, F, C, D, B, A
B) G, A, E, C, B, F, D
C) E, G, A, D, F, C, B
D) E, C, F, B, A, D, G

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

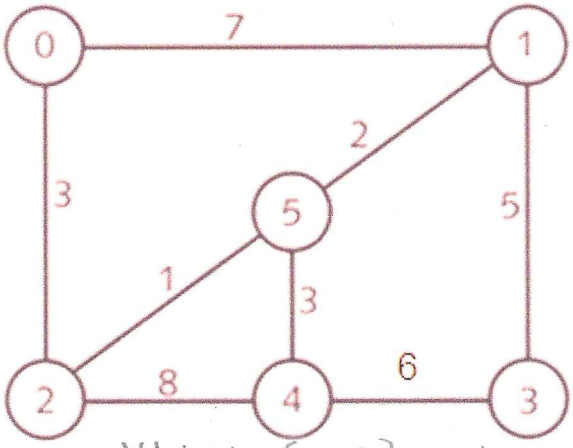


5. Find the shortest <u>distance</u> from Atlanta to every other city

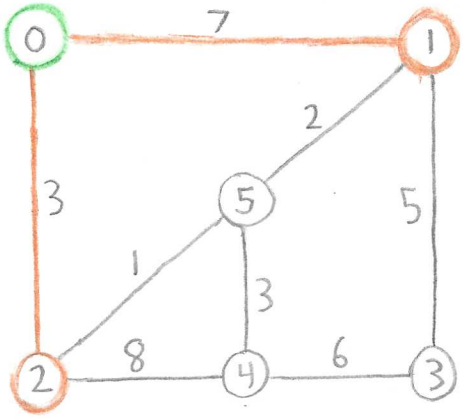


Atlanta → Houston = 800
Atlanta → Washington = 600
Atlanta → Washington → Dallas = 1900
Atlanta → Washington → Dallas → Denver = 2680
Atlanta → Washington → Dallas → Austin = 2100
Atlanta → Washington → Dallas → Chicago = 2800

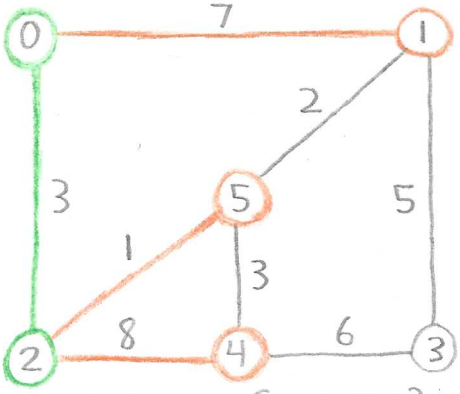
6. Find the minimal spanning tree using Prim's algorithm. Use 0 as the source vertex . Show the steps.



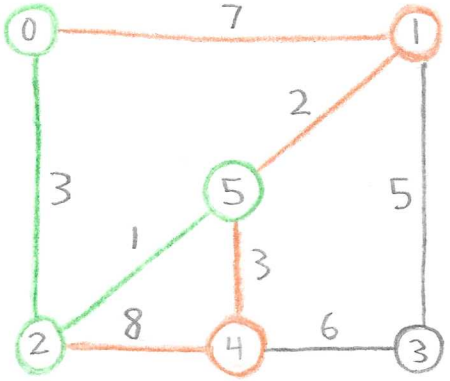
Visited = {0}



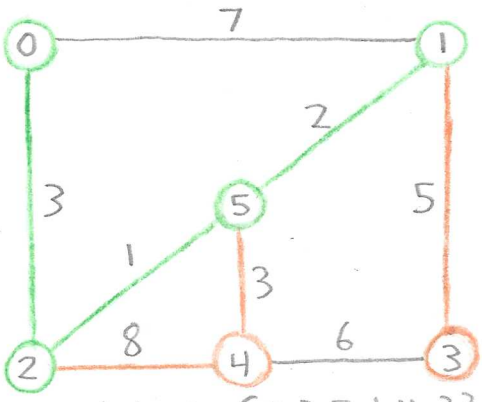
Visited = {0, 2}



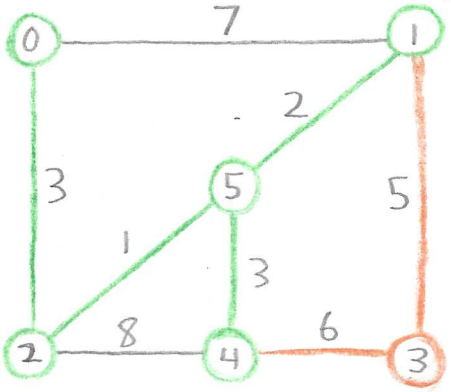
Visited = {0, 2, 5}



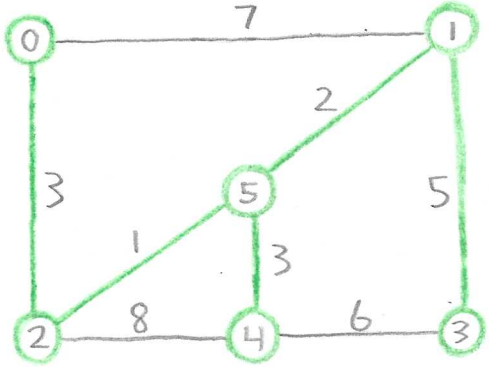
Visited = {0, 2, 5, 1}



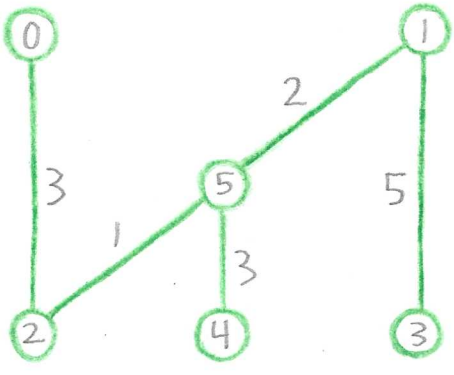
Visited = {0, 2, 5, 1, 4}



Visited = {0, 2, 5, 1, 4, 3}



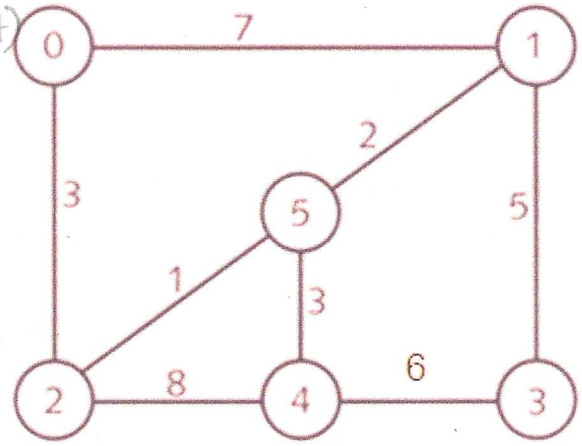
Minimal Spanning Tree



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

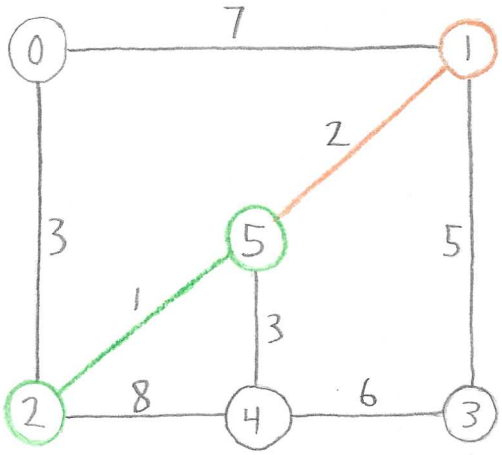
Sorted Edge List

Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)



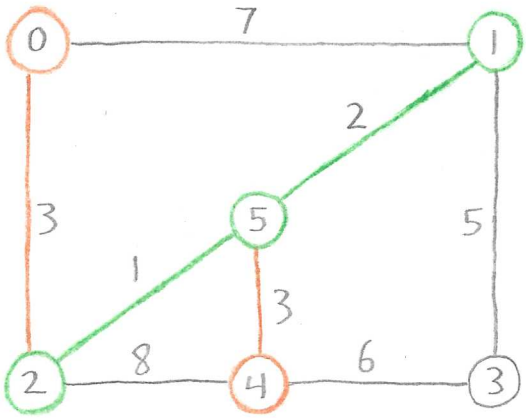
Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)

Visited = {2, 5}



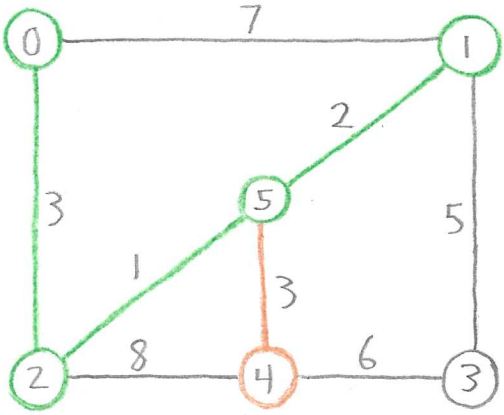
Visited = {2, 5, 1}

Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)



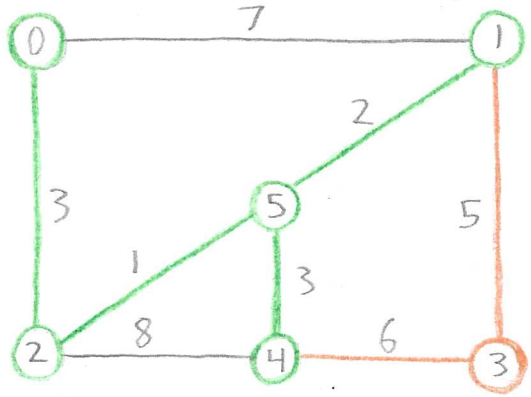
Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)

Visited = {2, 5, 1, 0}



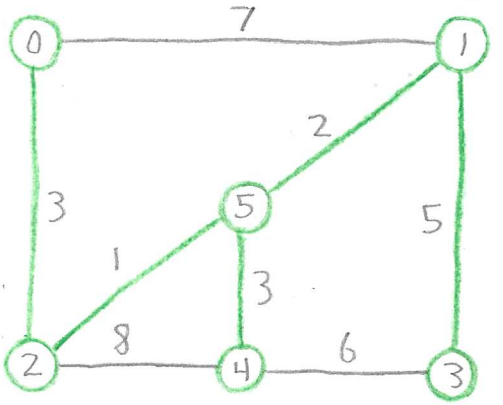
Visited = {2, 5, 1, 0, 4}

Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)

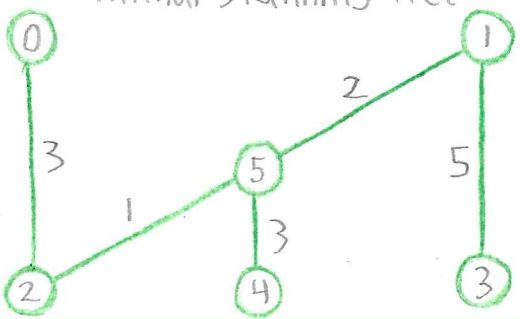


Edge (weight)
2-5 (1)
1-5 (2)
0-2 (3)
4-5 (3)
1-3 (5)
3-4 (6)
0-1 (7)
2-4 (8)

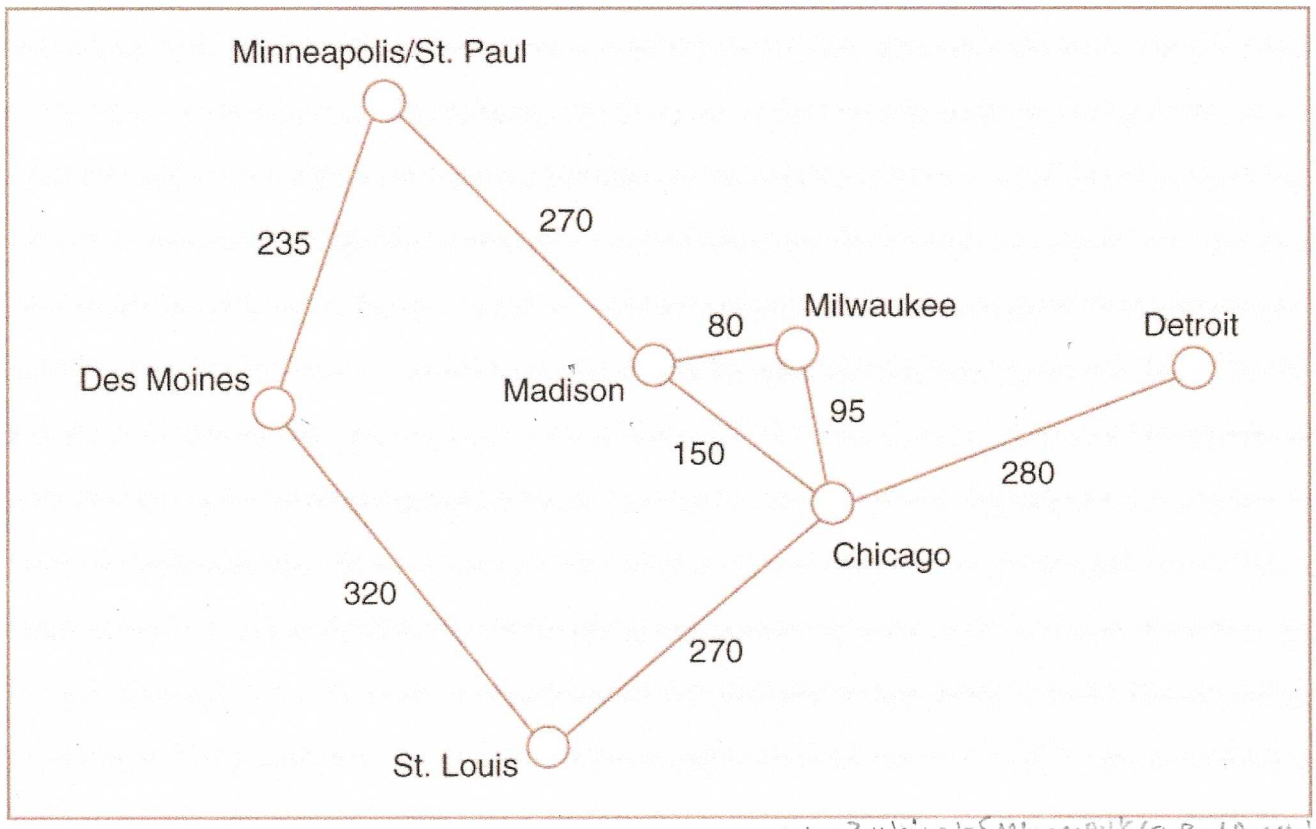
Visited = {2, 5, 1, 0, 4, 3}



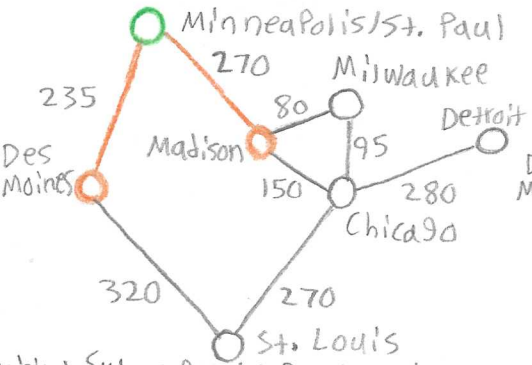
Minimal Spanning Tree



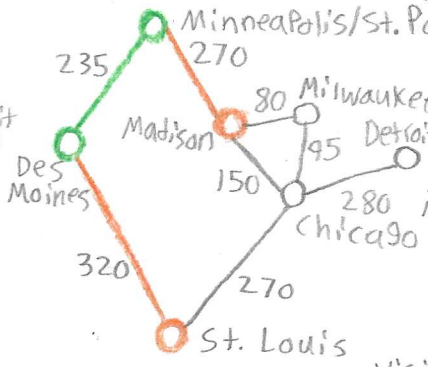
8. Find the minimal spanning tree using the algorithm you prefer. Use Minneapolis/St. Paul as the source vertex



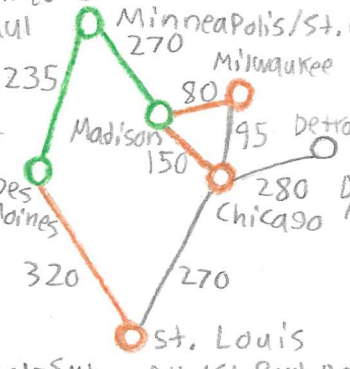
Visited={Minneapolis/St. Paul}



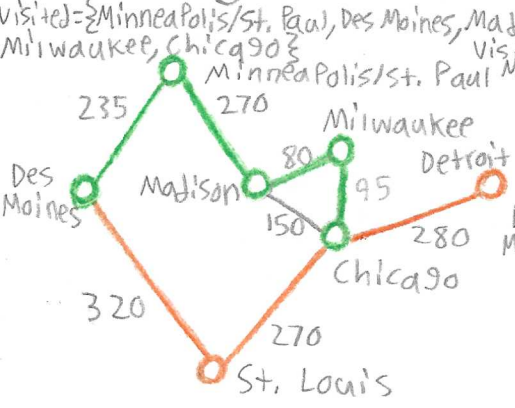
Visited={Minneapolis/St. Paul, Des Moines}



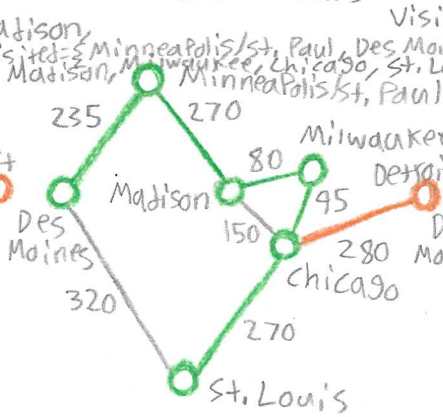
Visited={Minneapolis/St. Paul, Des Moines, Madison, Milwaukee}



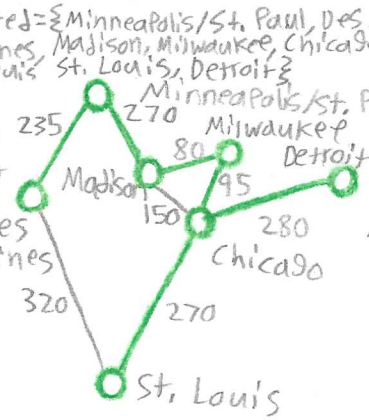
Visited={Minneapolis/St. Paul, Des Moines, Madison, Milwaukee, Chicago}



Visited={Minneapolis/St. Paul, Des Moines, Madison, Milwaukee, Chicago, St. Louis}

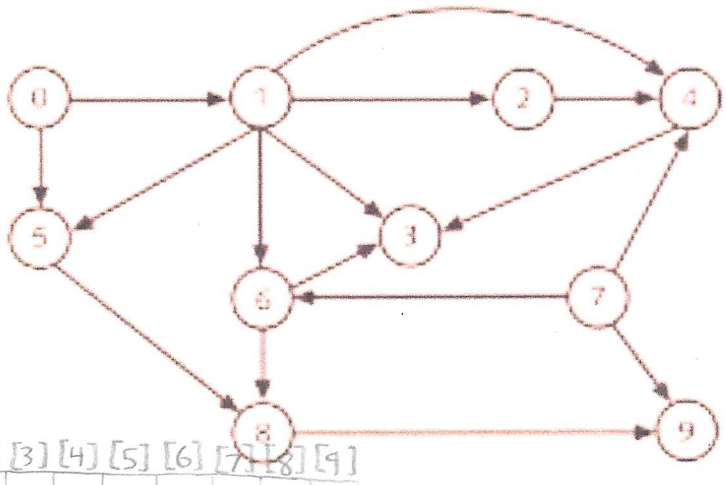


Visited={Minneapolis/St. Paul, Des Moines, Madison, Milwaukee, Chicago, St. Louis, Detroit}



Minimal Spanning Tree

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



predCount: 0 1 1 3 3 2 2 0 2 2

topologicalOrder:

queue:

predCount: 0 1 3 2 1 1 2 1

topologicalOrder:

queue: 0, 7

predCount: 0 2 1 0 0 2 1

topologicalOrder:

queue: 0, 7, 1

predCount: 1 0 0 1

topologicalOrder:

queue: 0, 7, 1, 2, 5, 6

predCount: 0 0

topologicalOrder:

queue: 0, 7, 1, 2, 5, 6, 4, 8

predCount:

topologicalOrder:

queue: 0, 7, 1, 2, 5, 6, 4, 8, 3, 9

predCount: 0 1 1 3 3 2 2 0 2 2

topologicalOrder:

queue: 0, 7

predCount: 0 1 3 2 1 1 2 1

topologicalOrder:

queue: 0, 7, 1

predCount: 0 2 1 0 0 2 1

topologicalOrder:

queue: 0, 7, 1, 2, 5, 6

predCount: 1 0 0 1

topologicalOrder:

queue: 0, 7, 1, 2, 5, 6, 4, 8

predCount: 0 0

topologicalOrder:

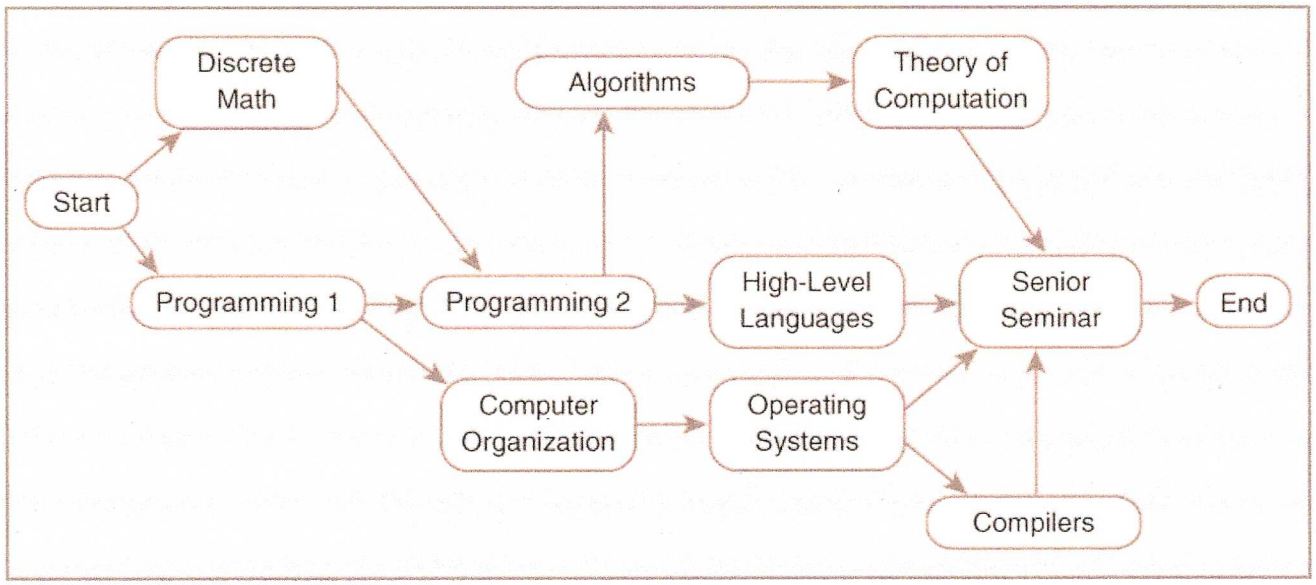
queue: 0, 7, 1, 2, 5, 6, 4, 8, 3, 9

predCount:

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

queue:

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



Nodes in breadth first topological ordering:

Start → Discrete Math → Programming 1 → Programming 2 → Computer Organization → Algorithms → High-level Languages → Operating Systems → Theory of Computation → Compilers → Senior seminar → End