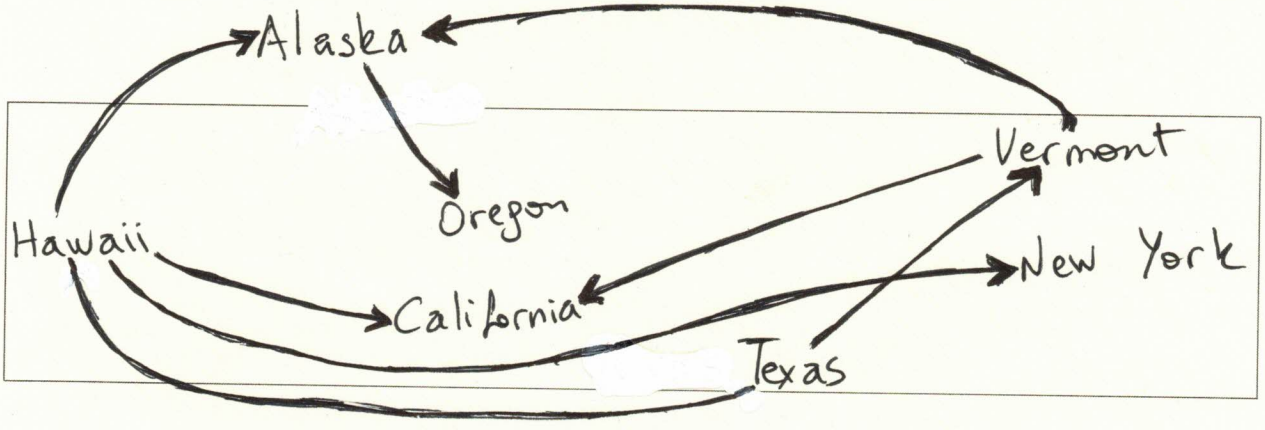
CMSC204
Kartchner

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 unconnected




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

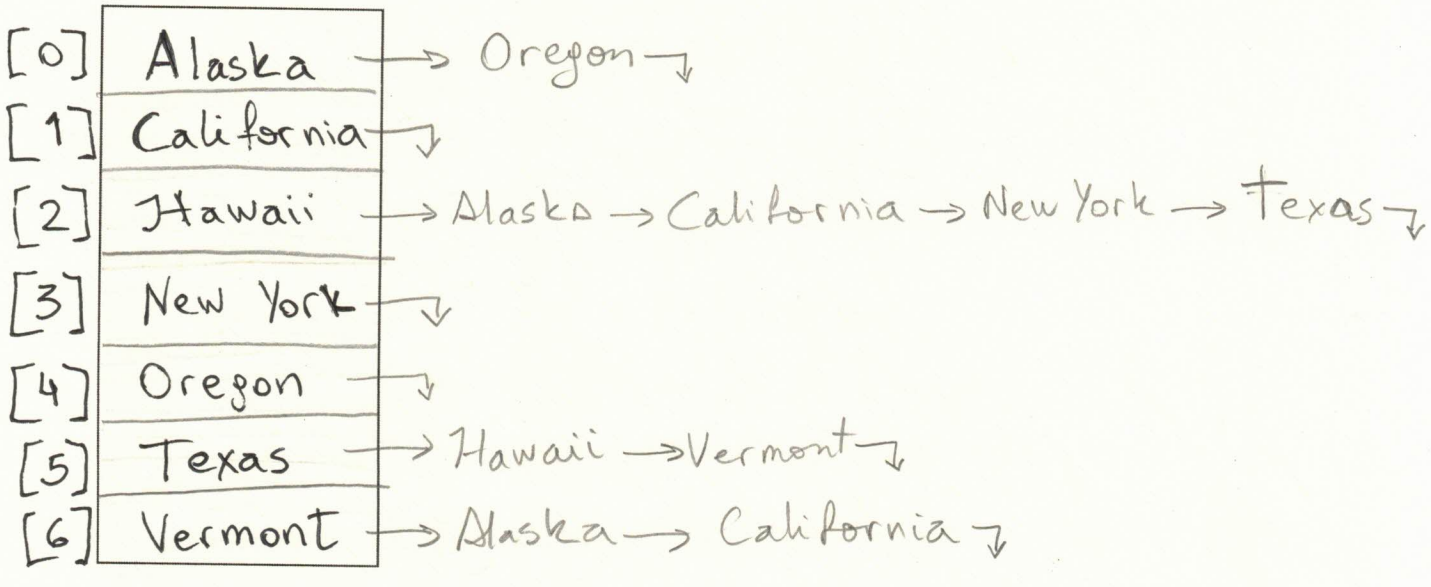
c. From which state(s) in the graph is there a path to Hawaii? Texas

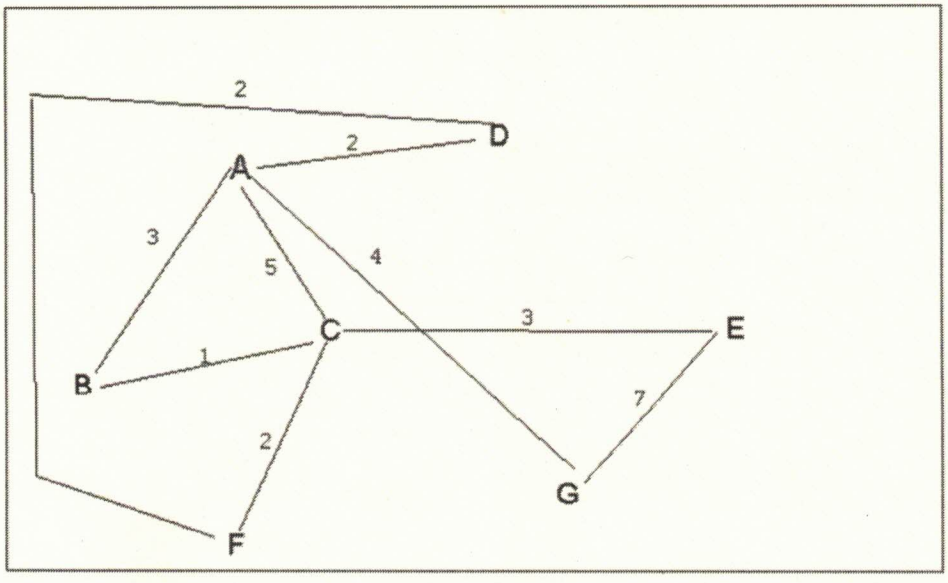
3. a. Show the adjacency matrix that would describe the edges in the graph. Store the vertices in alphabetical order

| index | States |
|---|---|
| [0] | Alaska |
| [1] | California |
| [2] | Hawaii |
| [3] | New York |
| [4] | Oregon |
| [5] | Texas |
| [6] | Vermont |

| Alaska | California | Hawaii | New York | Oregon | Texas | Vermont |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 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 → Answer

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

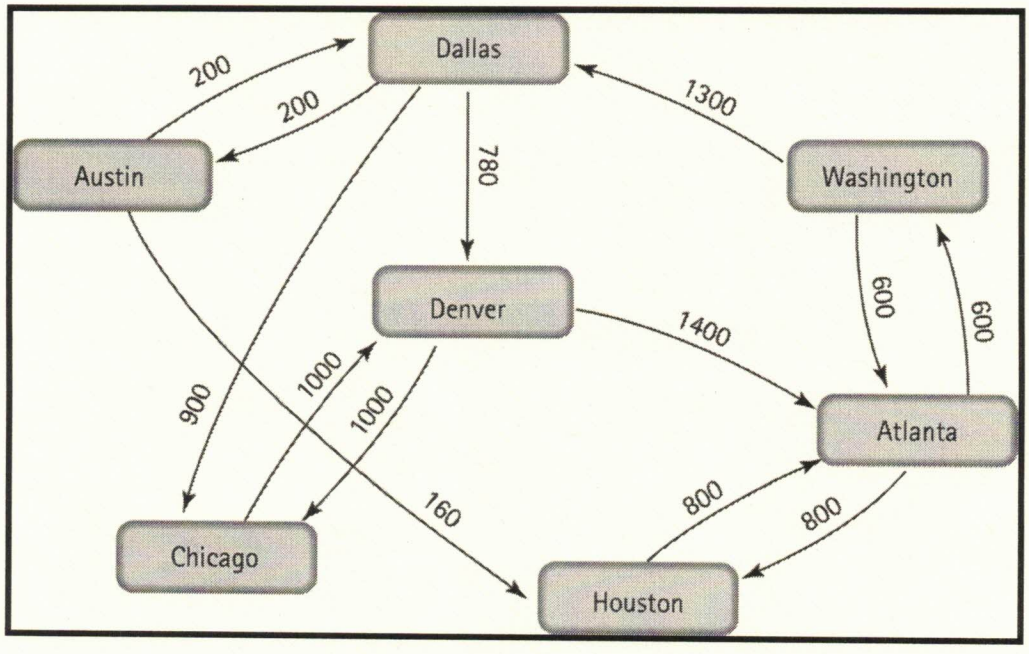
4 b. Which of the following lists the graph nodes in breadth first order beginning at F?

A) F, C, D, A, B, E, G → Answer

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 600 → Washington = 600 distance

800 → Houston = 800 distance

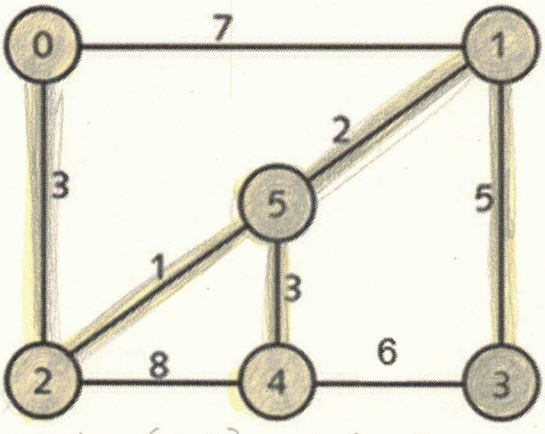
600 → Washington 1300 → Dallas = 1900 distance

600 → Washington 1300 → Dallas 780 → Denver = 2680 distance

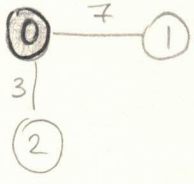
600 → Washington 1300 → Dallas 900 → Chicago = 2800 distance

600 → Washington 1300 → Dallas 200 → Austin = 2100 distance

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

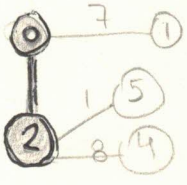



1) Edges to consider {0,2} and {0,1}.

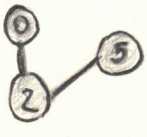



2) Edge selected {0,2}.

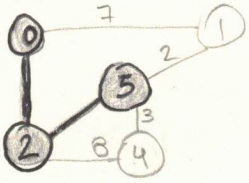

3) Edges to consider {0,1}, {2,5}, {2,4}.



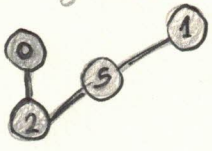

4) Edge selected {2,5}



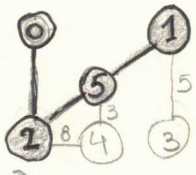

5) Edges to consider {0,1}, {2,4}, {5,4}, {5,1



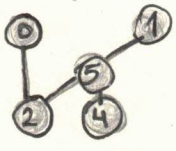

6) Edge selected {5,1}



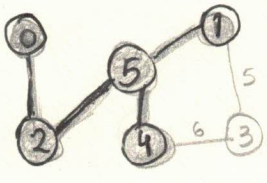

7) Edges to consider {1,3}, {5,4}, {2,4}.



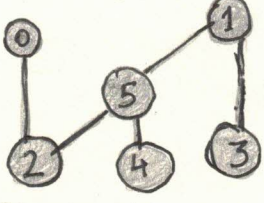

8) Edge selected {5,4}




9) Vertex (3) is the only one left to be included. Edges to consider {1,3}, {4,3}




10) Edge selected {1,3}




This is the minimal spanning tree using Prim's algorithm.

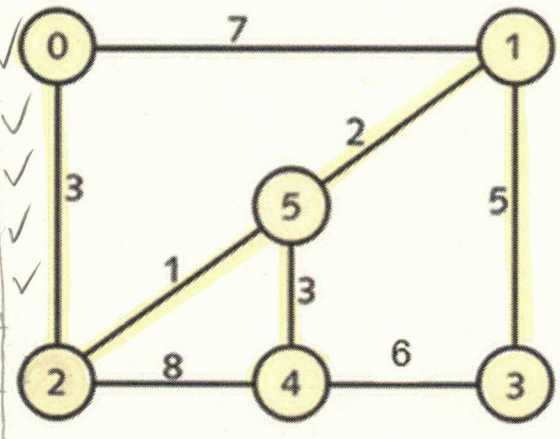
Find Edge weight = 3+1+3+2+5 = 14

MST

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

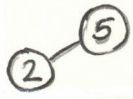
Sorting:

| weight | edge |
|---|---|
| 1 | 2–5 |
| 2 | 5–1 |
| 3 | 0–2 |
| 3 | 5–4 |
| 5 | 1–3 |
| 6 | 4–3 |
| 7 | 0–1 |
| 8 | 2–4 |

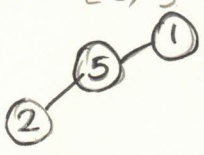



The graph contains 6 vertices, so the minimum spanning tree will have (6-1) = 5 edges after sorting.

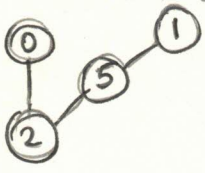
1) Pick {2, 5} edge, no cycles, so add



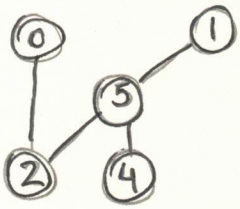

2) pick {5, 1} edge, no cycles so add



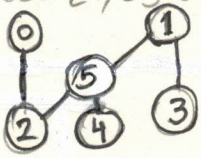

3) Pick {0, 2} edge, no cycles so add




4) Pick {5, 4} edge, no cycles so add

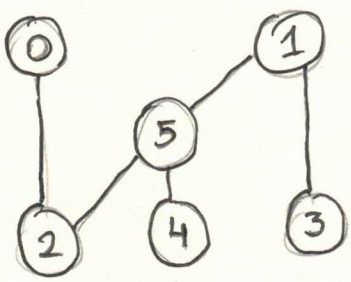



5) Pick {1, 3} edge, no cycles so add



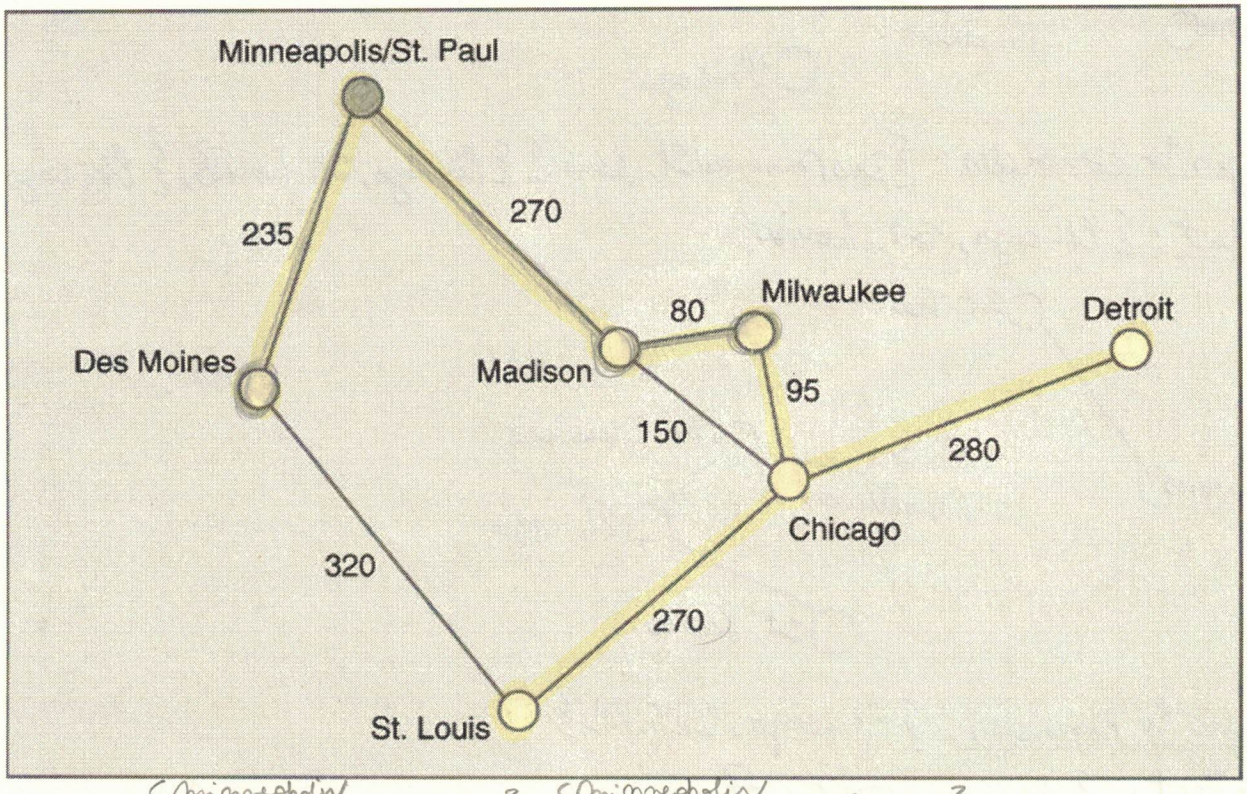

6) Since the number of edges included in the MST equals to 5 (vertices - 1), the algorithm stops here.

Final minimal spanning Tree:

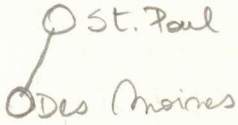


→ MST

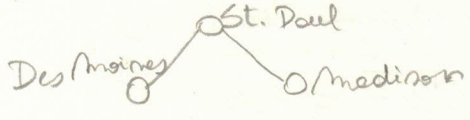
8. Find the minimal spanning tree using the algorithm you prefer. Use Minneapolis/St. Paul as the source vertex



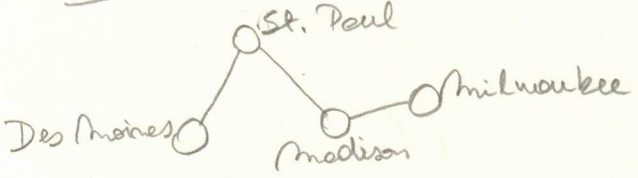

1) Edges to consider: {Minneapolis/St. Paul, Madison}, {Minneapolis/St. Paul, Des Moines}.
Select: {Minneapolis/St. Paul, Des Moines}




2) Edges to consider {Minneapolis/St. Paul, Madison}, {Des Moines, St. Louis}
Select {Minneapolis/St. Paul, Madison}

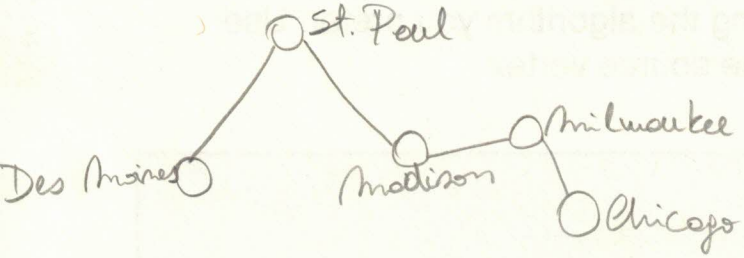



3) Edges to consider {Des Moines, St. Louis}, {Madison, Milwaukee}, {Madison, Chicag
Select: {Madison, Milwaukee}



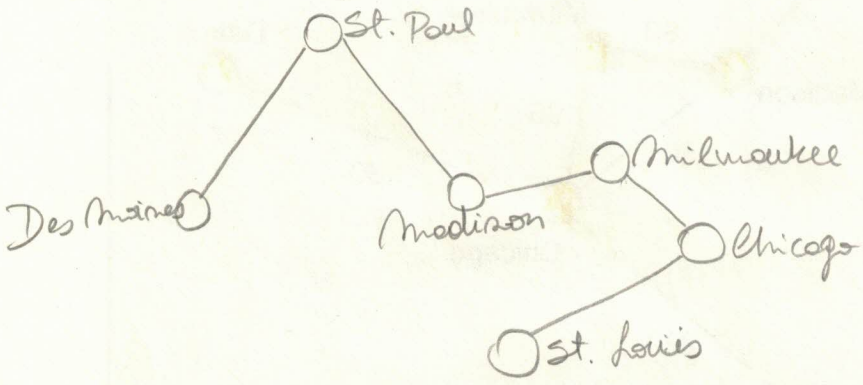
4) Edges to consider: {Madison, Chicago}, {Milwaukee, Chicago}, {Des Moines, St. Louis}

Select: {Milwaukee, Chicago




5) Edges to consider: {Des Moines, St. Louis}, {Chicago, St. Louis}, {Chicago, Detroit}

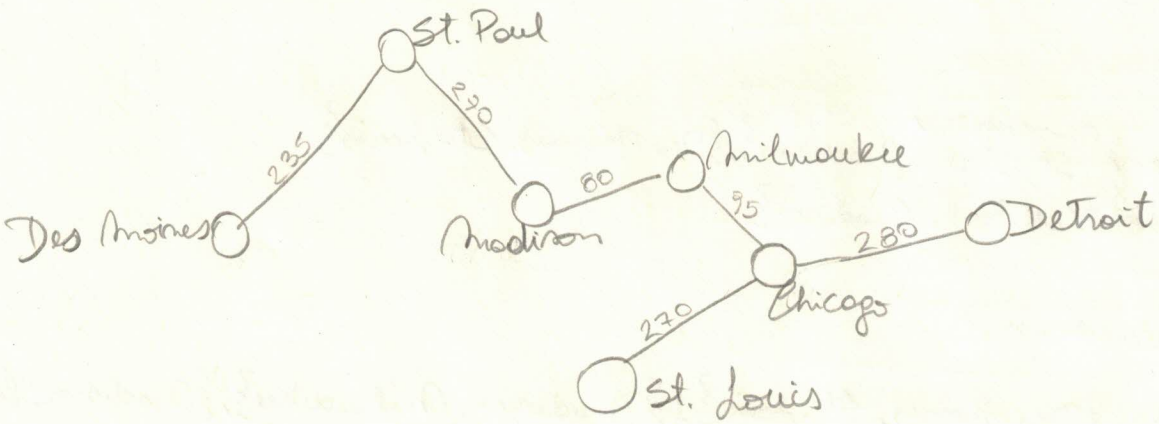
Select: {Chicago, St. Louis}




6) Edges to consider: {Chicago, Detroit}

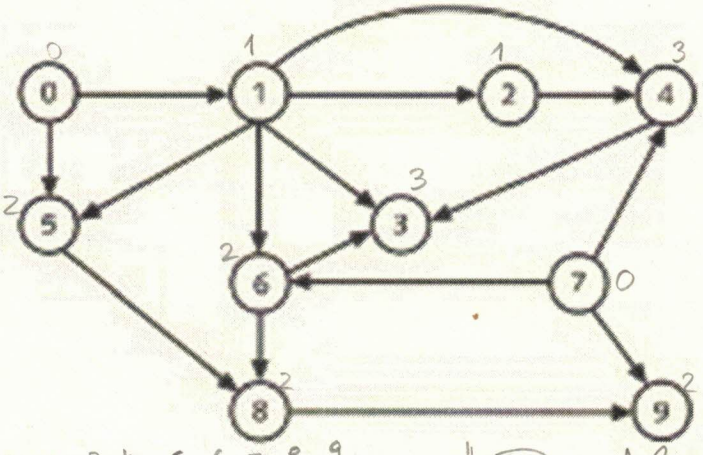
Select: {Chicago, Detroit}

Find minimal spanning tree:




Total weight: 235 + 270 + 80 + 95 + 280 + 270 = 1230

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

topologicalOrder =

2) predCount = | 0 | ~~1~~ | 1 | 3 | 3 | ~~2~~ | 2 | 0 | 2 | 2 |
0 1

queue: 7 1

topologicalOrder = 0

3) predCount = | 0 | 0 | 1 | 3 | ~~3~~ | 1 | ~~2~~ | 0 | 2 | ~~2~~ |
2 1 1

queue: 1

topologicalOrder = 0 7

4) predCount = | 0 | 0 | ~~1~~ | ~~3~~ | ~~2~~ | ~~1~~ | ~~1~~ | 0 | 2 | 1 |
0 2 1 0 0

queue: 2 5 6

topologicalOrder = 0 7 1

5) predCount = | 0 | 0 | 0 | 2 | ~~1~~ | 0 | 0 | 0 | 2 | 1 |
0

queue: 5 6 4

topologicalOrder = 0 7 1 2

6) predCount = | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | ~~2~~ | 1 |
1

queue: 6 4

topologicalOrder = 0 7 1 2 5

7) predCount = | 0 | 0 | 0 | ~~2~~ | 0 | 0 | 0 | 0 | ~~1~~ | 1 |
1 0

queue: 4 8

topologicalOrder = 0 7 1 2 5 6

8) predCount = | 0 | 0 | 0 | ~~1~~ | 0 | 0 | 0 | 0 | 0 | 1 |
0

queue: 8 3

topologicalOrder = 0 7 1 2 5 6 4

9) predCount = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~~1~~ |
0

queue: 3 9

topologicalOrder = 0 7 1 2 5 6 4 8

10) predCount = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

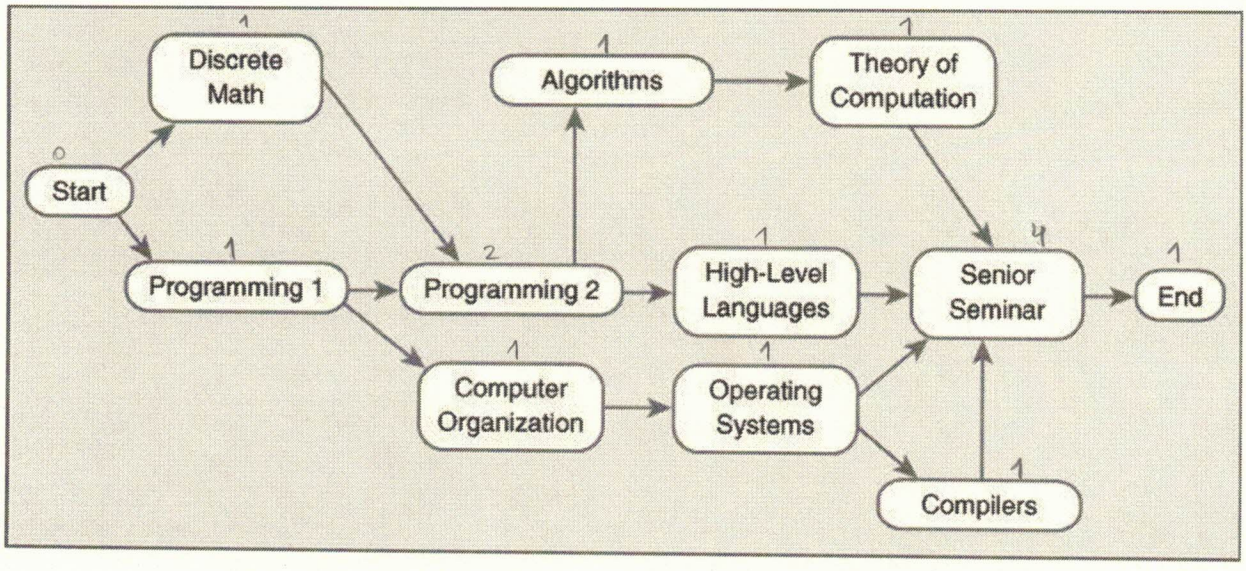
queue: 9

topological order = 0 7 1 2 5 6 4 8 3

11) predCount values indegrees are all zero, we dequeue the last vertex:

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

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




12 ✓

Nodes in breadth first topological order:

start → Discrete Math → Programming 1 → Programming 2 → Computer Organization → Algorithms → high-level languages → Operating Systems → Theory of computation → Compilers → Senior Seminar → End