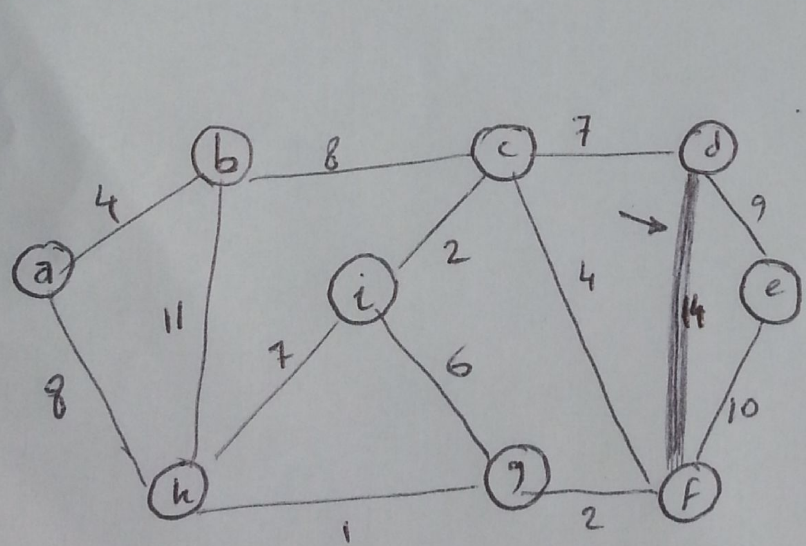
*Kruskals's Algorithms:-

Kruskal's Algorithm finds a safe edge to add to the growing forest by finding, of all edges That connect any two trees in the forest an edge (u,v) of heighest weight.

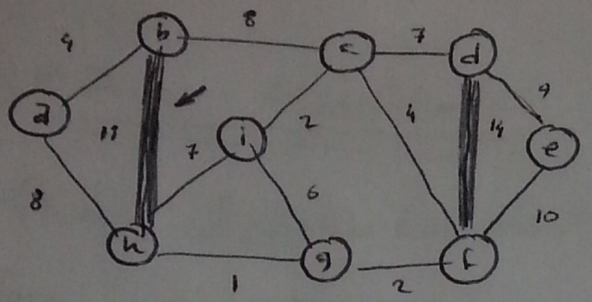
Let C_1, C_2 denote the two trees that are connected by (u,v) Since (u,v) must be high weight edge connecting C_1 to some other tree.

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

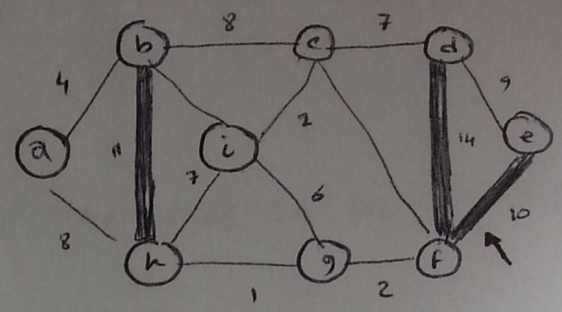
(a)



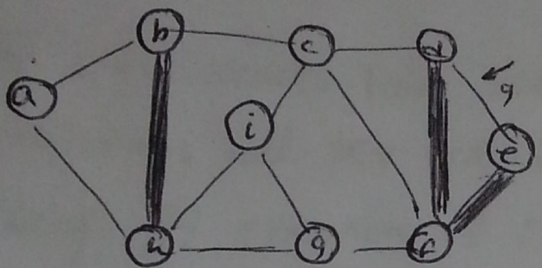
(b)


a
b
c
d
e
f
g
h
i
4
8
7
9
11
7
2
4
14
8
6
10
1
2

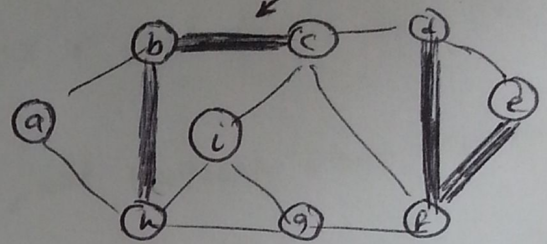
(c.)


a
b
c
d
e
f
g
h
i
4
8
7
9
11
7
2
6
14
8
10
1
2

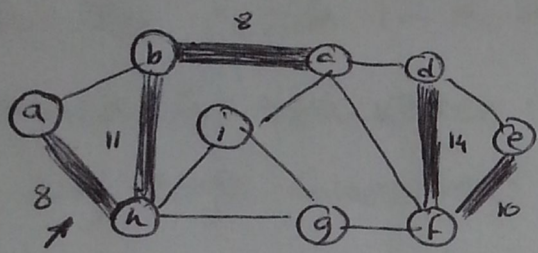
(d)


a
b
c
d
e
f
g
h
i
9

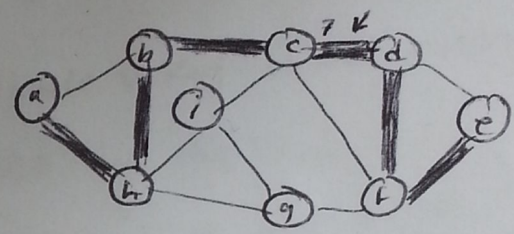
(e)


a
b
c
d
e
f
g
h
i

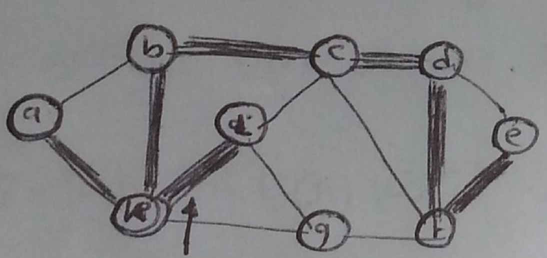
(f)


a
b
c
d
e
f
g
h
i
8
11
14
8
10

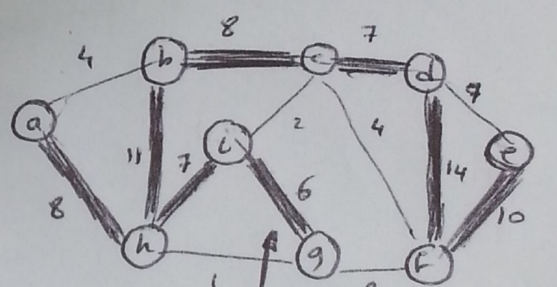
(g)


a
b
c
d
e
f
g
h
i
7

(h)


a
b
c
d
e
f
g
h
i

(i)


a
b
c
d
e
f
g
h
i
4
8
7
9
11
7
2
4
14
8
6
10
1
2

* Prim's Algorithm:

- Prim's algorithm has the property that the edges in the set A always form a single tree.
- The tree starts form arbitrary vertix r and grows until the tree spans all the vertices in V.

During the execution of the algorithm, all vertices that are not in the tree reside in a max-priority queue Q based on a key attribute. For each vertex v, the attibute v.key is the maximum weight of any edge connecting v to a vertex in the tree.

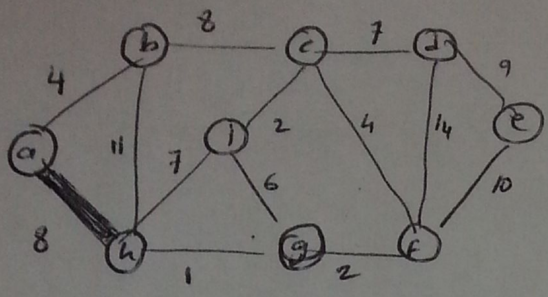
by convention $v.key = -\infty$ if there is no such edge.

the attribut $v.\pi$ names the parent of v in the tree.

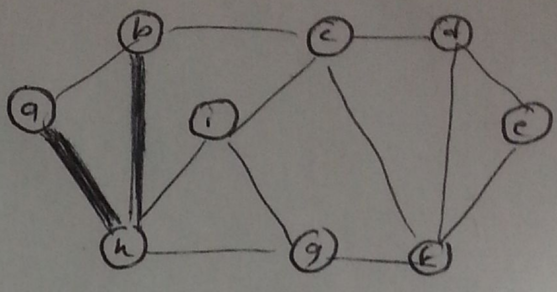
MST-PRIM (G, w, r)

```
for each u ∈ G.V
    u.key = -∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = Extract-Max(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u,v) > v.key
            v.π = u
            v.key = w(u,v)
```

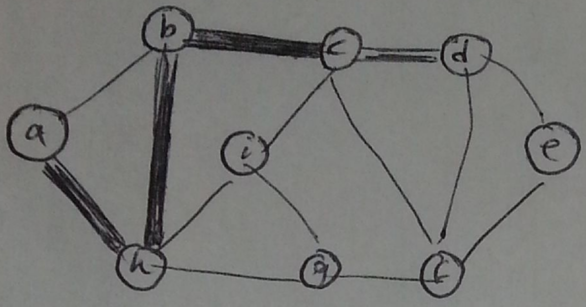
(a)



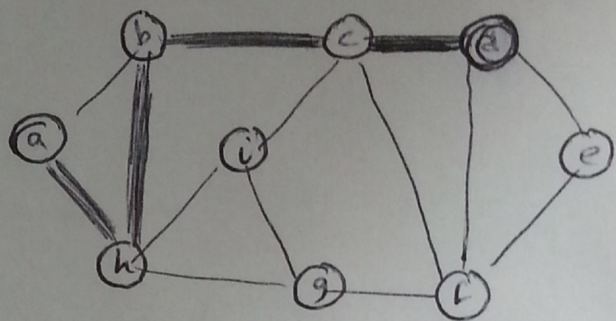
(b)



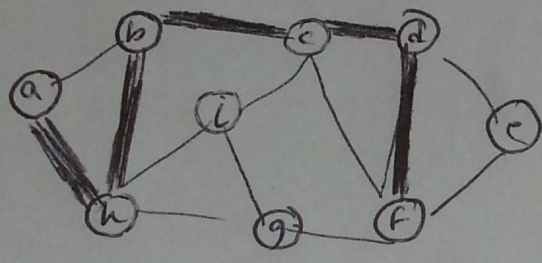
(c)



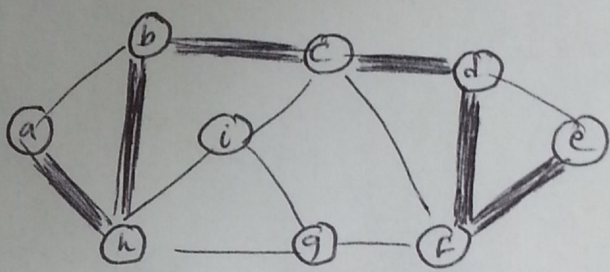
(d)



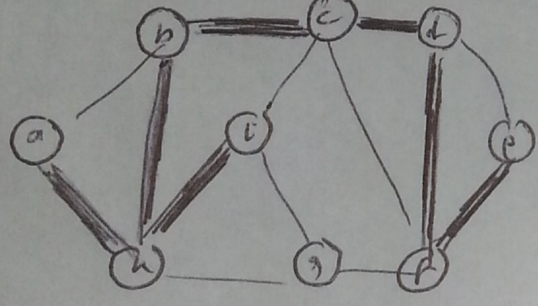
(e)



(f)



(g)



(h)