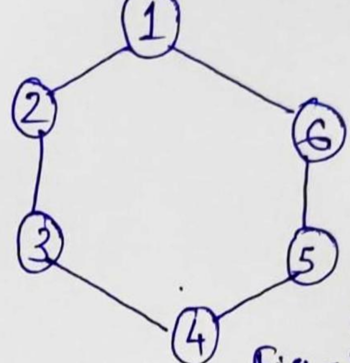
# Minimum Spanning tree (MST)

①

A graph (G) can be represented as a set of vertices (V) / nodes & Edges (E).

$\therefore G = \{V, E\}$




$|V| = 6$
$|E| = 6$

Fig-1.

Edges can be directed or undirected & they can also have weight attribute.

**Connected Graph:** A graph is connected when there is a path between every pair of vertices.

**Cycle:** A path that start & ends with the same vertex.

**Tree:** It is a connected acyclic graph (i.e. a graph with no cycle).

Now, Let's consider the graph in figure-1. There,—

$G = \{V, E\}$

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (2,3), (3,4), (4,5), (5,6), (6,1)\}$

block (I)

**Spanning Tree:** A Spanning Tree is a subgraph ($G'$) of a graph (G).

It implies that we should take subset of the vertices & edges to generate a spanning tree.

But, The question is we should take subset of both vertices & edges. ?

The Answer to that question is 'NO.'

$\Rightarrow$ In spanning tree the total number of vertices should remain the same, i.e. we have to consider all the vertices. Number of vertices can be denoted as $|V|$.

$\Rightarrow$ However the number of edges will be equal to $(|V|-1)$. This indicates that each vertices is connected exactly one.

If we consider Figure-2. It is a spanning tree of Figure-1,

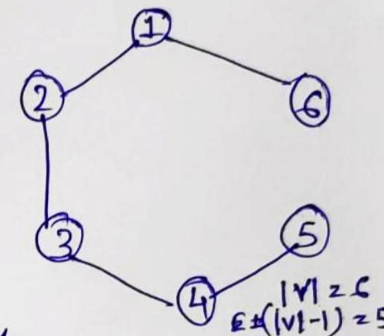



Figure-2

Here

$$\left.\begin{aligned} G' &= \{V', E'\}. \\ V' &= \{1, 2, 3, 4, 5, 6\}. \\ E' &= \{(1,2), (2,3), (3,4), (4,5)\} \end{aligned}\right\} \text{block (II)}$$

Now, if we compare block-I & block-II then we can see that $V' = V$ but $E' \neq E$.

<u>Definition:</u> Formally, for a graph $G = \{V, E\}$, the spanning tree is $E' \subseteq E$ such that:

$$\exists u \in V : \{(u, v) \in E' \lor (v, u) \in E'\} \ \forall v \in V$$

<u>Alternative Def^n:</u> A graph $G' = \{V', E'\}$ is a spanning tree of a graph $G = \{V, E\}$ if $V = V'$ & $E' \subseteq E$ and $G'$ does not contain any cycle. Further the number of edges in $G'$ ~~should be~~ is given by, $|E'| = |V| - 1$.

Now Let's consider Figure-2 again. is that the only possible spanning tree?
Answer is 'NO.' there are many possibilities. which are given below (some of them).

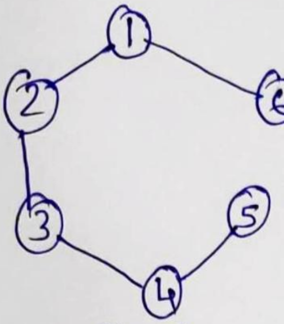


Fig-3

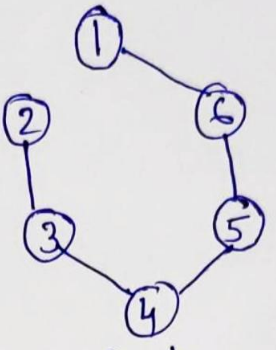


Fig-4

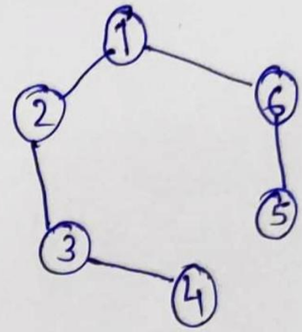


Fig-5

All the above variants are spanning tree.
[TASK: check if it satisfies the definition given earlier]

Thus we can see that for a given graph (undirected) there might be many possible spanning trees. The total number of possible spanning tree can be calculated by the following formula:

$$\text{no. of spanning tree} = {}^{|E|}C_{(|V|-1)} - \text{no. of cycles.}$$

Excercise: Calculate the total number of spanning tree for the following graph, -

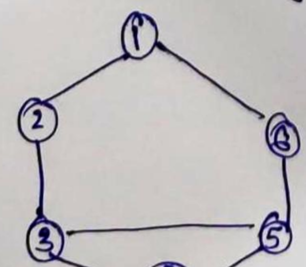


Fig - 6.

no. of cycle = 2.

$\therefore N = {}^{7}C_{5} - 2$ $\left[ \text{Formula: } {}^{n}C_{r} = \frac{n!}{r!\,(n-r)!} \right]$

For Figure-1 the total no. of spanning tree will be. ${}^{6}C_{5} = \frac{6!}{5!\,1!} = \underline{6}$.

Consider a weighted graph depicted in Figure-7.

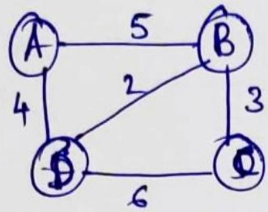
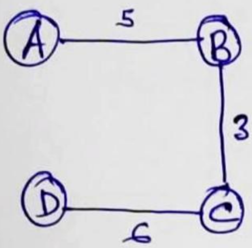
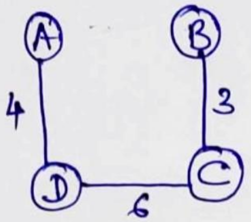



Figure-7

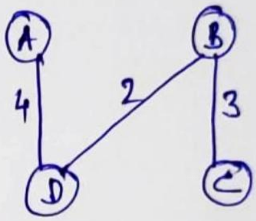
What If we consider the different spanning trees of the graph in Figure-7, ~~then~~




Cost = 14
(Figure-8)




Cost = 13
(figure-9)




Cost = 9.
(Figure-10).

Figure-8, 9, & 10, represents the minimal spanning tree for the graph-7. From there we can see that in Figure-10 we can reach all the nodes with minimum cost 9 {what we called as minimal spanning tree}.

Now, we find out the minimal spanning tree manually, is there any method we can apply to find out the minimal spanning tree?

Answer is, - 'YES'. These are

① - Kruskal Algorithm. &

② - Prim's Algorithm.

① **Kruskal's Algorithm:** Kruskal's Algorithm adds the cheapest edge which does not create a cycle.

## Algorithm

Step 0: set $A = \phi$ & $F = E$, the set of all the edges

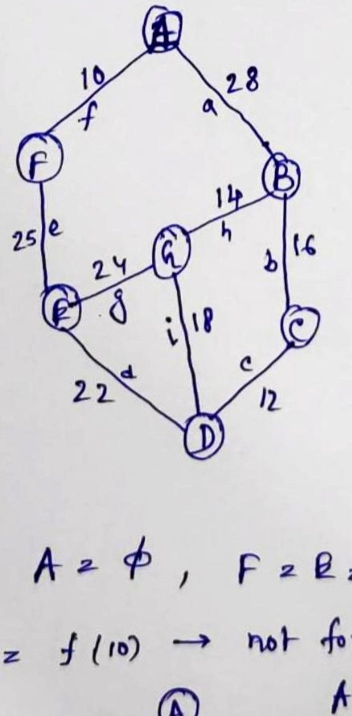
Step 1: choose an edge 'e' in F of minimum weight and check whether adding 'e' to 'A' creates a cycle

Step 1.1: If YES — remove e from F

Step 1.2: If NO — move 'e' from F to A.

Step 2: If $F = \phi$ or, $|A| = |V|-1$, stop and output the minimal spanning tree (V, A)
otherwise go to Step 1.

## Example




$G = \{V, E\}$

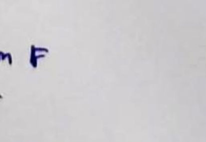
$V = \{A, B, C, D, E, F, G\}$

$E = \{(A,B), (B,C), (C,D), (D,E), (E,F), (F,A), (F,G), (G,B), (G,H)\}$.

or, $E = \{a, b, c, d, e, f, g, h\}$

Initially, — $A = \phi$, $F = E = \{a, b, c, d, e, f, g, h, i\}$.

① $\min(F) = f(10) \rightarrow$ not forming cycle.




(not forming cycle).

- remove from F
- Add it to A.

$A = \{f\}$

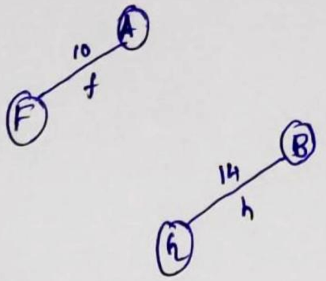
$F = \{a, b, c, d, e, g, h, i\}$

~~$F \neq \phi$ & $|A| \neq (|V|-1) = (7-1) = 6$~~

$F \neq \phi$ & $|A| \neq [(|V|-1) = (7-1) = 6]$ thus, proceed to next step.

2. min (f) = h (14). → ~~not forming a cycle~~

(not forming cycle)



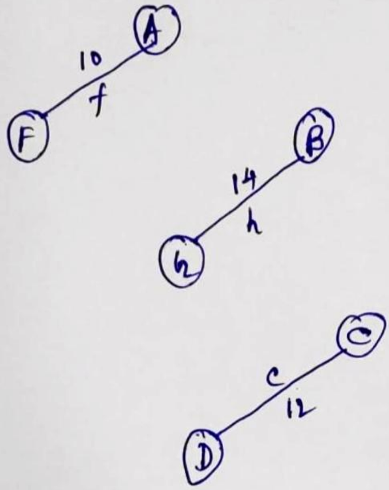
A = {f, h}

F = {a, b, c, d, e, g, i}.

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

3. min (F) = c (12) -

(not forming cycle)



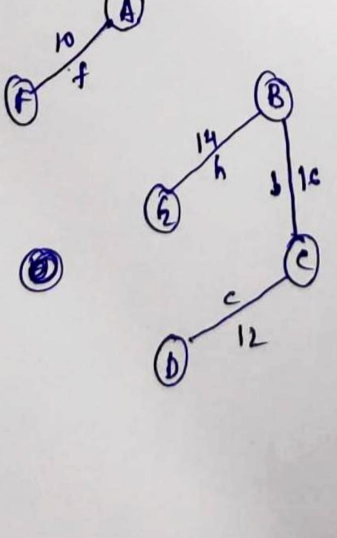
A = {f, h, c}

F = {a, b, d, e, g, i}.

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

4. min (F) = b (16)

(not forming cycle)

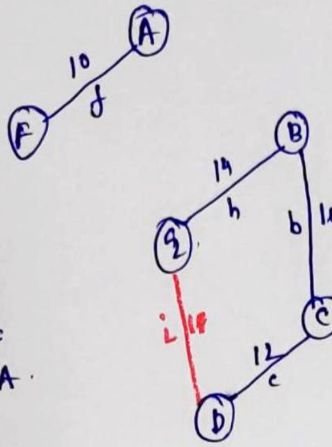


A = {f, h, c, b}

F = {a, d, e, g, i}

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

5 min (F) = i (18)




(forming cycle)

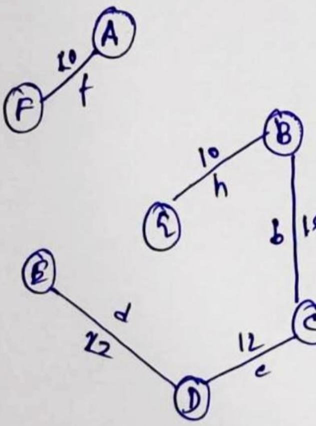
- ignore i (18)
- remove from F
- Don't add to A.

A = { f, h, c, b }

F = { a, d, e, g }

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

6 min (F) = d (22)



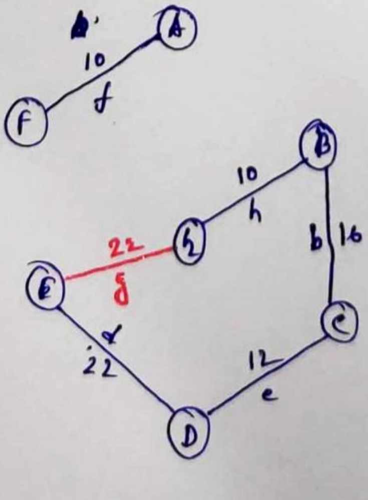

(not forming cycle)

A = { f, h, c, b, d }

F = { a, e, g }.

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

7 min(F) = g (24)



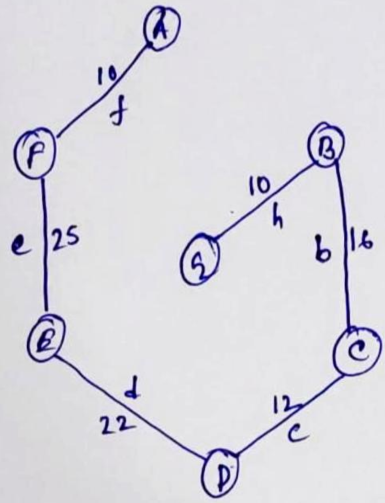

(forming cycle)

A = { f, h, c, b, d }

F = { a, e }.

F ≠ ∅ & |A| ≠ 6 thus proceed to next step.

8 $\min(F) = e(25)$




$A = \{f, h, c, b, d, e\}$

$F = \{a\}$.

$F \neq \phi$ but $\underline{|A| = 6}$ thus stop.

This is the minimal spanning tree.

$G' = \{V, A\}$

$V = \{A, B, C, D, E, F, G\}$

$A = \{f, h, c, b, d, e\}$.

## Time Complexity : $O(E \log E)$ or, $O(E \log V)$.

- Sorting edges takes $O(E \log E)$ time.
- After sorting we iterate through all edges and apply find union algorithm, which can take $O(E \log V)$ time.

Thus overall complexity $= O(E \log E + E \log V)$

$\simeq O(E \log E)$ or $O(E \log V)$.

## Detailed Analysis :

```
Sort E in increasing order by weight w. --- O(E log E)
A = Φ.
for each u in V
{
  Create set(u)         ---------------- O(V)
}
for e_i = (u_i v_i) from 1 to |E|
{
  if (Findset(u_i) != Findset(v_i))
  {
     add {u_i v_i} to A;           ------ O(E log V)
     Union (u_i v_i);
  }
}
Return (A);
```

Total $= O(E \log E) + O(V) + O(E \log V)$

$\simeq O(E \log E + E \log V)$

$\simeq O(E \log E)$ or $O(E \log V)$.

② **Prim's Algorithm**: It grows a single tree and adds a light adds a light edge in each iteration.

**Algorithm:**

Step 0: Choose any element $r$; set $S = \{r\}$ and $E' = \phi$. (i.e. Take '$r$' as the root of our spanning tree).

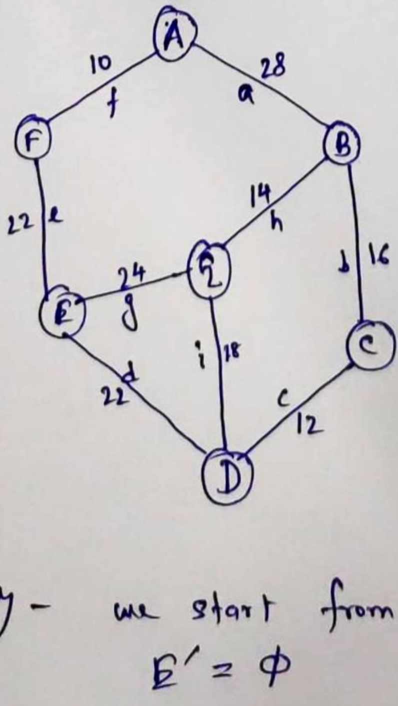
Step 1: Find a lightest edge such that one endpoint is in $S$ & the other is in $(V-S)$.

Step 1.1: Add ~~other endpoint~~ this edge $(r, x)$ where $x \in (V-S)$ to $E'$.

Step 1.2: Add other endpoint '$x$' to $S$.

Step 2: if $(V-S) \neq \phi$, then stop & output minimum spanning tree ~~(S, E)~~. $(S, E')$.
otherwise go to Step 1.

**Example:**




$G = \{V, E\}$

$V = \{A, B, C, D, E, F, G\}$

$E = \{(A,B), (B,C), (C,D), (D,E), (E,F), (F,A), (E,G), (G,B), (G,D)\}$.

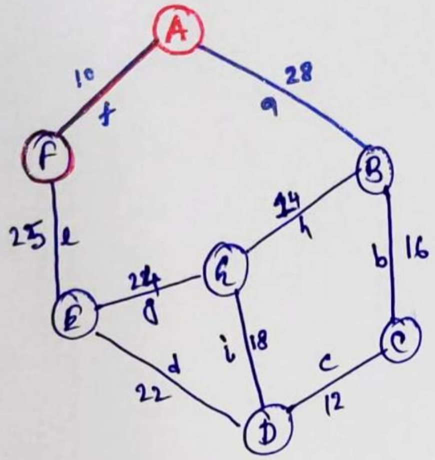
or,

$E = \{a, b, c, d, e, f, g, h, i\}$.

Initially - we start from $A$. $\therefore S = \{A\}$

$E' = \phi$

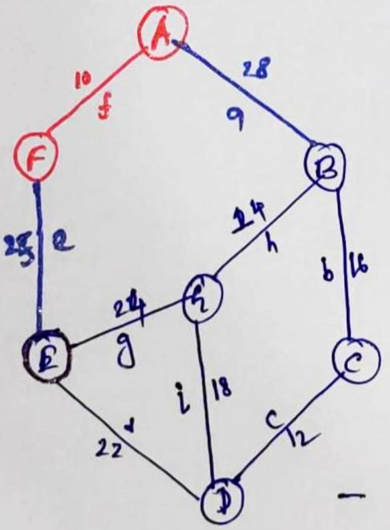
1



→ Check the adjacent ~~nodes~~ edges from a
→ here it is f, a.
→ f has the min weight
→ select f.

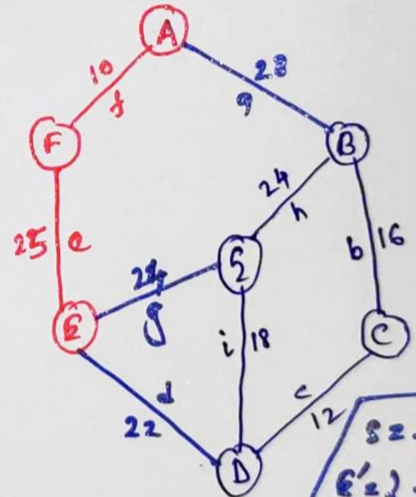
$\boxed{\begin{array}{l} S = \{A, F\} \\ E' = \cancel{\phi}\ \{f\} \end{array}}$

2



⇒

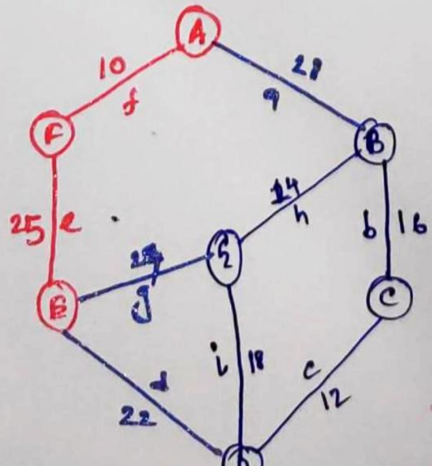
3



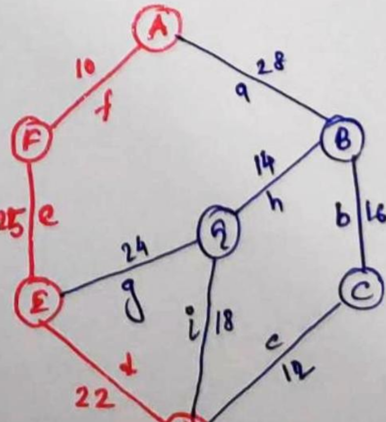
$\boxed{\begin{array}{l} S = \{A, F, E\} \\ E' = \{f, e\} \end{array}}$

- S = {A, f} have connected nodes e, a.
- 'e' is smaller, so 'e' is selected.

3



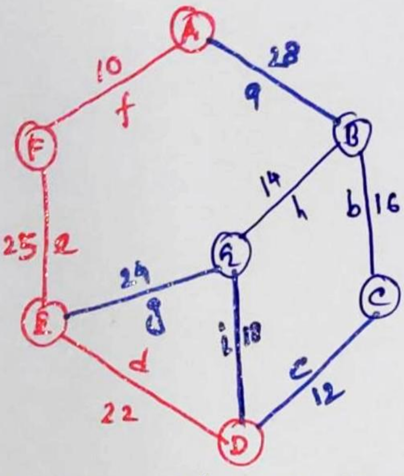
⇒



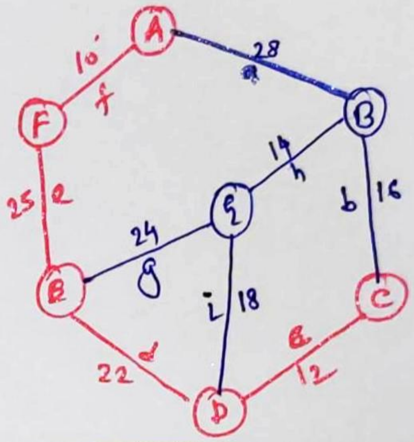
- S = {A, F, E, ~~●~~} have connected ~~nodes~~ edges {a, g, d}.
- 'd' is the smaller, select d.

⇒ $\boxed{\begin{array}{l} S = \{A, F, E, D\} \\ E' = \{f, e, d\} \end{array}}$

(4)



⇒

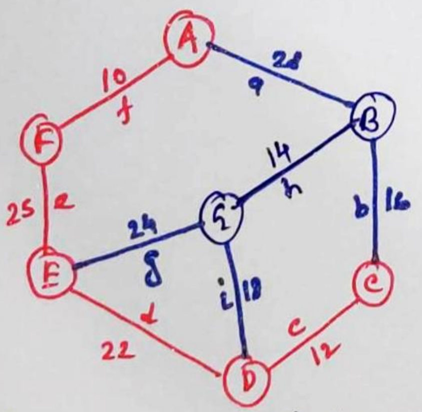


- S = {A, F, E, D} have connected edges {c, g, a, i}
- c is smaller, select c.

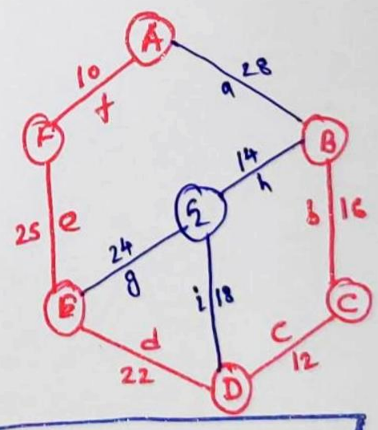
⇒

$$\boxed{\begin{array}{l} S = \{A, F, E, D, C\} \\ E' = \{f, e, d, c\}. \end{array}}$$

(5)



⇒

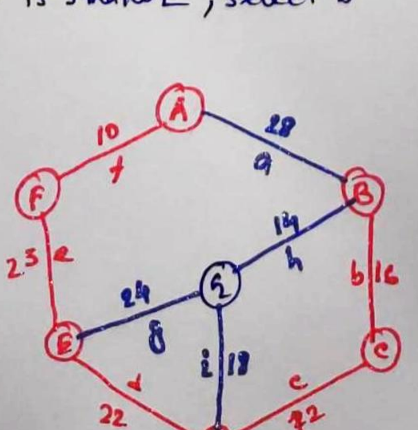


- S = {A, F, E, D, C} have connected edges {g, i, b, a}.
- b is smaller, select b

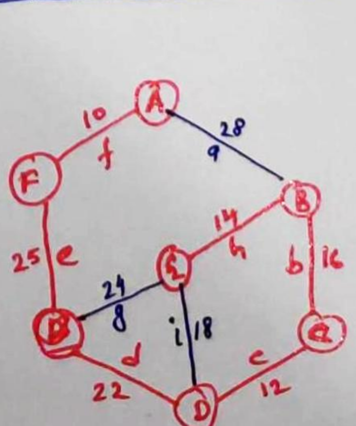
⇒

$$\boxed{\begin{array}{l} S = \{A, F, E, D, C, B\}. \\ E' = \{f, e, d, c, b\}. \end{array}}$$

(6)



⇒



- S = {A, F, E, D, C, B} have connected edges {g, h, i, a} → h is smaller.

⇒ S = {A, F, E, D, C, B, G}
E' = {f, e, d, c, b, h}

$O(\log n)$ to extract each vertex from the queue.
Done once for each vertex = $O(n \log n)$

$O(\log n)$ time to decrease the key value of neighbouring vertex

Done atmost once for each edge = $O(e \log n)$.

total cost = $O(n \log n + e \log n)$

$= O((n+e) \log n)$.