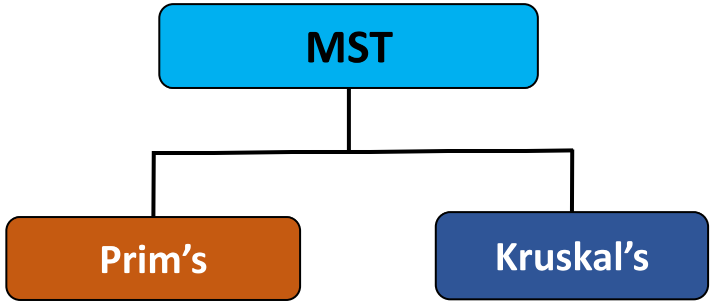
CS2x1:Data Structures and Algorithms

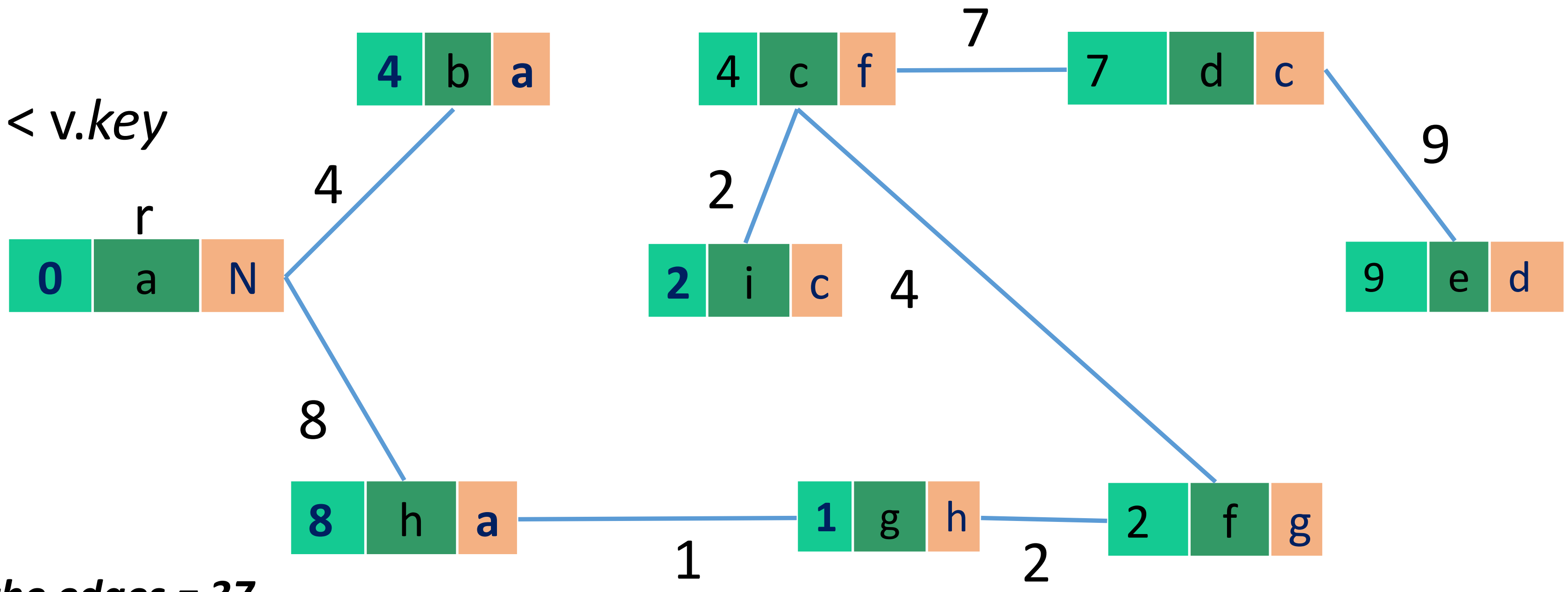
Koteswararao Kondepu

k.kondepu@iitdh.ac.in

Recap: Minimum Spanning Tree (MST)

```
MST-PRIM (G, w, r) {
    for each u ϵ G.V
        u.key = ∞
        u. π = NIL
    r.key = 0
    Q = G.V
    while Q ≠ ∅ ;
      u = EXTRACT-MIN(Q)
      for each v ϵ G.Adj[u]
        if v ϵ Q and w(u,v) < v.key
          v. π = u
          v.key = w(u,v)
}
```

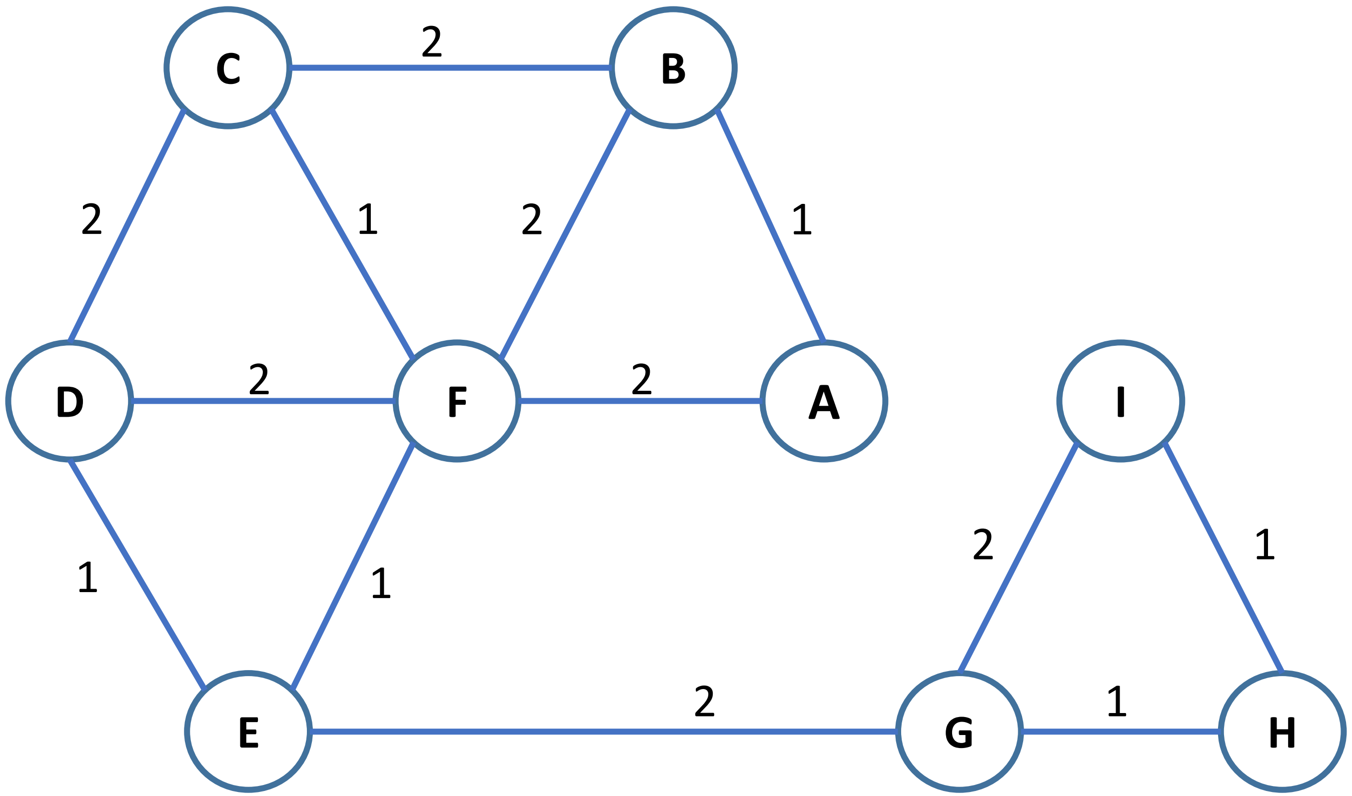




<u>MST → Length of the edges = 37</u>

Exercise: Prim's algorithm

1. What will the path obtained after applying Prim's algorithm with a starting *vertex C ?*



2. The number of distinct minimum spanning trees for the weighted graph below is
    a. 4
    b. 5
    c. 6
    d. 7

MST → Prim's algorithm→ Time complexity analysis

```
MST-PRIM (G, w, r) {
    for each u ϵ G.V
        u.key = ∞                          O(V) if Q is implemented as a min-heap
        u. π = NIL
    r.key = 0
    Q = G.V                            ←  O(log V)
    while Q ≠ ∅ ;                      ←  |V| times
      u = EXTRACT-MIN(Q)               ←  O(log V)        O(V logV)
      for each v ϵ G. Adj[u]           ←  |E| times
        if v ϵ Q and w(u, v) < v. key
            v. π = u                                      O(E logV)
            v.key = w(u,v)             ←  O(log V)
}
```

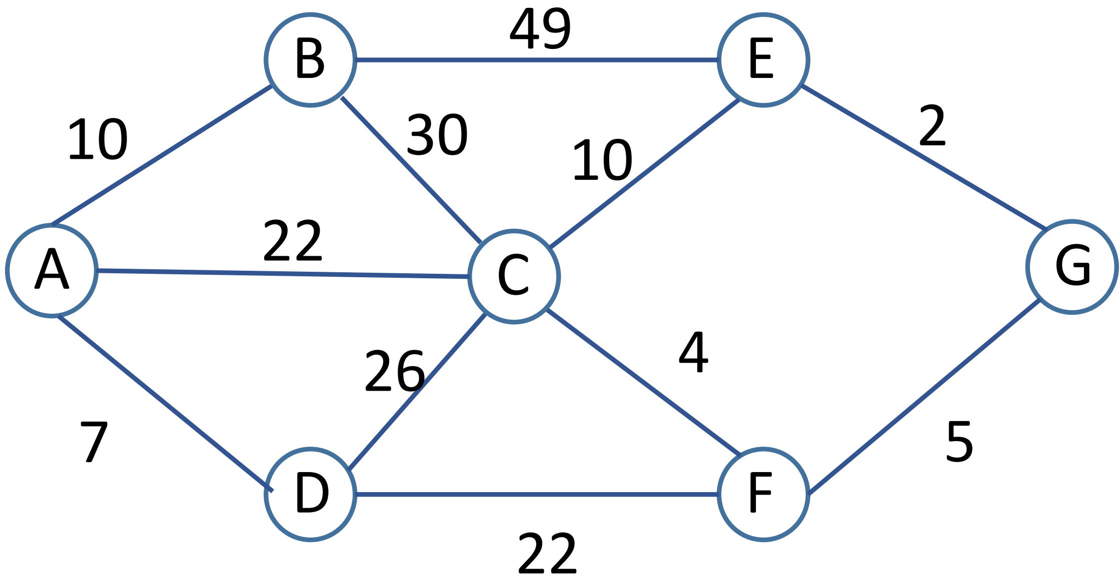
Total time complexity: **<u>O(V logV + E logV) = O((V+E) logV) = O(E logV)</u>**

Exercise: MST Prim’s algorithm

Consider the following graph:

Using **Prim's algorithm to construct a minimum spanning tree starting with node A**, which one of the following sequences of edges represents a possible order in which the edges would be added to construct the minimum spanning tree?

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



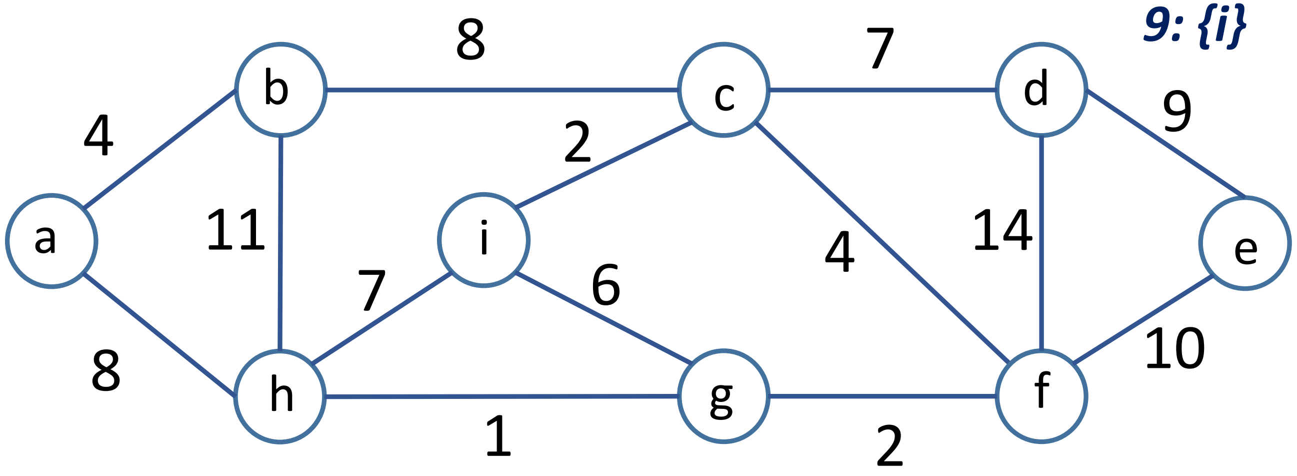
MST → Kruskal's algorithm (1)

```
MST- KRUSKAL(G, w) {
1   A = ∅
2   for each vertex v ϵ G. V
3           MAKE-SET (v)
4   sort the edges of G. E into non-decreasing order by weight w
5   for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
6           if FIND-SET (u) ≠ FIND-SET (v)
7                   A = A ∪ {(u, v)}
8                   UNION (u, v)
9   return A
}
```

Step 1: Initialize the set A to the empty set

Step 2 -3: Make sets for each vertex

1: {a}
2: {b}
3: {c}
4: {d}
5: {e}
6: {f}
7: {g}
8: {h}
9: {i}



MST → Kruskal's algorithm (2)

MST- KRUSKAL(G, w) {

1 A = ∅

2 *for* each vertex *v* ϵ G. V

3   MAKE-SET (*v*)

4 sort the edges of G. E into *__non-decreasing__* order by weight *w*

5 *for* each edge (*u, v*) ϵ G. E, taken in *__non-decreasing__* order by weight

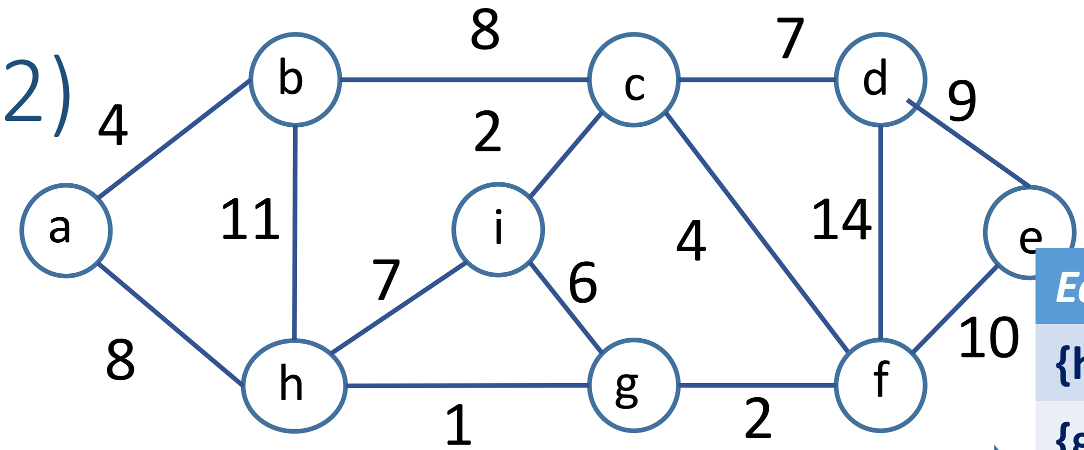
6   if FIND-SET (*u*) ≠ FIND-SET (*v*)

7   A = A ∪ {(u, v)}

8   UNION (*u, v*)

9 *return* A

}



Step 4: Sort the edges → nondecreasing order

Step 2-3:

1: {a}

2: {b}

3: {c}

4: {d}

5: {e}

6: {f}

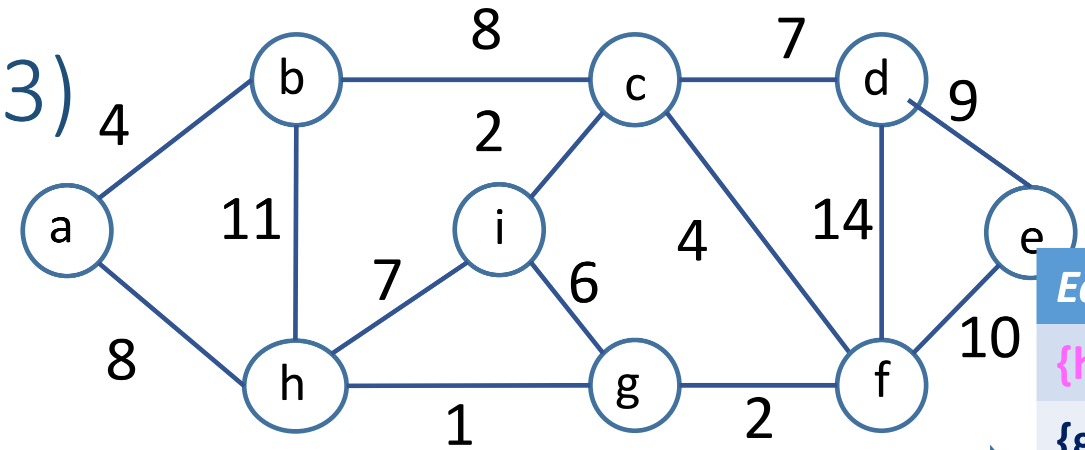
7: {g}

8: {h}

9: {i}

Edges	***Weight***
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST → Kruskal's algorithm (3)

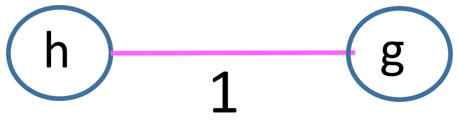


Step 4: Sort the edge → nondecreasing order

```
MST- KRUSKAL(G, w) {
1   A = ∅
2   for each vertex v ϵ G. V
3         MAKE-SET (v)
4   sort the edges of G. E into non-decreasing order by weight w
5   for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
6         if FIND-SET (u) ≠ FIND-SET (v)
7                 A = A ∪ {(u, v)}
8                 UNION (u, v)
9   return A
}
```

Step 2-3:
1: {a}
2: {b}
3: {c}
4: {d}
5: {e}
6: {f}
7: {g}
8: {h}
9: {i}

Step 5:



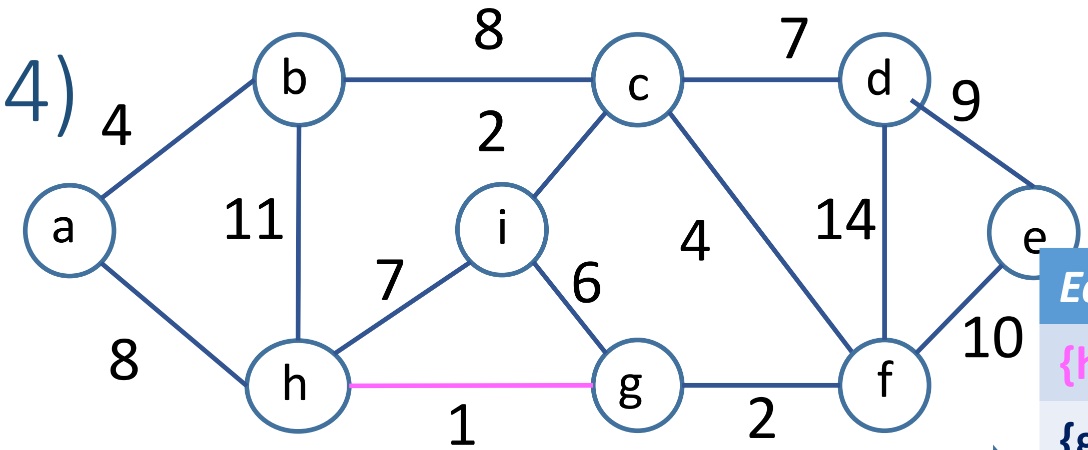
Step 6: *FIND-SET (h) ≠ FIND-SET (g)*
8 ≠ 7

Step 7: *A = {(h, g)}*

Step 8: *8: {h, g}*

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST → Kruskal's algorithm (4)

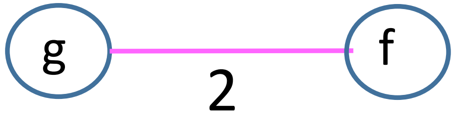


Step 4: Sort the edge → nondecreasing order

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into <u>*non-decreasing*</u> order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in <u>*non-decreasing*</u> order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 2-3:
1: {a}
2: {b}
3: {c}
4: {d}
5: {e}
6: {f}
8: {h, g}
9: {i}

Step 5:



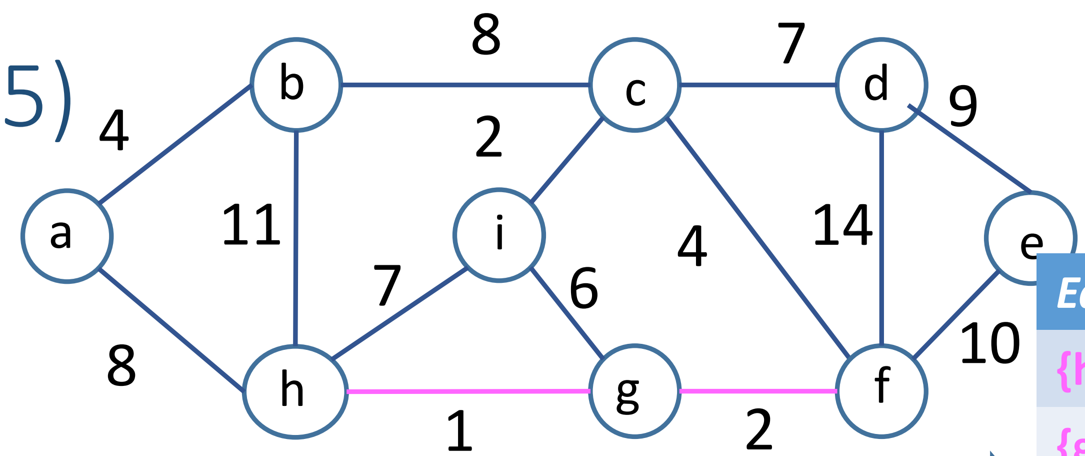
Step 6: *FIND-SET (g) ≠ FIND-SET (f)*
8 ≠ 6

Step 7: *A = {(h, g) ∪ (g, f)}*
= {(h, g), (g, f)}

Step 8: *8: {h, g, f}*

Edges	***Weight***
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST → Kruskal's algorithm (5)



Step 4: Sort the edge → nondecreasing order

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST- KRUSKAL(G, w) {

1 A = ∅

2 *for* each vertex *v* ϵ G. V

3 MAKE-SET (*v*)

4 sort the edges of G. E into *non-decreasing* order by weight *w*

5 *for* each edge (*u, v*) ϵ G. E, taken in *non-decreasing* order by weight

6 if FIND-SET (*u*) ≠ FIND-SET (*v*)

7 A = A ∪ {(u, v)}

8 UNION (*u, v*)

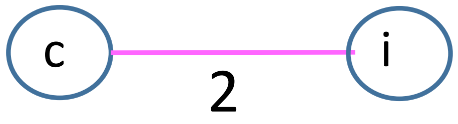
9 *return* A

}

Step 2-3:

1: {a}
2: {b}
3: {c}
4: {d}
5: {e}
8: {h, g, f}
9: {i}

Step 5:

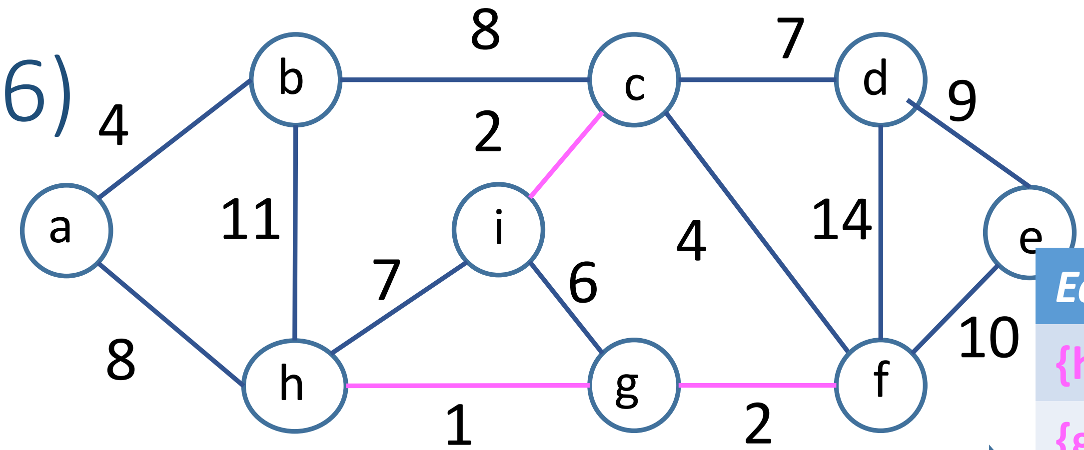


Step 6: ***FIND-SET (c) ≠ FIND-SET (i)***
3 ≠ 9

Step 7: ***A = {(h, g), (g, f)} ∪ {(c, i)}***
= {(h, g), (g, f), (c, i)}

Step 8: ***8: {c, i}***

MST → Kruskal's algorithm (6)



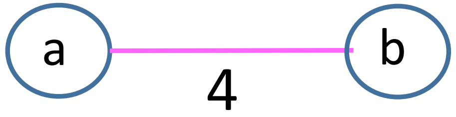
Step 4: Sort the edge → nondecreasing order

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into ***non-decreasing*** order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in ***non-decreasing*** order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 2-3:
1: {a}
2: {b}
3: {c, i}
4: {d}
5: {e}
8: {h, g, f}

Step 5:

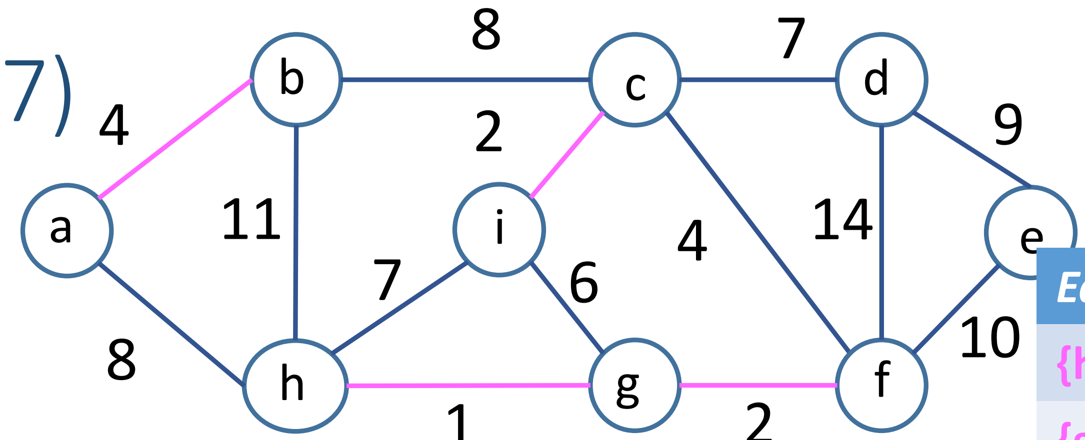


Step 6: ***FIND-SET (a) ≠ FIND-SET (b)***
1 ≠ 2

Step 7: ***A = {(h, g), (g, f), (c, i)} ∪ {(a, b)}***
= {(h, g), (g, f), (c, i), (a, b)}

Step 8: ***8: {a, b}***

MST → Kruskal's algorithm (7)

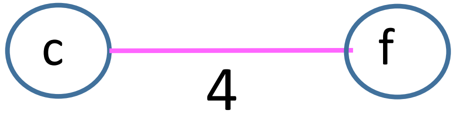


Step 4: Sort the edge → nondecreasing order

```
MST- KRUSKAL(G, w) {
1   A = ∅
2   for each vertex v ϵ G. V
3           MAKE-SET (v)
4   sort the edges of G. E into non-decreasing order by weight w
5   for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
6           if FIND-SET (u) ≠ FIND-SET (v)
7                   A = A ∪ {(u, v)}
8                   UNION (u, v)
9   return A
}
```

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



Step 6: *FIND-SET (c) ≠ FIND-SET (f)*

3 ≠ 8

Step 7: *A = {(h, g), (g, f), (c, i), (a, b)} ∪ {(c, f)}*

= {(h, g), (g, f), (c, i), (a, b), (c, f)}

Step 8: *8: {h, g, f, c, i}*

Step 2-3:

1:{a, b}

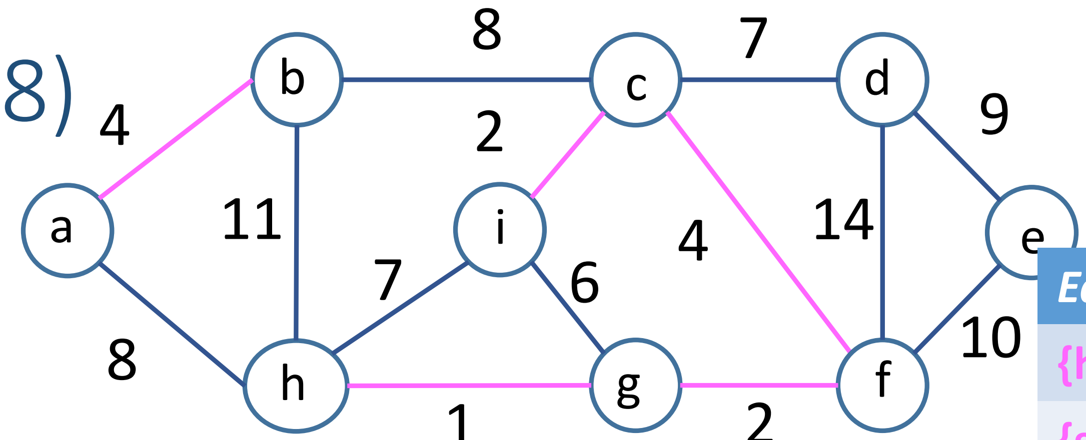
3: {c, i}

4: {d}

5: {e}

8: {h, g, f}

MST → Kruskal's algorithm (8)

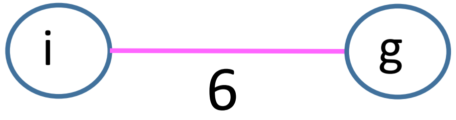


Step 4: Sort the edge → nondecreasing order

```
MST- KRUSKAL(G, w) {
1   A = ∅
2   for each vertex v ϵ G. V
3         MAKE-SET (v)
4   sort the edges of G. E into non-decreasing order by weight w
5   for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
6         if FIND-SET (u) ≠ FIND-SET (v)
7                 A = A ∪ {(u, v)}
8                 UNION (u, v)
9   return A
}
```

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



Step 6: ***FIND-SET (i) ≠ FIND-SET (g)***

8 ≠ 8

Step 7:

Step 8:

Step 2-3:

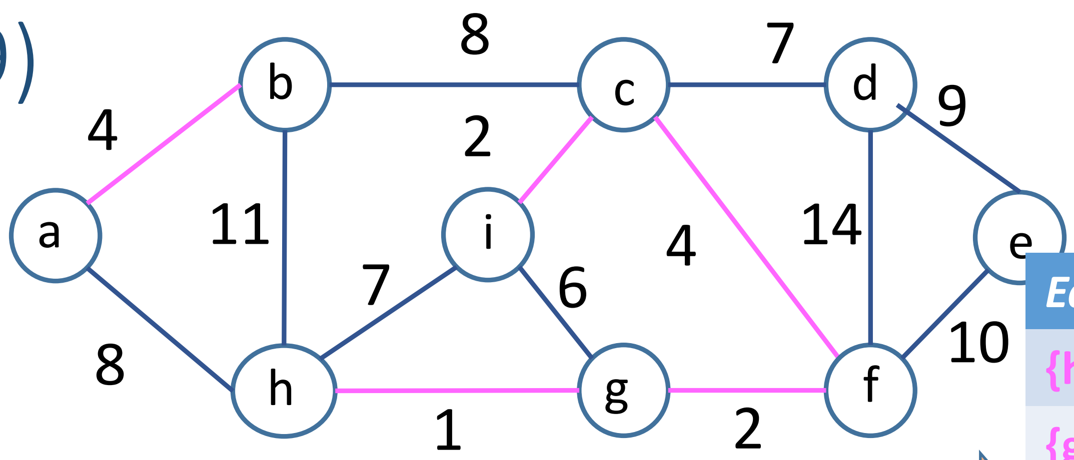
1:{a, b}

4: {d}

5: {e}

8: {h, g, f, c, i}

MST → Kruskal's algorithm (9)

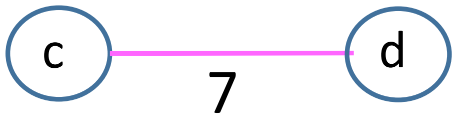


MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into <u>*non-decreasing*</u> order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in <u>*non-decreasing*</u> order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 4: Sort the edge → nondecreasing order

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



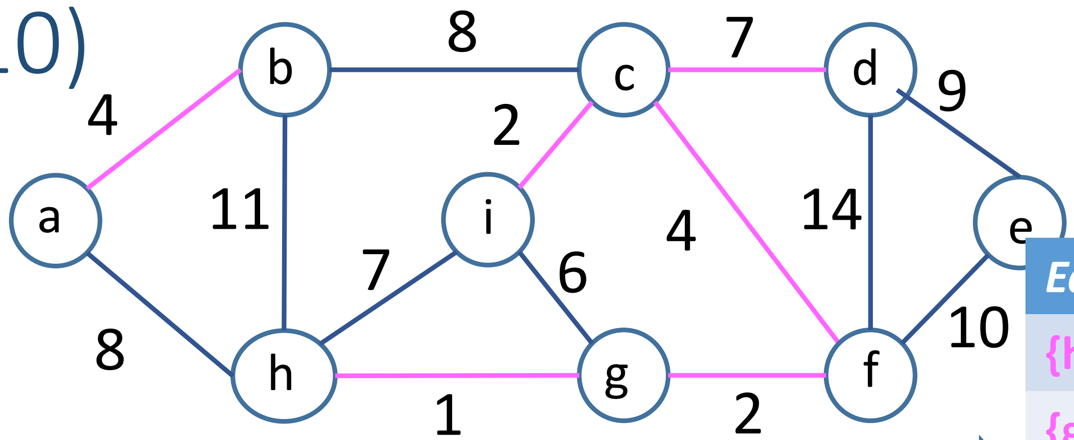
Step 6: ***FIND-SET (c) ≠ FIND-SET (d)***
8 ≠ 4

Step 7: ***A = {(h, g), (g, f), (c, i), (a, b), (c, f)} ∪ {(c, d)}***
= {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d)}

Step 8: ***8: {h, g, f, c, i, d}***

Step 2-3:
1:{a, b}
4: {d}
5: {e}
8: {h, g, f, c, i}

MST → Kruskal's algorithm (10)



Step 4: Sort the edge → nondecreasing order

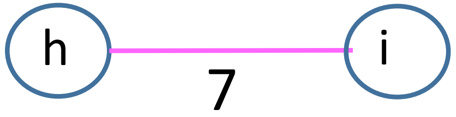
Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into <u>*non-decreasing*</u> order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in <u>*non-decreasing*</u> order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 2-3:

1:{a, b}
5: {e}
8: {h, g, f, c, i, d}

Step 5:

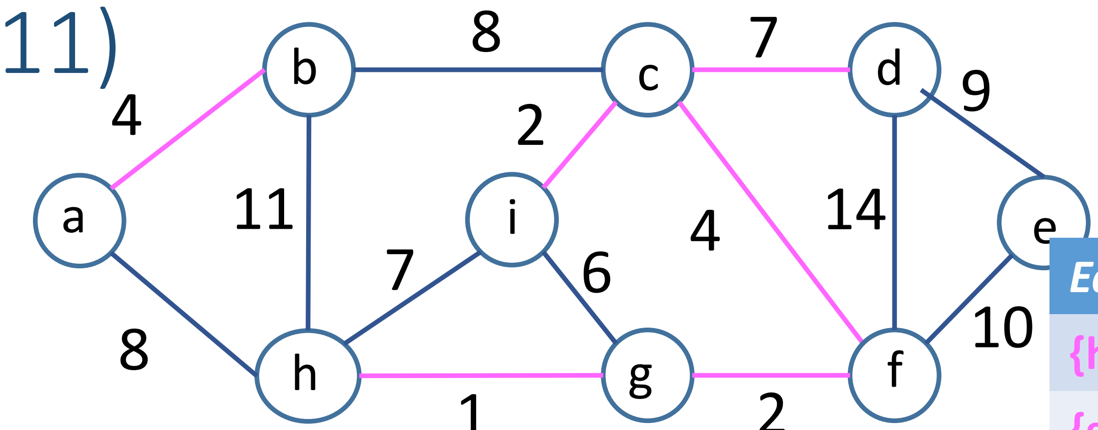


Step 6: ***FIND-SET (h) ≠ FIND-SET (i)***
8 ≠ 8

Step 7:

Step 8:

MST → Kruskal's algorithm (11)

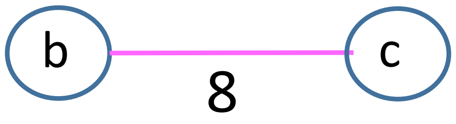


MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into *non-decreasing* order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in *non-decreasing* order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 4: Sort the edge → nondecreasing order

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



Step 6: FIND-SET (b) ≠ FIND-SET (c)

1 ≠ 8

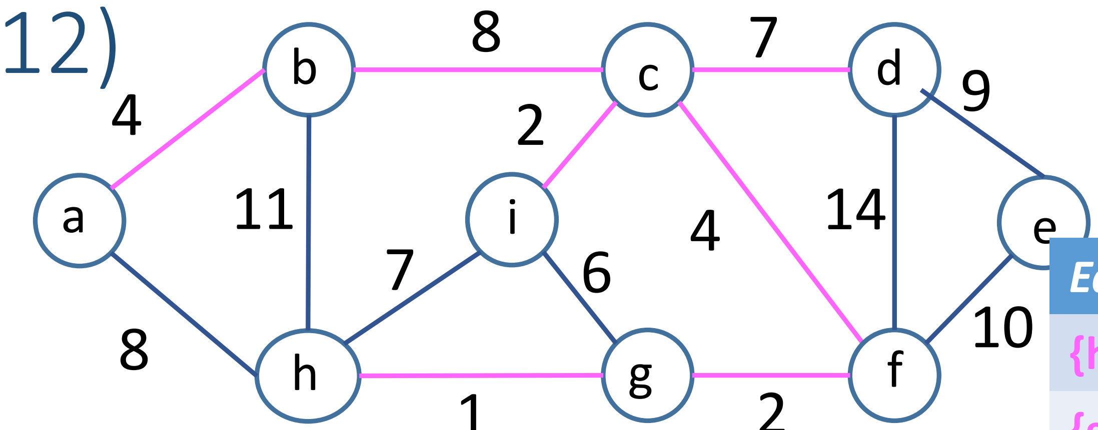
Step 7: A = {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d)} ∪ {(b, c)}
= {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d), (b, c)}

Step 2-3:

1:{a, b}
5: {e}
8: {h, g, f, c, i, d}

Step 8: 8: {h, g, f, c, i, d, a, b}

MST → Kruskal's algorithm (12)

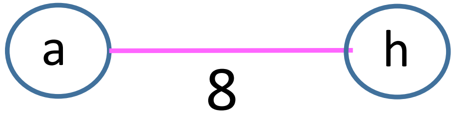


Step 4: Sort the edge → nondecreasing order

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	**4**
{i, g}	**6**
{c, d}	7
{h, i}	**7**
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into <u>*non-decreasing*</u> order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in <u>*non-decreasing*</u> order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 5:



Step 6: ***FIND-SET (a) ≠ FIND-SET (h)***
8 ≠ 8

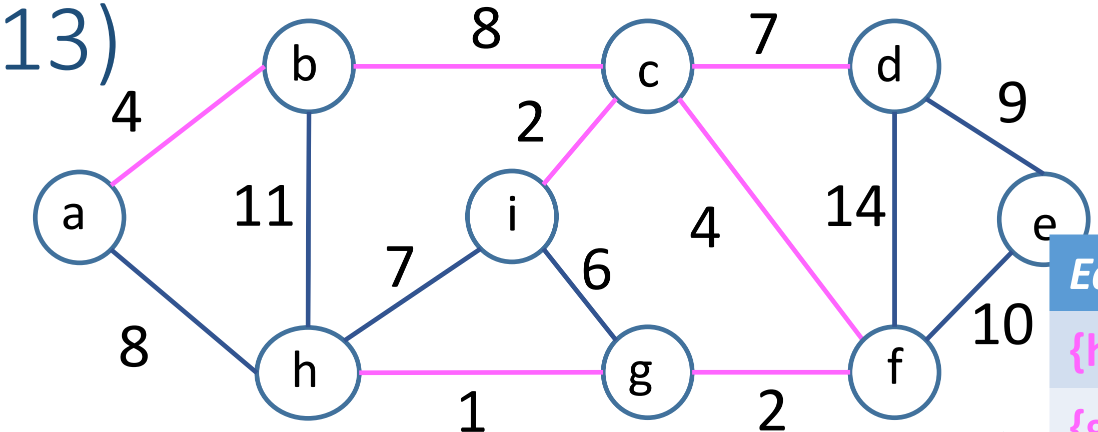
Step 7:

Step 8:

Step 2-3:

5: {e}
8: {h, g, f, c, i, d, a, b}

MST → Kruskal's algorithm (13)

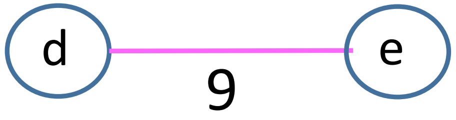


MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into *non-decreasing* order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in *non-decreasing* order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 4: Sort the edge → nondecreasing order

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



Step 6: FIND-SET (d) ≠ FIND-SET (e)

8 ≠ 5

Step 2-3:

Step 7: A = {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d} ∪ {(d, e)}

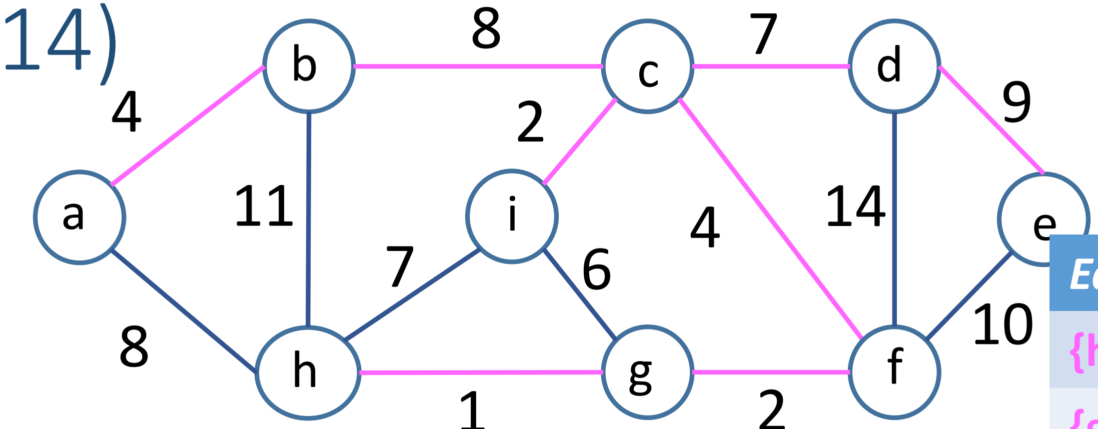
= {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d), (b, c), (d, e)}

5: {e}

8: {h, g, f, c, i, a, b, d}

Step 8: 8: {h, g, f, c, i, a, b, d, e}

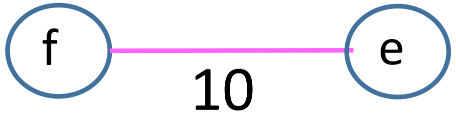
MST → Kruskal's algorithm (14)



Step 4: Sort the edge → nondecreasing order

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into *non-decreasing* order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in *non-decreasing* order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Step 5:



Step 6: FIND-SET (f) ≠ FIND-SET (e)

8 ≠ 8

Step 2-3:

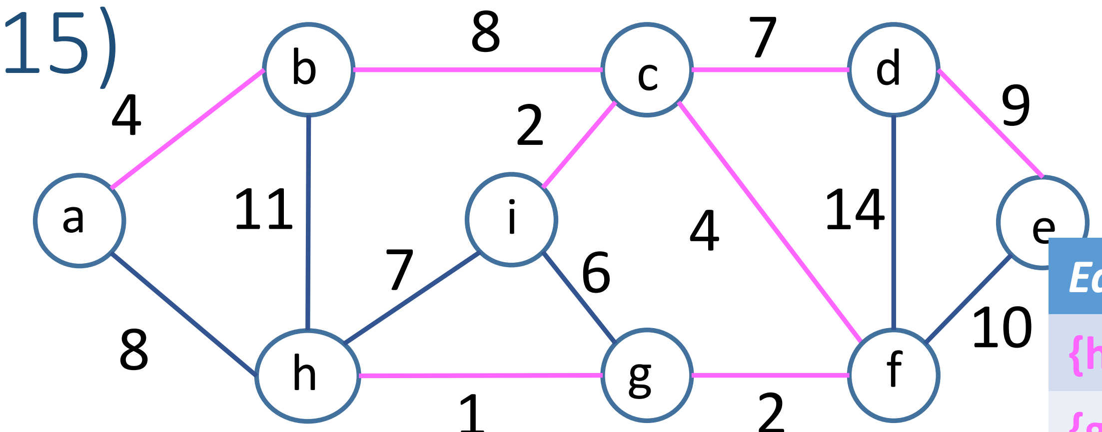
Step 7:

Step 8:

8: {h, g, f, c, i, a, b, d, e}

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST → Kruskal's algorithm (15)

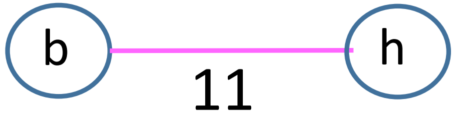



```
MST- KRUSKAL(G, w) {
1   A = ∅
2   for each vertex v ϵ G. V
3         MAKE-SET (v)
4   sort the edges of G. E into non-decreasing order by weight w
5   for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
6         if FIND-SET (u) ≠ FIND-SET (v)
7                 A = A ∪ {(u, v)}
8                 UNION (u, v)
9   return A
}
```

Step 4: Sort the edge → nondecreasing order

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

Step 5:



Step 6: ***FIND-SET (b) ≠ FIND-SET (h)***

8 ≠ 8

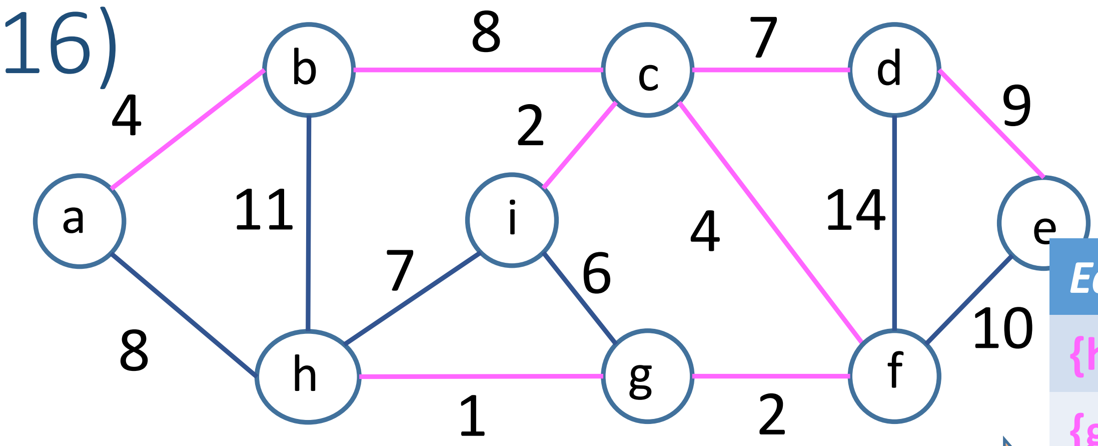
Step 2-3:

Step 7:

Step 8:

8: {h, g, f, c, i, a, b, d, e}

MST → Kruskal's algorithm (16)



Step 4: Sort the edge → nondecreasing order

Edges	*Weight*
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

MST- KRUSKAL(G, w) {

1 A = ∅

2 *for* each vertex *v* ϵ G. V

3 MAKE-SET (*v*)

4 sort the edges of G. E into *non-decreasing* order by weight *w*

5 *for* each edge (*u, v*) ϵ G. E, taken in *non-decreasing* order by weight

6 if FIND-SET (*u*) ≠ FIND-SET (*v*)

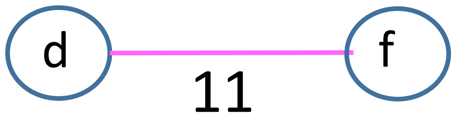
7 A = A ∪ {(u, v)}

8 UNION (*u, v*)

9 *return* A

}

Step 5:



Step 6: ***FIND-SET (d) ≠ FIND-SET (f)***

8 ≠ 8

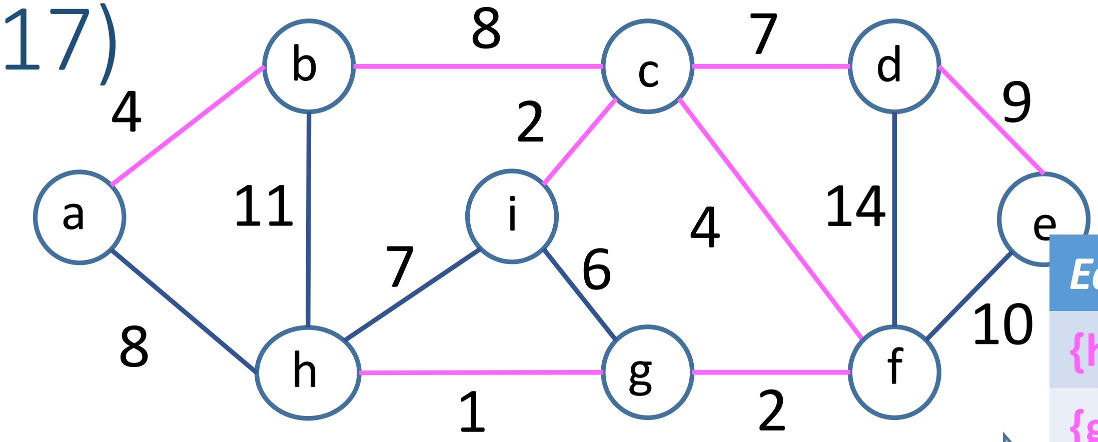
Step 2-3:

Step 7:

Step 8:

8: {h, g, f, c, i, a, b, d, e}

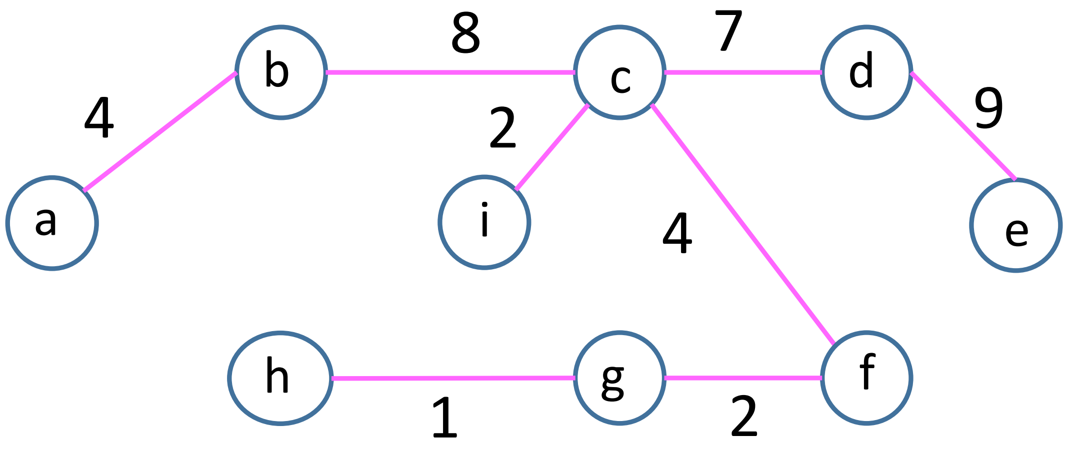
MST → Kruskal's algorithm (17)



Step 4: Sort the edge → nondecreasing order

MST- KRUSKAL(G, w) {
1 A = ∅
2 *for* each vertex *v* ϵ G. V
3 MAKE-SET (*v*)
4 sort the edges of G. E into ***non-decreasing*** order by weight *w*
5 *for* each edge (*u, v*) ϵ G. E, taken in ***non-decreasing*** order by weight
6 if FIND-SET (*u*) ≠ FIND-SET (*v*)
7 A = A ∪ {(u, v)}
8 UNION (*u, v*)
9 *return* A
}

Edges	Weight
{h, g}	1
{g, f}	2
{c, i}	2
{a, b}	4
{c, f}	4
{i, g}	6
{c, d}	7
{h, i}	7
{b, c}	8
{a, h}	8
{d, e}	9
{f, e}	10
{b, h}	11
{d, f}	14

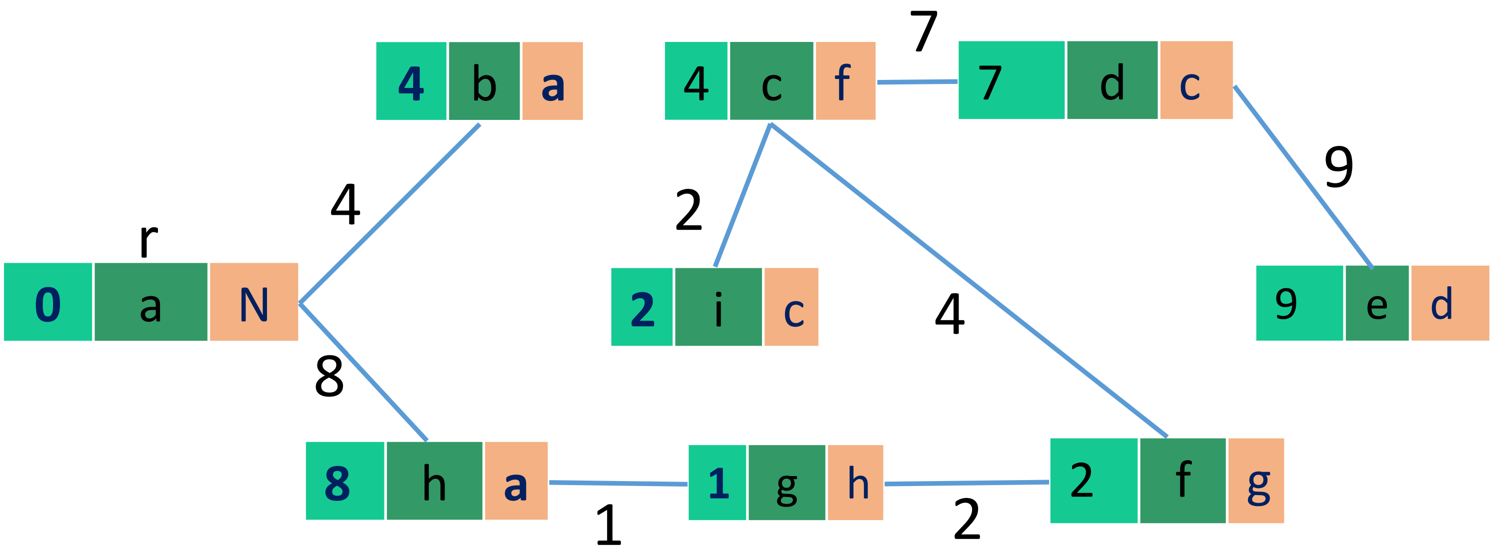


A= {(h, g), (g, f), (c, i), (a, b), (c, f), (c, d), (b, c), (d, e)}

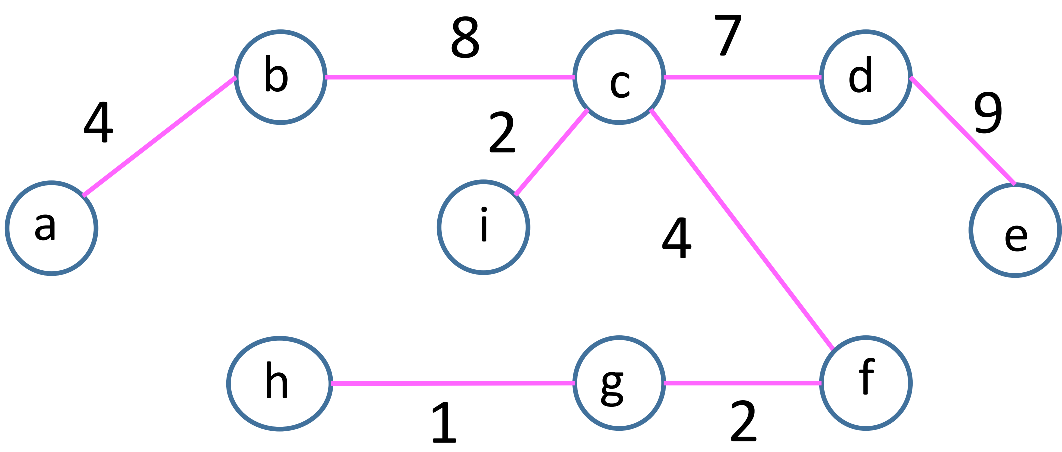
{h, g, f, c, i, a, b, d, e}

MST → Length of the edges = 37

MST → Prim's and Kruskal's algorithm



Prim's



Kruskal's

<u>MST → Length of the edges = 37</u>

MST → Kruskal's algorithm → Time complexity analysis

```
MST- KRUSKAL(G, w) {
  A = ∅
  for each vertex v ϵ G. V
          MAKE-SET (v)
  sort the edges of G. E into non-decreasing order by weight w
  for each edge (u, v) ϵ G. E, taken in non-decreasing order by weight
          if FIND-SET (u) ≠ FIND-SET (v)
                  A = A ∪ {(u, v)}
                  UNION (u, v)
  return A
}
```

O(1), for initialization of Set A

O(n) → n = # of vertices → O (V)

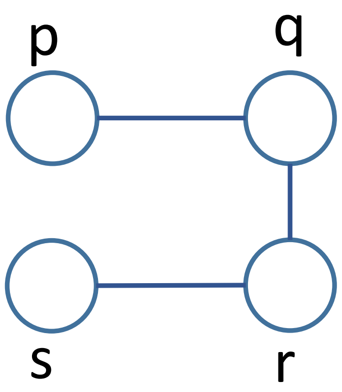
O(m logm) → m = # of Edge → O(E loge)

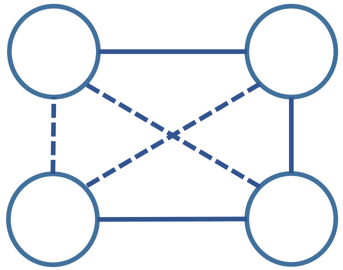
O(m), it has to execute for each edge

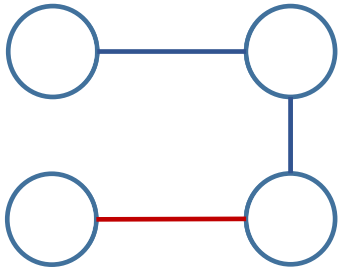
Total time complexity: <u>O(n) + O (m log m) + O(m) = O(m logm)</u>
Worst case → $m = n^2$
Total time complexity = O (m logm) = O (m logn^2) = O (2m logn) = O (m logn)

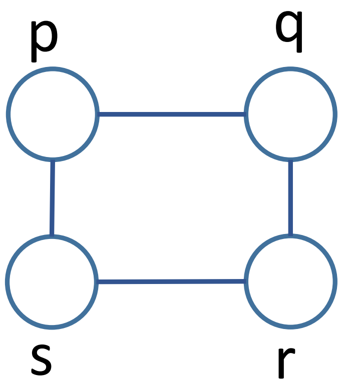
Properties of MST (1)

1) G = (V, E), spanning tree : *(V-1) edges*
2) If any edge is added to a spanning tree, then the new graph becomes cyclic
   - ✓ spanning tree is *maximally acyclic*
3) Every sapping tree is *minimally connected*
   - ✓ removal of an edge will disconnect the graph
4) There may be several *minimal spanning trees (MSTs) of the same weight*
5) If each edge has a distinct weight → *exactly one unique MST is possible*
   - ✓ Uniqueness
6) *Cycle-property: For any cycle C in graph, if the edge weight is larger then all other edges in C, then that edge cannot belongs to MST*



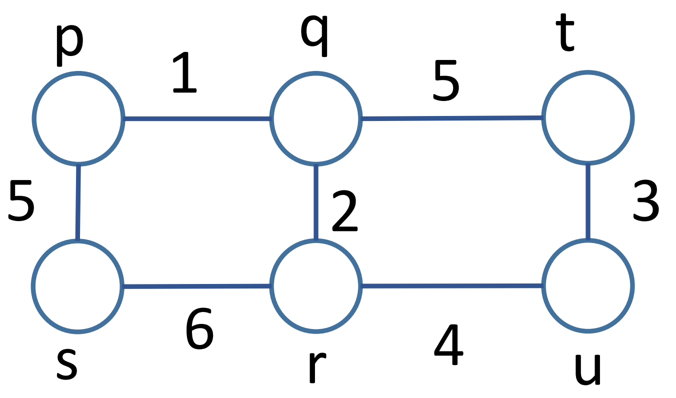




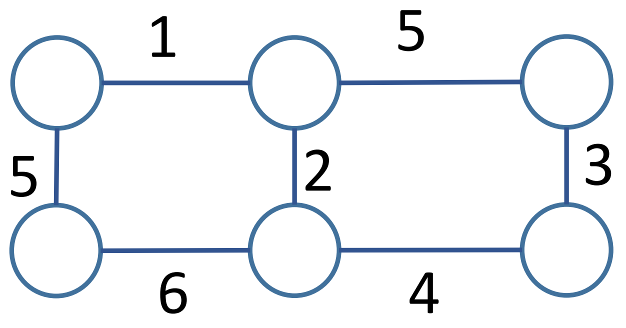


Properties of MST (2)

7) *Cut-property:* For any cut *Ct,* if the weight of an edge in the cut-set is strictly smaller all other weight of an edge in the cut-set → this edge must be present in MST



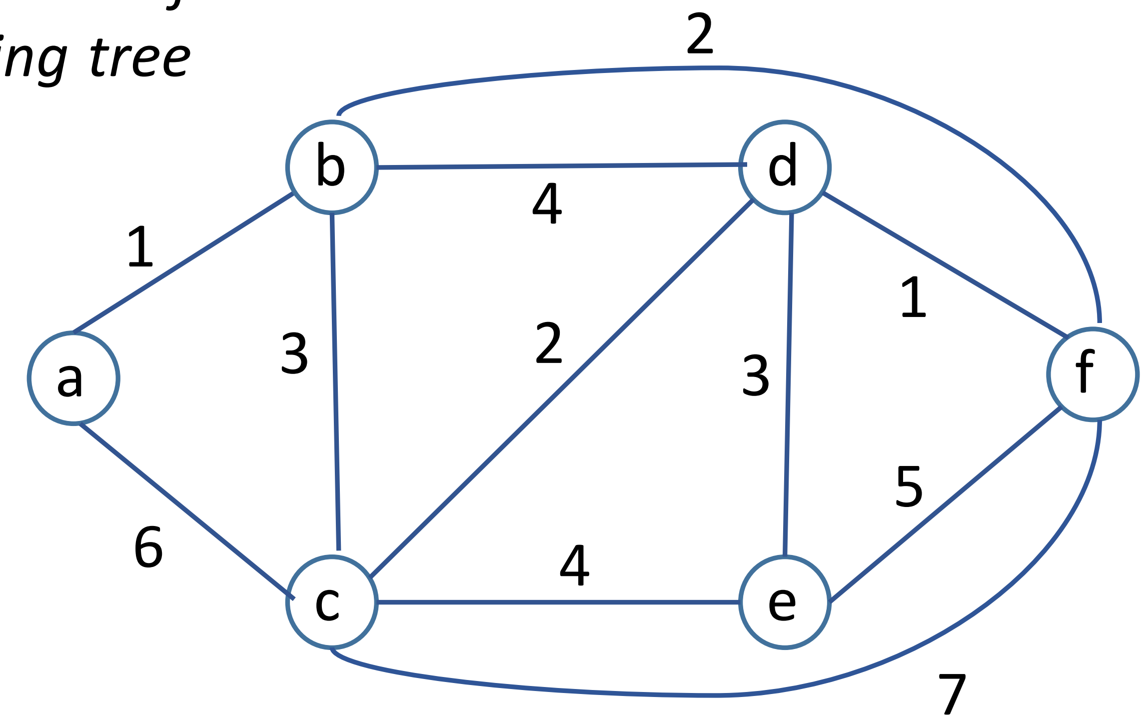
8) *Min cost edge- property:* if the min-weighted edge in G is unique → this edge must be present in MST



Exercise: MST Kruskal's algorithm

Consider the following graph:

*Which one of the <u>following cannot</u> be the sequence of edges added, **in that order**, to a minimum spanning tree using Kruskal's algorithm?*



A. {(*a*, *b*), (*d*, *f*), (*b*, *f*), (*d*, *c*), (*d*, *e*)}
B. {(*a*, *b*), (*d*, *f*), (*d*, *c*), (*b*, *f*), (*d*, *e*)}
C. {(*d*, *f*), (*a*, *b*), (*d*, *c*), (*b*, *f*), (*d*, *e*)}
D. {(d, f), (a, b), (b, f), (d, e), (d, c)}

thank you!

email:
k.kondepu@iitdh.ac.in