In-Class Exercise

1. Use Prim's Algorithm to find a minimum spanning tree for the following graph. Show the values in nearest, distance, and F for each step.

$$
\begin{array}{c|c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 10 & \infty & 30 & 45 & \infty \\
10 & 0 & 50 & \infty & 40 & 25 \\
\infty & 50 & 0 & \infty & 35 & 15 \\
30 & \infty & \infty & 0 & \infty & 20 \\
45 & 40 & 35 & \infty & 0 & 55 \\
\infty & 25 & 15 & 20 & 55 & 0
\end{bmatrix}
\end{array}
$$

Answer

Step 1:

- Nearest = {-1, 1, 1, 1, 1, 1}
- Distance = {-1, 10, ∞, 30, 45, ∞}
- min = 10
- vnear = 2
- add(nearest[2], 2) to F
- F = {(1, 2)}
- Set distance[vnear] to -1. See if any remaining vertices are closer to vnear than to their current nearest vertex

Step 2:

- Nearest = {-1, -1, **2**, 1, **2**, **2**}
- Distance = {-1, -1, **50**, 30, **40**, **25**}
- min = 25
- vnear = 6
- add(nearest[6], 6) to F
- F = {(1, 2), (2, 6)}
- Set distance[vnear] to -1. See if any remaining vertices are closer to vnear than to their current nearest vertex

Answer

Step 3:

- Nearest = {-1, -1, **6**, **6**, 2, -1}
- Distance = {-1, -1, **15**, **20**, 40, -1}
- min = 15
- vnear = 3
- add(nearest[3], 3) to F
- F = {(1, 2), (2, 6), (6, 3)}
- Set distance[vnear] to -1. See if any remaining vertices are closer to vnear than to their current nearest vertex

Step 4:

- Nearest = {-1, -1, -1, 6, **3**, -1}
- Distance = {-1, -1, -1, 20, **35**, -1}
- min = 20
- vnear = 4
- add(nearest[4], 4) to F
- F = {(1, 2), (2, 6), (6, 3), (6, 4)}
- Set distance[vnear] to -1. See if any remaining vertices are closer to vnear than to their current nearest vertex

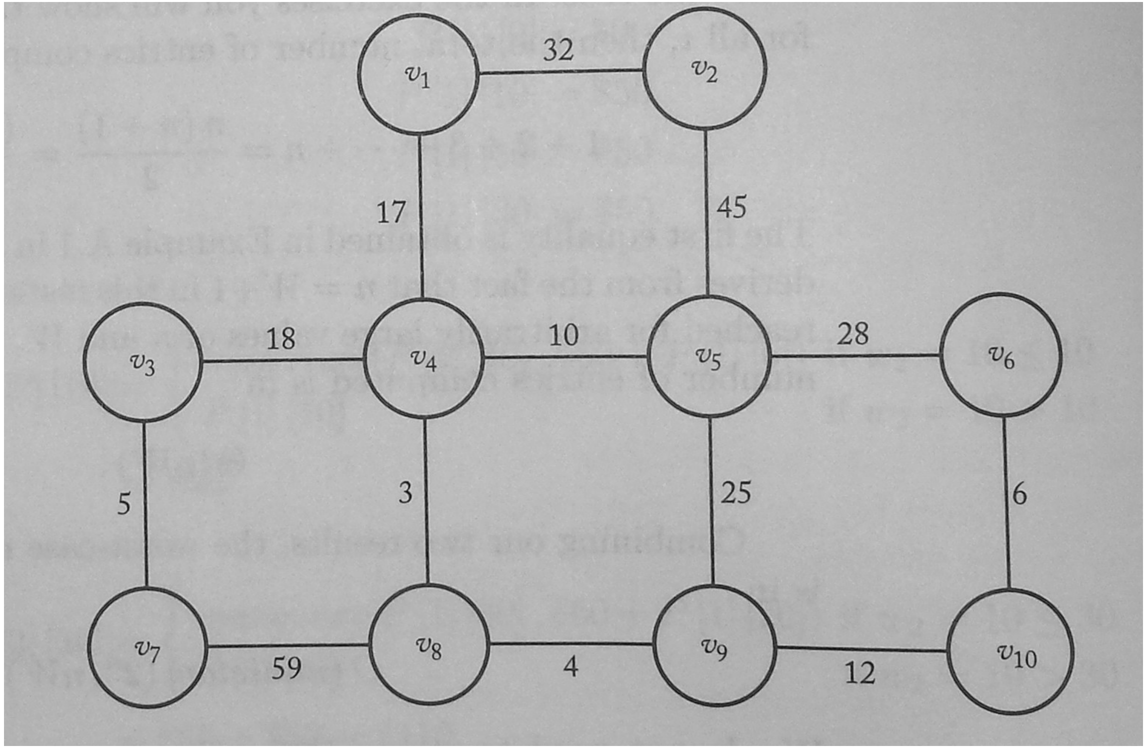
Answer

Step 5:

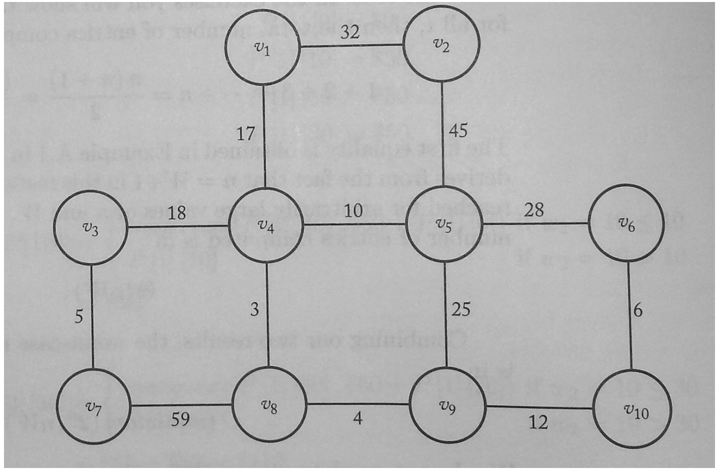
- Nearest = {-1, -1, -1, -1, 3, -1}
- Distance = {-1, -1, -1, -1, 35, -1}
- min = 35
- vnear = 5
- add(nearest[5], 5) to F
- F = {(1, 2), (2, 6), (6, 3), (6, 4), (3, 5)}
- We are done!

In-Class Exercise

1. Use Kruskal's Algorithm to find a minimum spanning tree for the following graph



Answer



V = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

F={}

Put vertices in disjoint sets:

Step 1:

{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}

Smallest edge remaining: 4 - 8

F: { (4, 8) }

Step 2:

{1}, {2}, {3}, {5}, {6}, {7}, {4, 8}, {9}, {10}

Smallest edge remaining: 8 - 9

F: { (4, 8), (8, 9) }

Answer

Step 3:

{1}, {2}, {3}, {5}, {6}, {7}, {4, 8, 9}, {10}

Smallest edge remaining: 3 - 7

F: { (3, 7), (4, 8), (8, 9) }

Step 4:

{1}, {2}, {5}, {6}, {3, 7}, {4, 8, 9}, {10}

Smallest edge remaining: 6 - 10

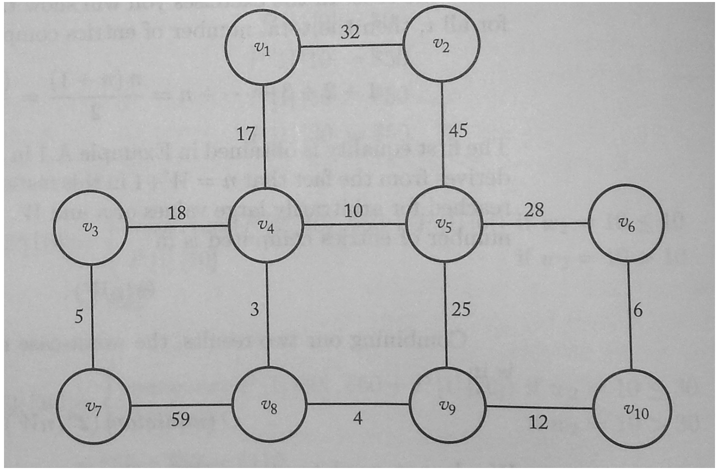
F: { (3, 7), (4, 8), (8, 9), (6, 10)}

Step 5:

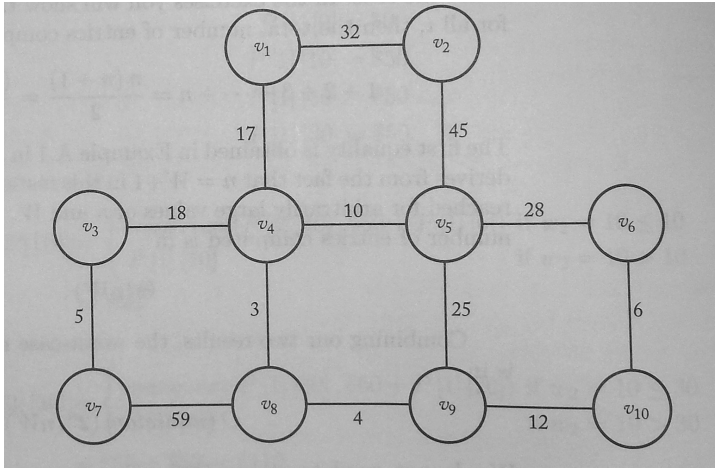
{1}, {2}, {5}, {3, 7}, {4, 8, 9}, {6, 10}

Smallest edge remaining: 4 - 5

F: { (3, 7), (4, 8), (4, 5), (8, 9), (6, 10)}



Answer



Step 6:

{1}, {2}, {3, 7}, {4, 5, 8, 9}, {6, 10}

Smallest edge remaining: 9 - 10

F: { (3, 7), (4, 8), (4, 5), (8, 9), (6, 10), (9, 10)}

Step 7:

{1}, {2}, {3, 7}, {4, 5, 6, 8, 9, 10}

Smallest edge remaining: 1 - 4

F: { (1, 4), (3, 7), (4, 8), (4, 5), (8, 9), (6, 10), (9, 10)}

Step 8:

{2}, {3, 7}, {1, 4, 5, 6, 8, 9, 10}

Smallest edge remaining: 3 - 4

F: { (1, 4), (3, 7), (4, 8), (4, 5), (8, 9), (6, 10), (9, 10)}

Answer

Step 8:

{2}, {1, 3, 4, 5, 6, 7, 8, 9, 10}

Smallest edge remaining: 5-9

Edge rejected, not in disjoint sets

Smallest edge remaining: 5-6

Edge rejected, not in disjoint sets

Smallest edge remaining: 1 - 2

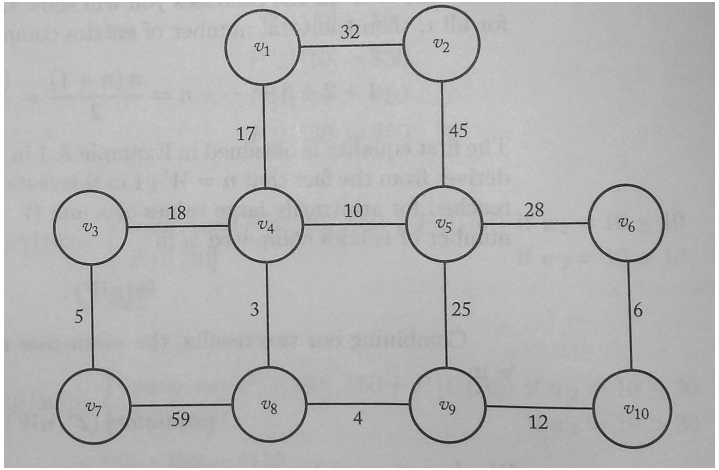
F: { (1, 2), (1, 4), (3, 7), (4, 8),
(4, 5), (8, 9), (6, 10), (9, 10)}

Step 8:

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

No more disjoint sets, so we are done!

F: { (1, 2), (1, 4), (3, 7), (4, 8), (4, 5), (8, 9), (6, 10), (9, 10)}



In-Class Exercise

1. Consider the following jobs, deadlines, and profits. Use the Scheduling with Deadlines algorithm to maximize the total profit

Job	Deadline	Profit
1	2	40
2	4	15
3	3	60
4	2	20
5	3	10
6	1	45
7	1	55

In-Class Exercise

Job	Deadline	Profit
3	3	60
7	1	55
6	1	45
1	2	40
4	2	20
2	4	15
5	3	10

First: Sort the jobs in nonincreasing order of profit

1. *finalSequence* is set to [3]
2. *temp* is set to [7, 3] and is determined to be feasible. *finalSequence* is set to [7, 3] since *temp* is feasible.
3. *temp* is set to [7, 6, 3] and is rejected. It is not feasible.
4. *temp* is set to [7, 1, 3] and is determined to be feasible. *finalSequence* is set to [7, 1, 3] because *temp* is feasible.
5. *temp* is set to [7, 1, 4, 3] and is rejected.
6. *temp* is set to [7, 1, 3, 2] and is determined to be feasible. *finalSequence* is set to [7, 1, 3, 2] because *temp* is feasible.
7. *temp* is set to [7, 1, 3, 5, 2] and is rejected.

The final value of *finalSequence* is [7, 1, 3, 2] with a profit of 170.

In-Class Exercise

1. Use Dijkstra's Algorithm to find the shortest path from vertex 5 to all the other vertices in the following graph. Show actions step by step.

	1	2	3	4	5	6
1	0	∞	1	5	9	2
2	∞	0	3	2	5	7
3	1	3	0	∞	15	9
4	5	2	∞	0	2	3
5	9	8	15	2	0	8
6	2	7	9	3	8	0

Answer

Step 1:

Y = {5}

F = {}

Touch = [5, 5, 5, 5, -1, 5]

Length = [9, 8, 15, 2, -1, 8]

vnear = 4, e = <5, 4>

Step 2:

Y = {4, 5}

F = {<5, 4>}

Touch = [4, 4, 5, -1, -1, 4]

Length = [7, 4, 15, -1, -1, 5]

vnear = 2, e = <4, 2>

Answer

Step 3:

Y = {2, 4, 5}

F = {<5, 4>, <4, 2>}

Touch = [4, -1, 2, -1, -1, 4]

Length = [7, -1, 7, -1, -1, 5]

vnear = 6, e = <4, 6>

Step 4:

Y = {2, 4, 5, 6}

F = {<5, 4>, <4, 2>, <4, 6>}

Touch = [4, -1, 2, -1, -1, -1]

Length = [7, -1, 7, -1, -1, -1]

vnear = 1, e = <4, 1>

Step 5:

Y = {1, 2, 4, 5, 6}

F = {<5, 4>, <4, 2>, <4, 6>, <4, 1>}

Touch = [-1, -1, 2, -1, -1, -1]

Length = [-1, -1, 7, -1, -1, -1]

vnear = 3, e = <2, 3>

F = {<5, 4>, <4, 2>, <4, 6>, <4, 1>, <2, 3>}