Minimum Spanning Trees algorithm

Algoritma Minimum Spanning Trees

algoritma Kruskal and algoritma Prim.

Kedua algoritma ini berbeda dalam metodologinya, tetapi keduanya mempunyai tujuan menemukan minimum spanning

- algorithm Kruskal menggunakan edge, dan
- algorithm Prim menggunakan vertex yang terhubung

Perbedaan antara algoritma prim dan kruskal

Perbedaan prinsip antara algoritma prim dan kruskal adalah,

jika pada algoritma prim sisi yang dimasukkan ke dalam T harus bersisian dengan sebuah simpul di T, maka pada algoritma kruskal sisi yang dipilih tidak perlu bersisian dengan sebuah simpul di T. asalkan penambahan sisi tersebut tidak membentuk cycle.

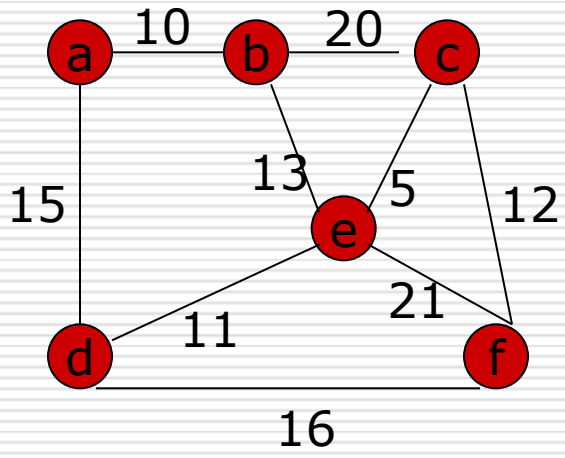
Kruskal's Algorithm:

Pada algoritma kruskal, sisi (edge) dari Graph diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar.

Sisi yang dimasukkan ke dalam himpunan T adalah sisi graph G yang sedemikian sehingga T adalah Tree (pohon). Sisi dari Graph G ditambahkan ke T jika ia tidak membentuk cycle.

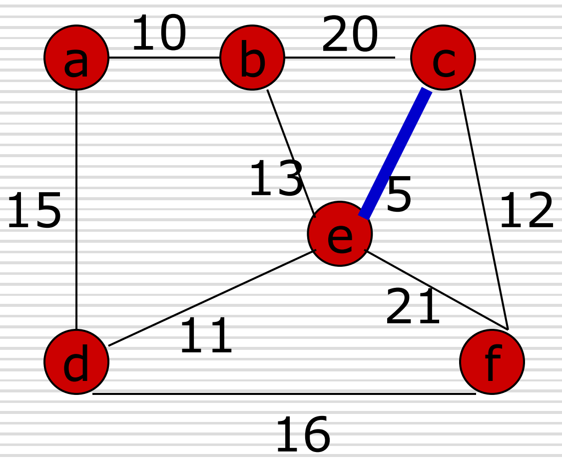
1. T masih kosong
2. pilih sisi (i,j) dengan bobot minimum
3. pilih sisi (i,j) dengan bobot minimum berikutnya yang tidak membentuk cycle di T, tambahkan (i,j) ke T
4. Ulangi langkah 3 sebanyak (n-2) kali.
5. Total langkah (n-1) kali

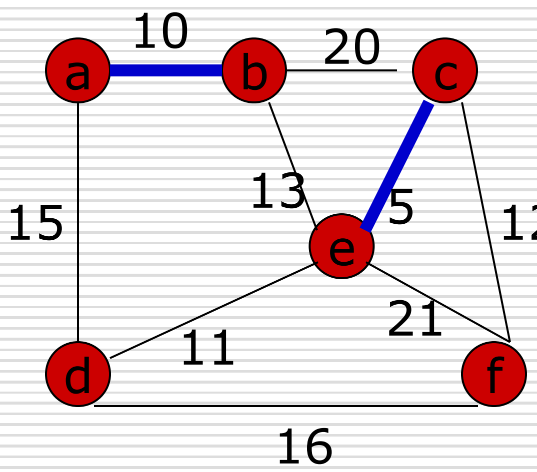
Kruskal's Algorithm:

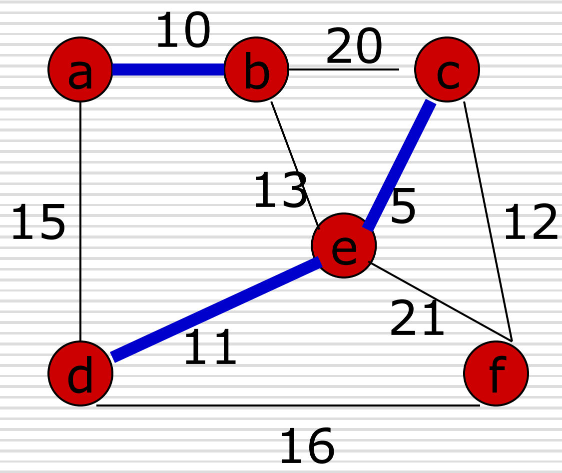


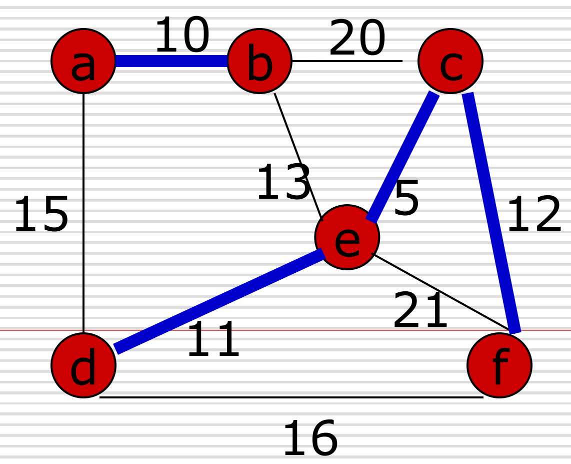
Langkah	Sisi	bobot
0		
1	e-c	5
2	a-b	10
3	d-e	11
4	c-f	12
5	b-e	13

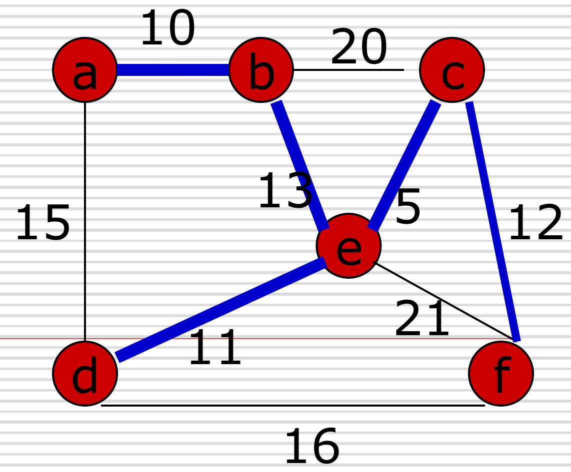
Kruskal's Algorithm:



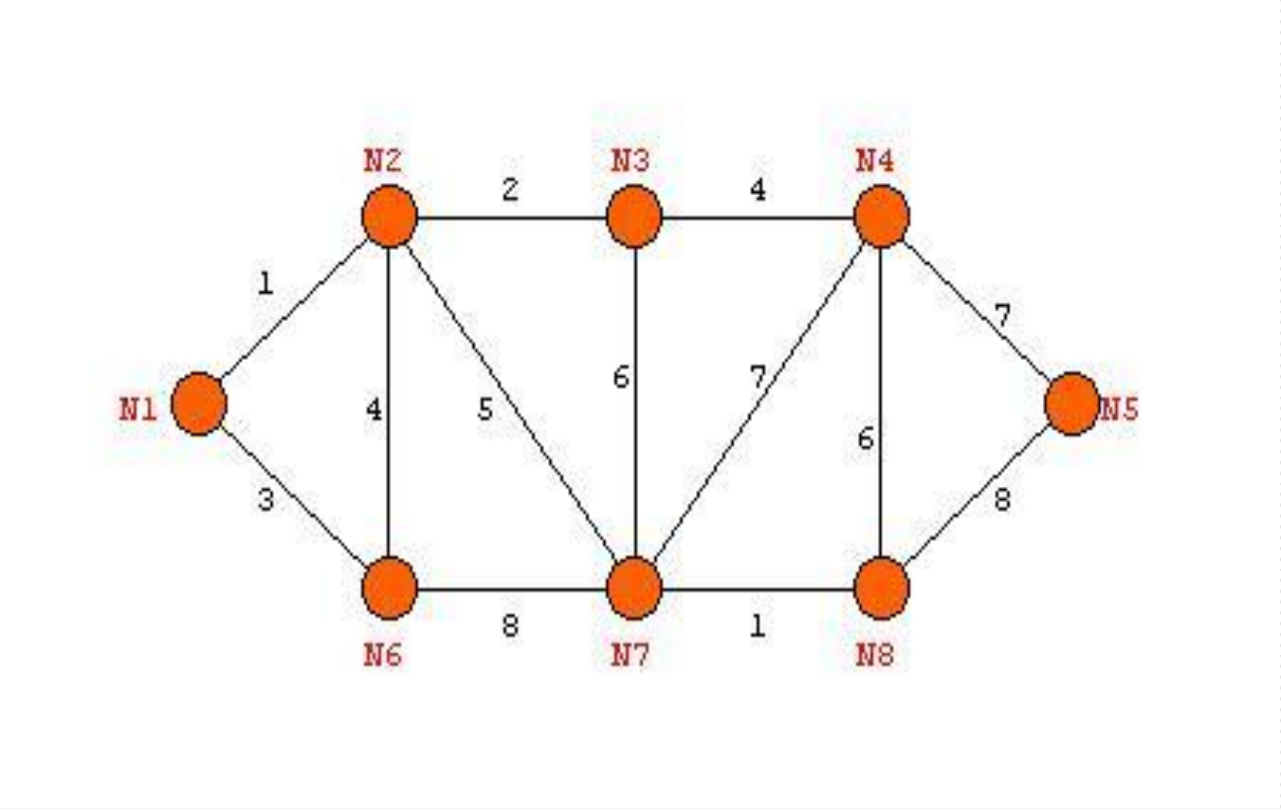




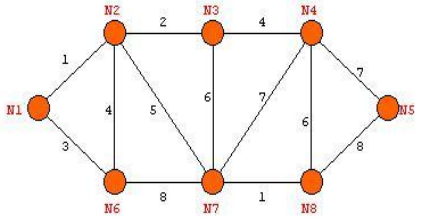




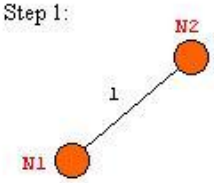
Contoh algoritma Kruskal



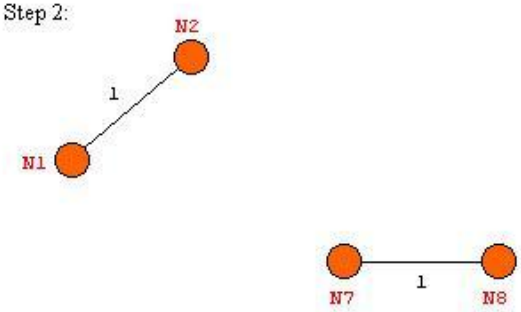
Contoh algoritma Kruskal



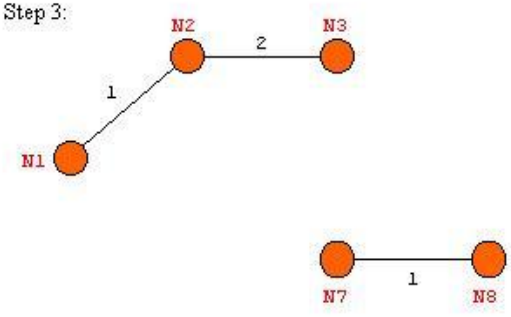
Step 1:



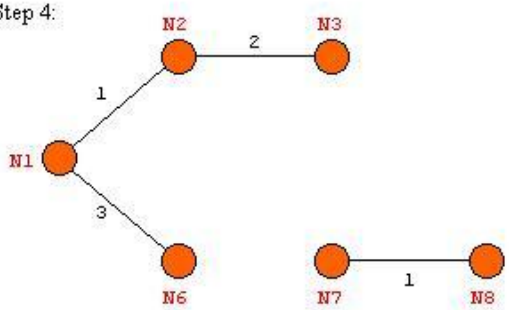
Step 2:



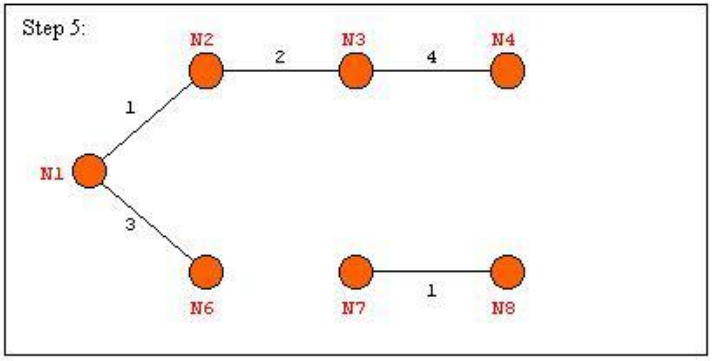
Step 3:

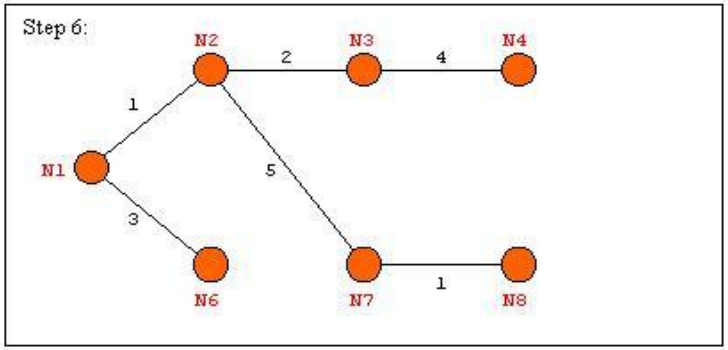


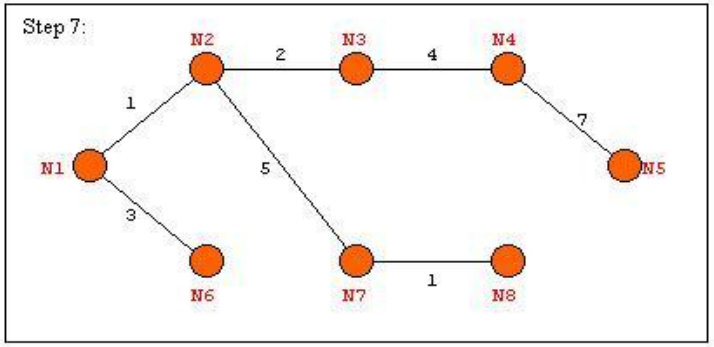
Step 4:



Contoh algoritma Kruskal

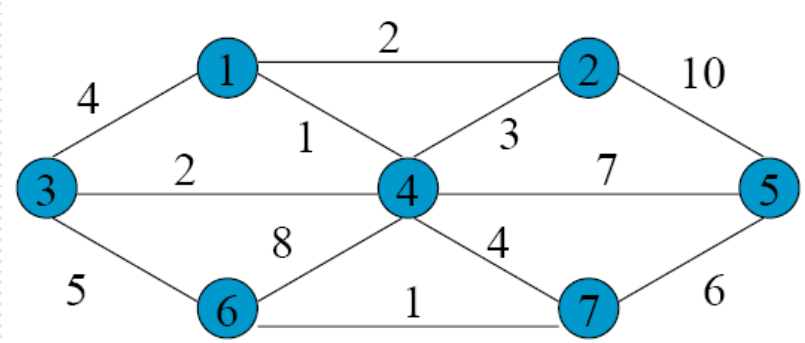






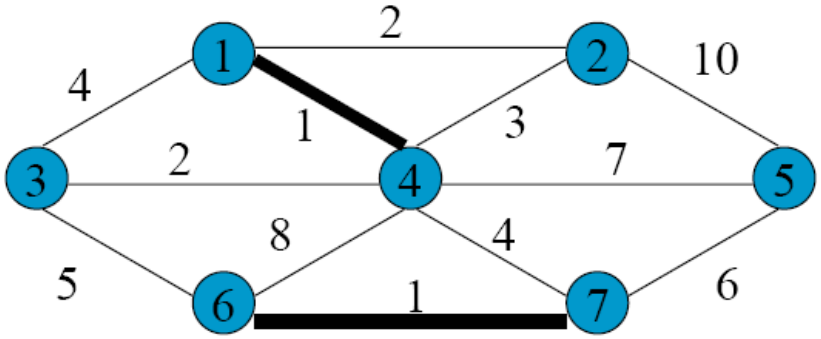
Langkah	Sisi	bobot
0		
1	N1,N2	1
2	N7,N8	1
3	N2,N3	2
4	N1,N6	3
5	N3,N4	4
6	N2,N7	5
7	N4,N5	7

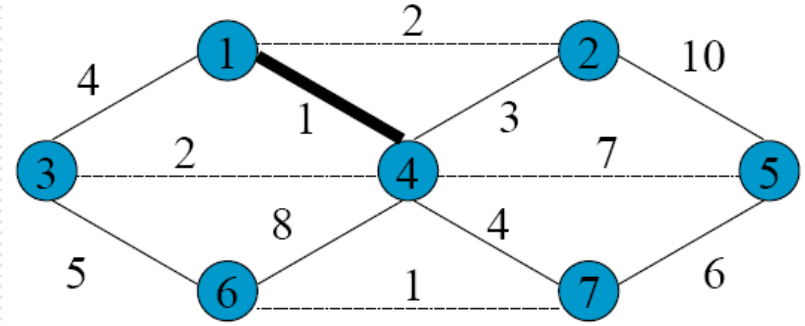
Contoh algoritma Kruskal



Langkah 1

Langkah 2

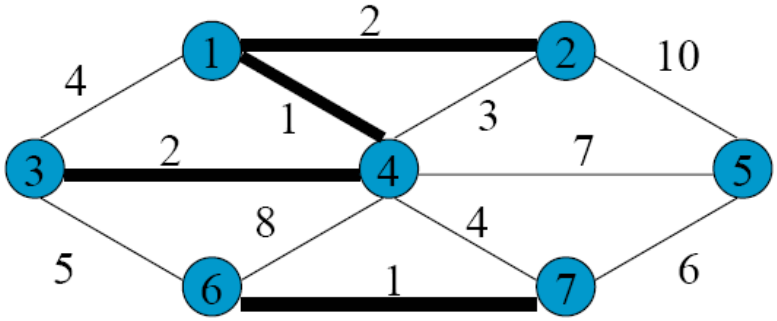


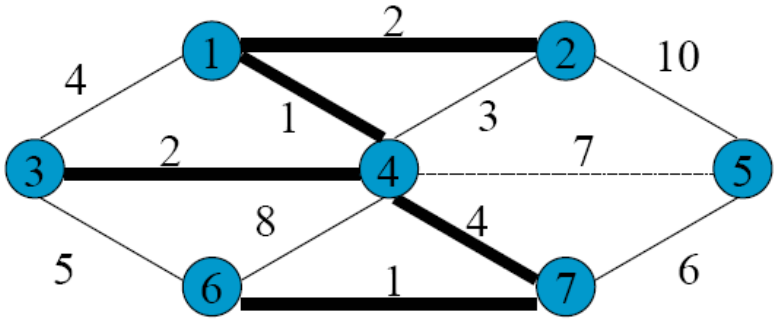


Langkah 3

Contoh algoritma Kruskal

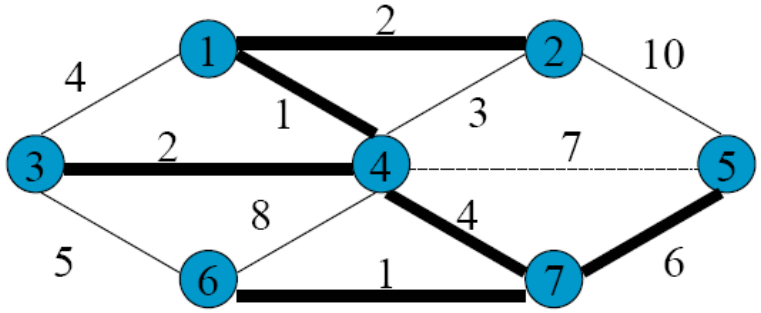
Langkah 4





Langkah 5

Langkah 6

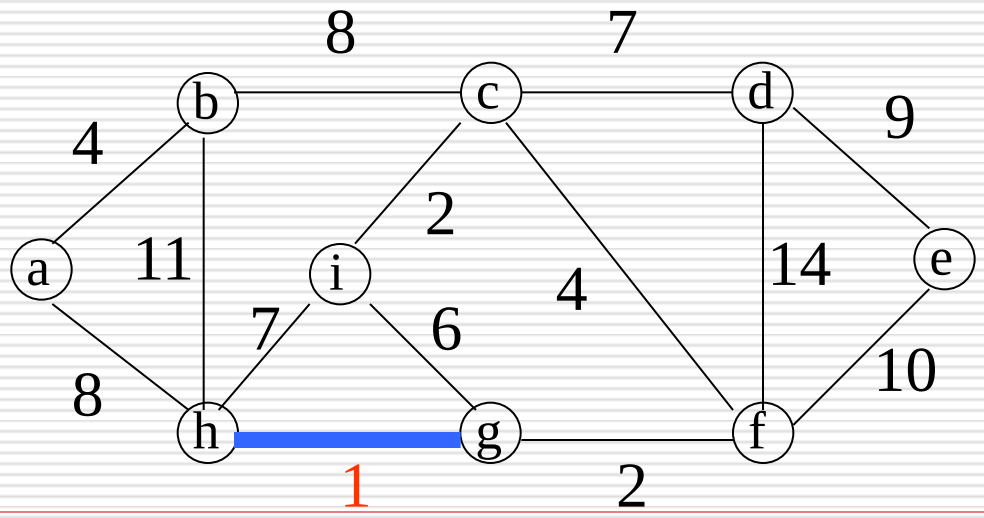


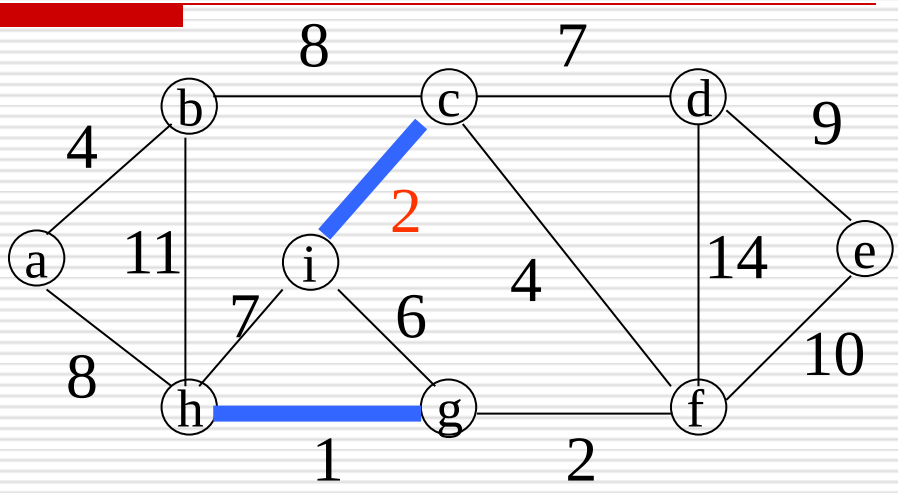
Langkah	Sisi	bobot
0		
1	1,4	1
2	6,7	1
3	3,4	2
4	1,2	2
5	4,7	4
6	5,7	6

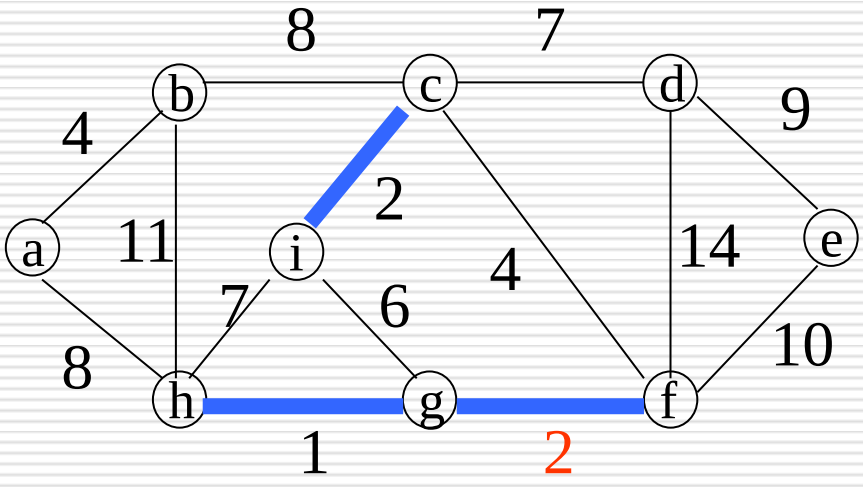
The execution of Kruskal's algorithm (Moderate part)

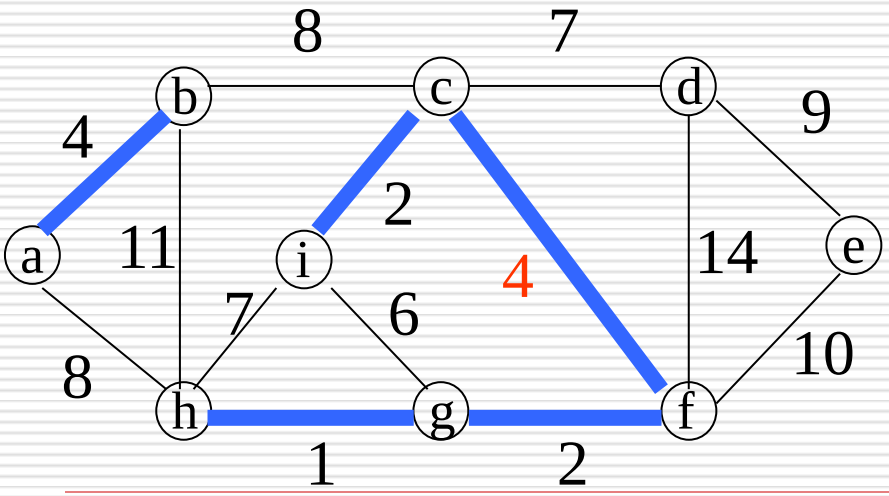
•The edges are considered by the algorithm in sorted order by weight.

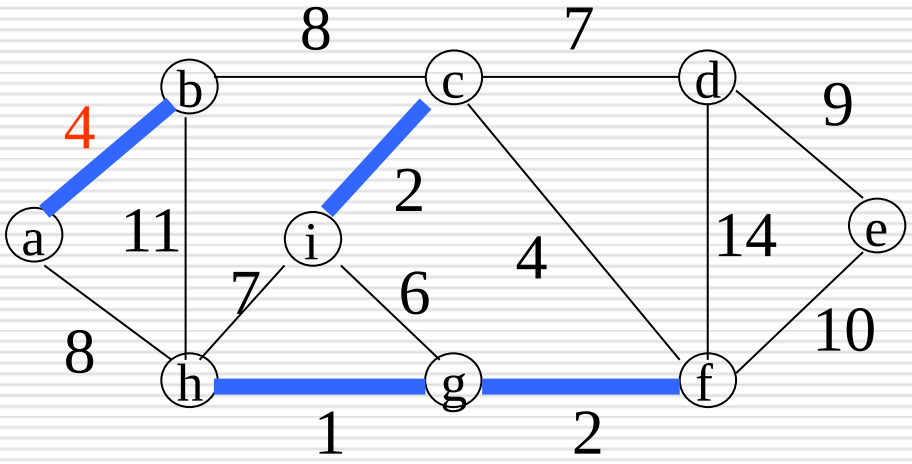
•The edge under consideration at each step is shown with a red weight number.

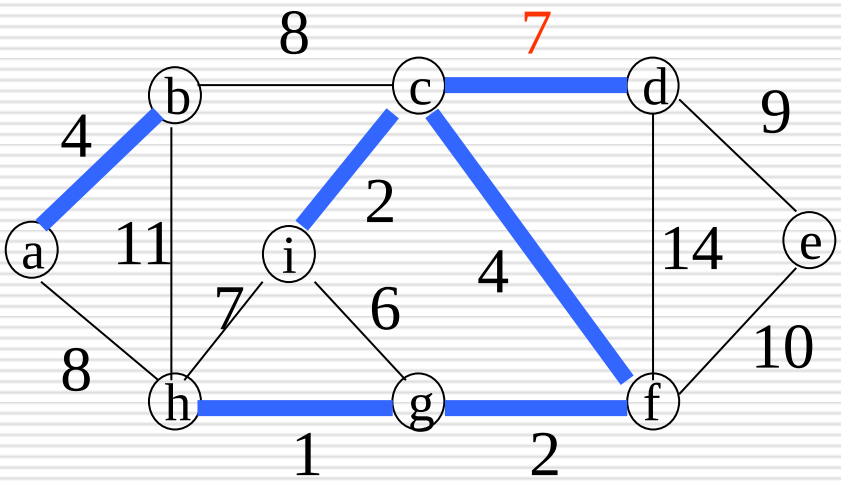


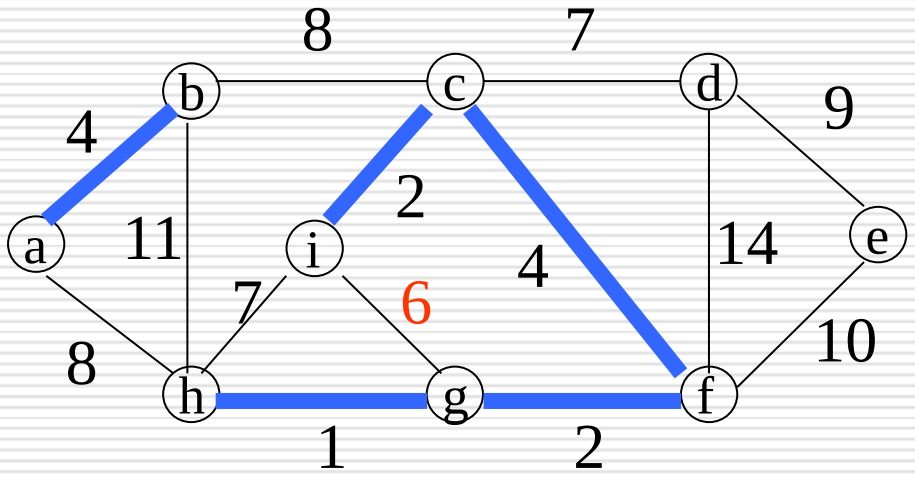

8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
10
8
h
g
f
1
2

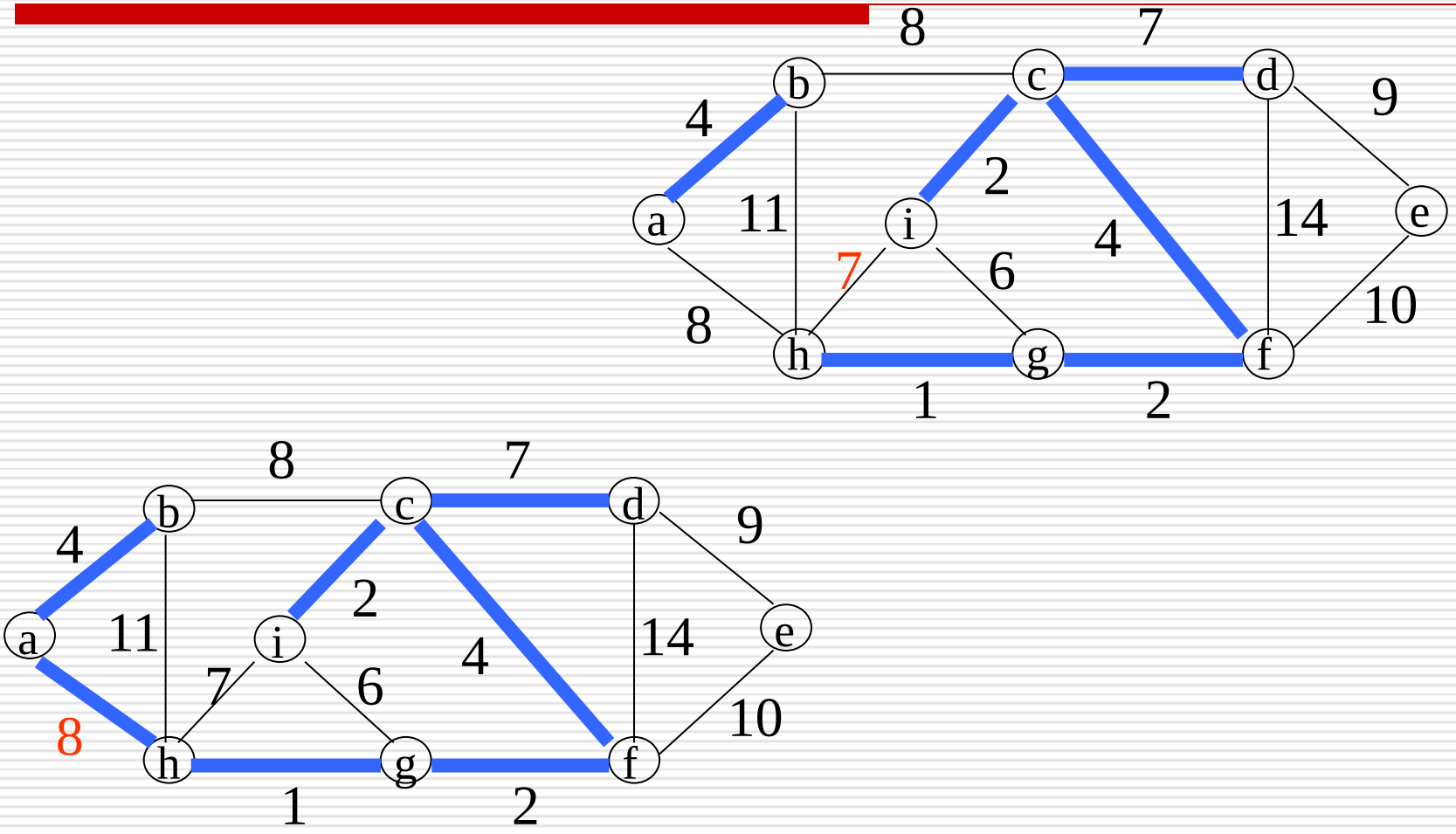

8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
10
8
h
g
f
1
2

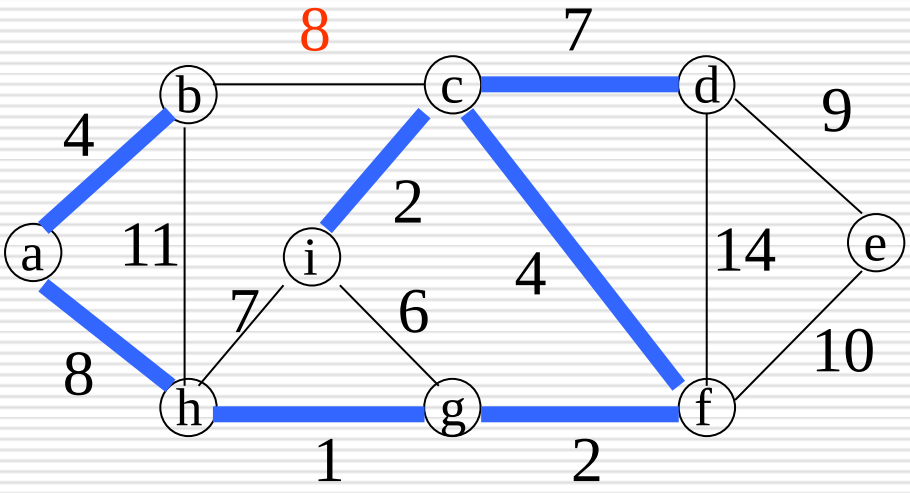

8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
8
h
g
f
10
1
2

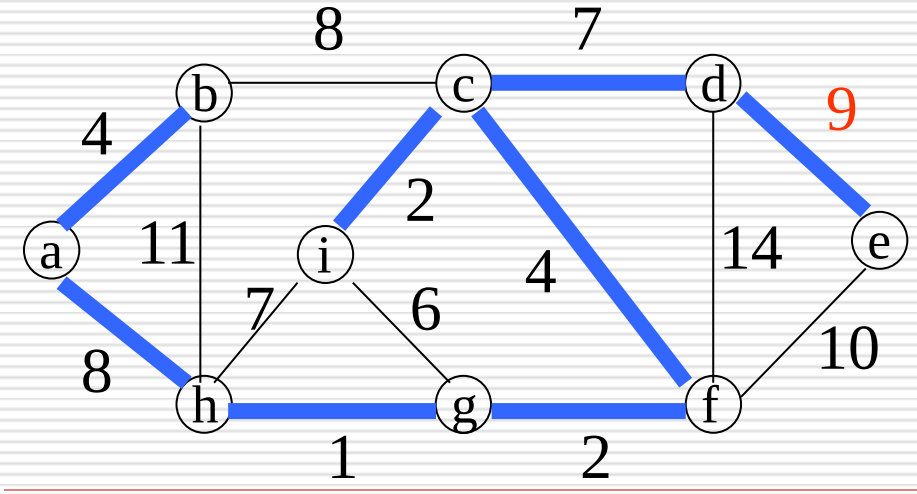

8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
8
h
g
f
10
1
2


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

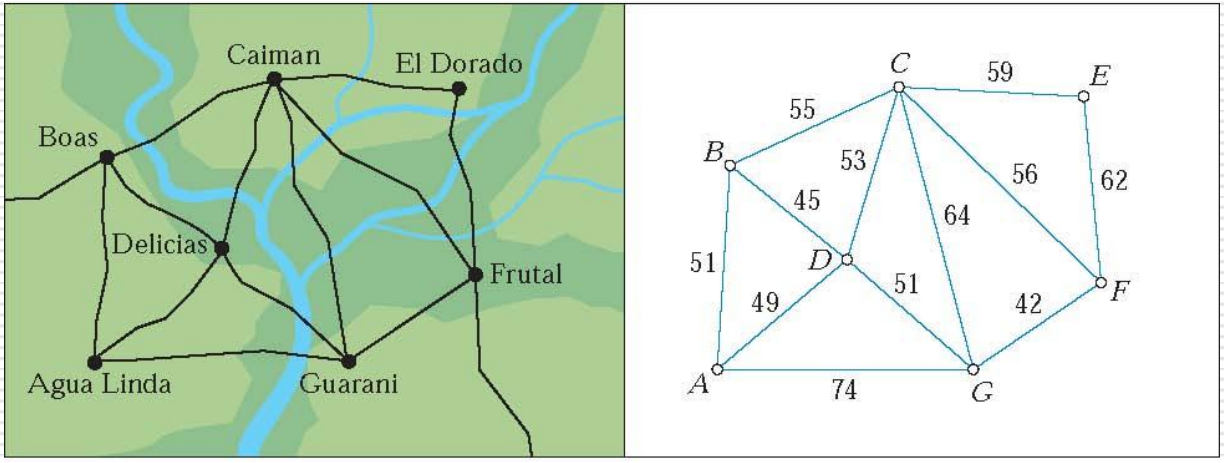

8
7
b
c
d
4
9
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


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


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


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

The Mathematics of Networks



What is the minimum spanning tree (MST) of the network shown in (b)?

Algorithma Prim

Pada algoritma prim, dimulai pada vertex yang mempunyai sisi (edge) dengan bobot terkecil.

Sisi yang dimasukkan ke dalam himpunan T adalah sisi graph G yang bersisian dengan sebuah simpul di T, sedemikian sehingga T adalah Tree (pohon). Sisi dari Graph G ditambahkan ke T jika ia tidak membentuk cycle.

(NOTE: dua atau lebih edge kemungkinan mempunyai bobot yang sama, sehingga terdapat pilihan vertice, dalam hal ini dapat diambil salah satunya.)

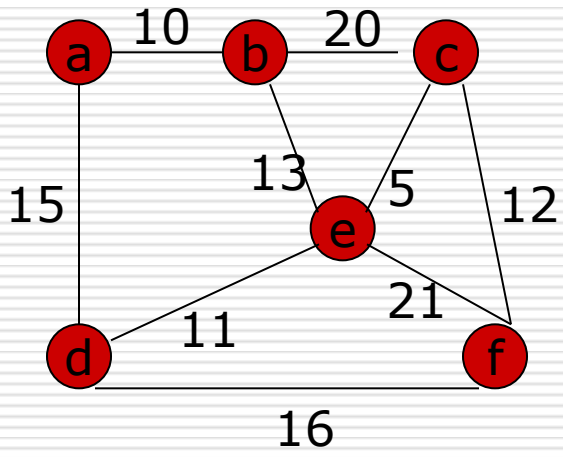
Algorithma Prim

1. Ambil sisi (edge) dari graph yg berbobot minimum, masukkan ke dalam T
2. Pilih sisi (edge) (i,j) yg berbobot minimum dan bersisisan dengan simpul di T, tetapi (i,j) tidak membentuk cycle di T. tambahkan (i,j) ke dalam T
3. Ulangi prosedur no 2 sebanyak (n-2) kali

Algorithma Prim

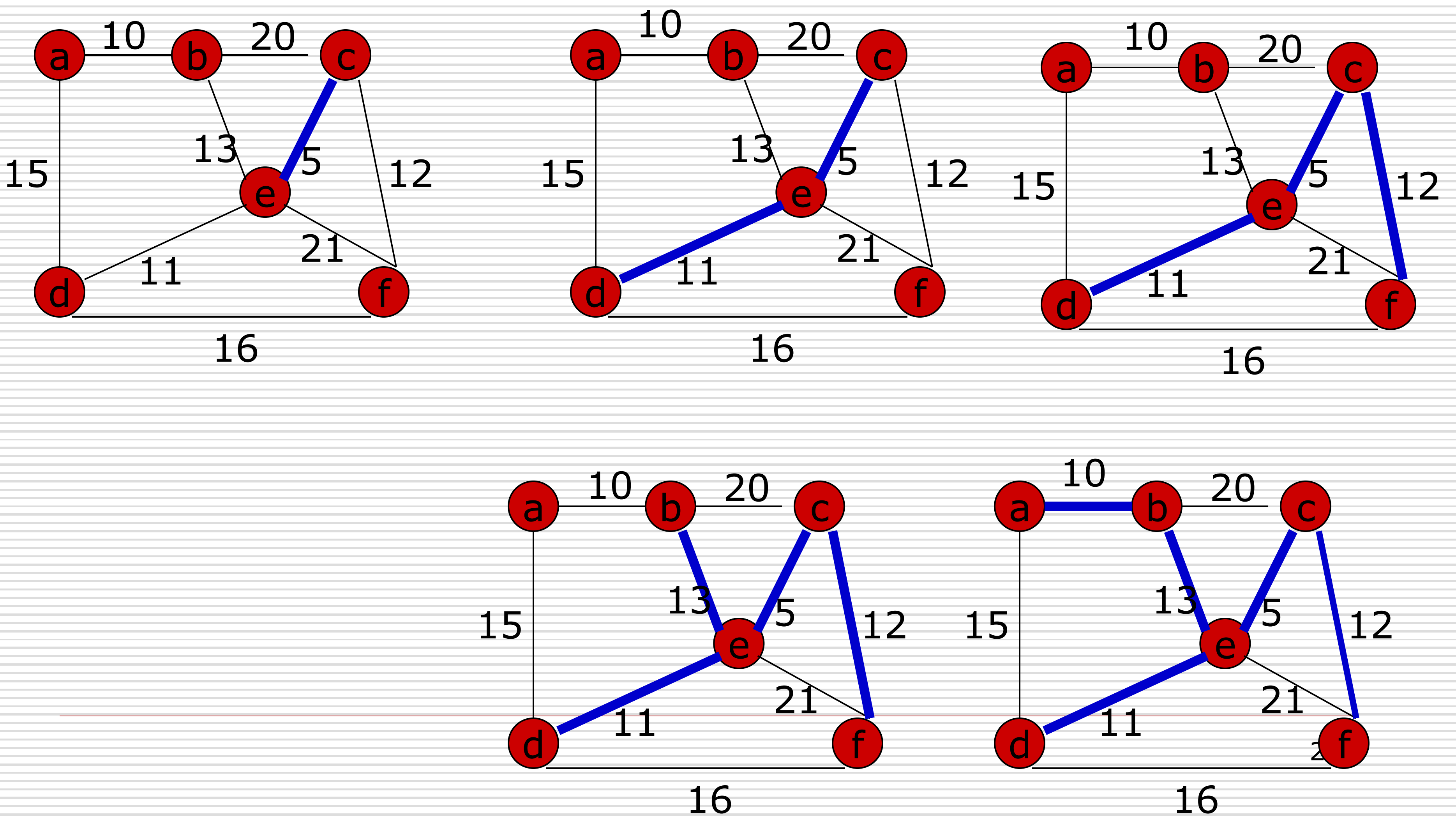
```
PROCEDURE Prim
(G: weighted connected undirected graph with n vertices)

  BEGIN
    T := a minimum-weight edge
    FOR   i := 1 to n-2 DO
      BEGIN
        e := a minimum-weight edge one
             of whose vertices is in T,
             and one  is not in T
        T := T with e added
      END
    RETURN T
  END
```

Algorithma Prim

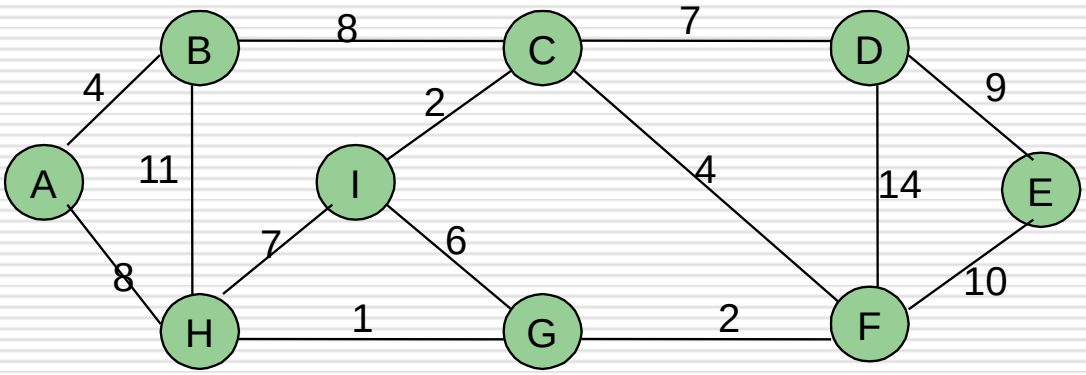


Langkah	Sisi	bobot
0		
1	e-c	5
2	d-e	11
3	c-f	12
4	b-e	13
5	a-b	10

Algorithm Prim



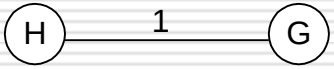
Algorithm Prim



LANGKAH	SISI	BOBOT
1	(H,G)	1
2	(G,F)	2
3	(F,C)	4
4	(C,I)	2
5	(C,D)	7
6	(C,B)	8
7	(B,A)	4
8	(D,E)	9

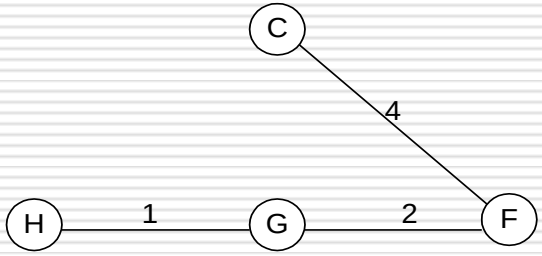
Algorithm Prim

Langkah 1

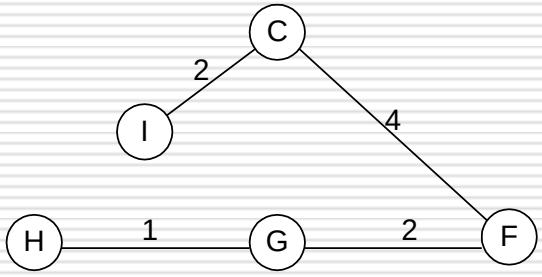


Langkah 2

Langkah 3

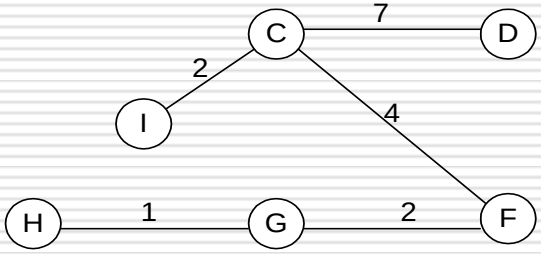


Langkah 3

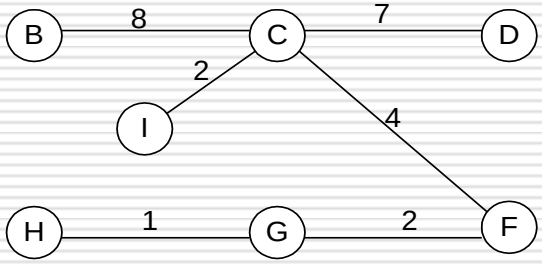


Algorithm Prim

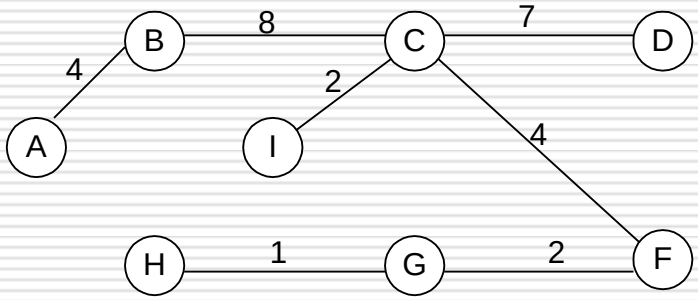
Langkah 4



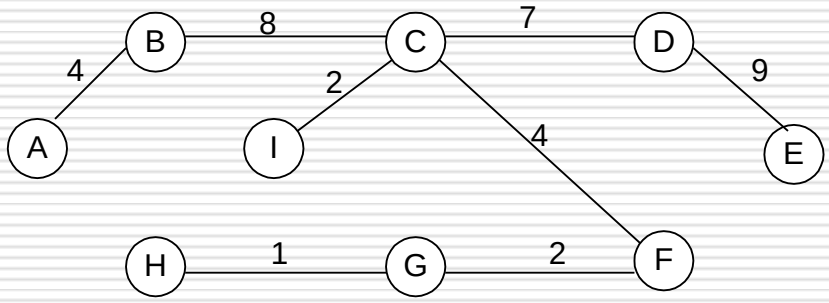
Langkah 5



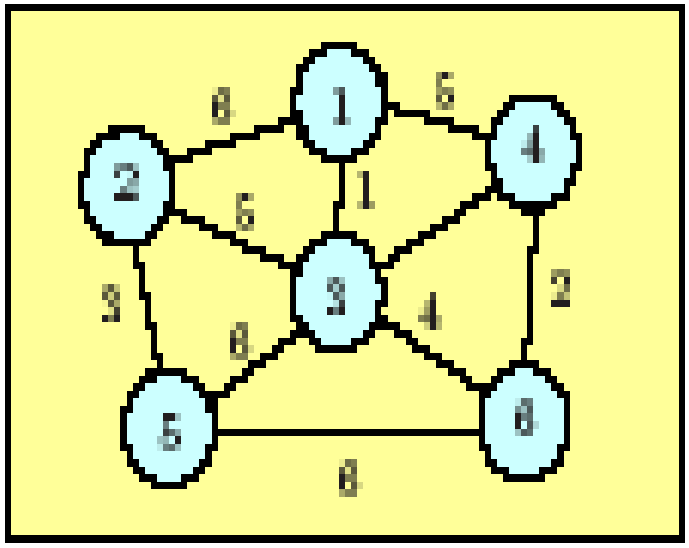
Langkah 6

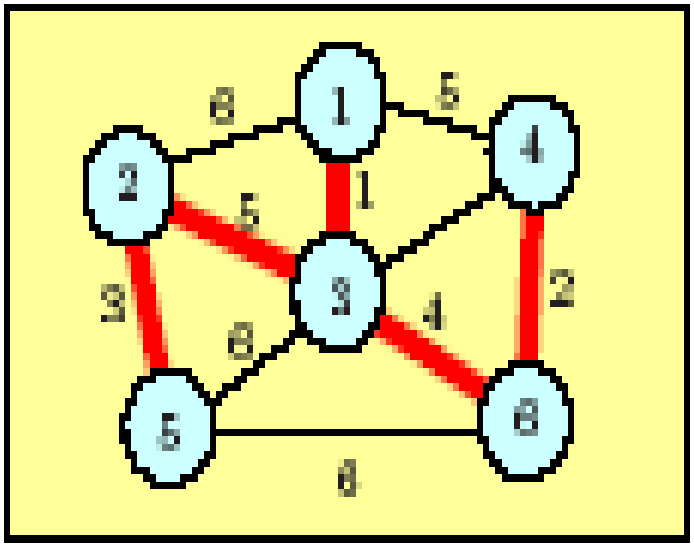


Langkah 7



Algorithm Prim





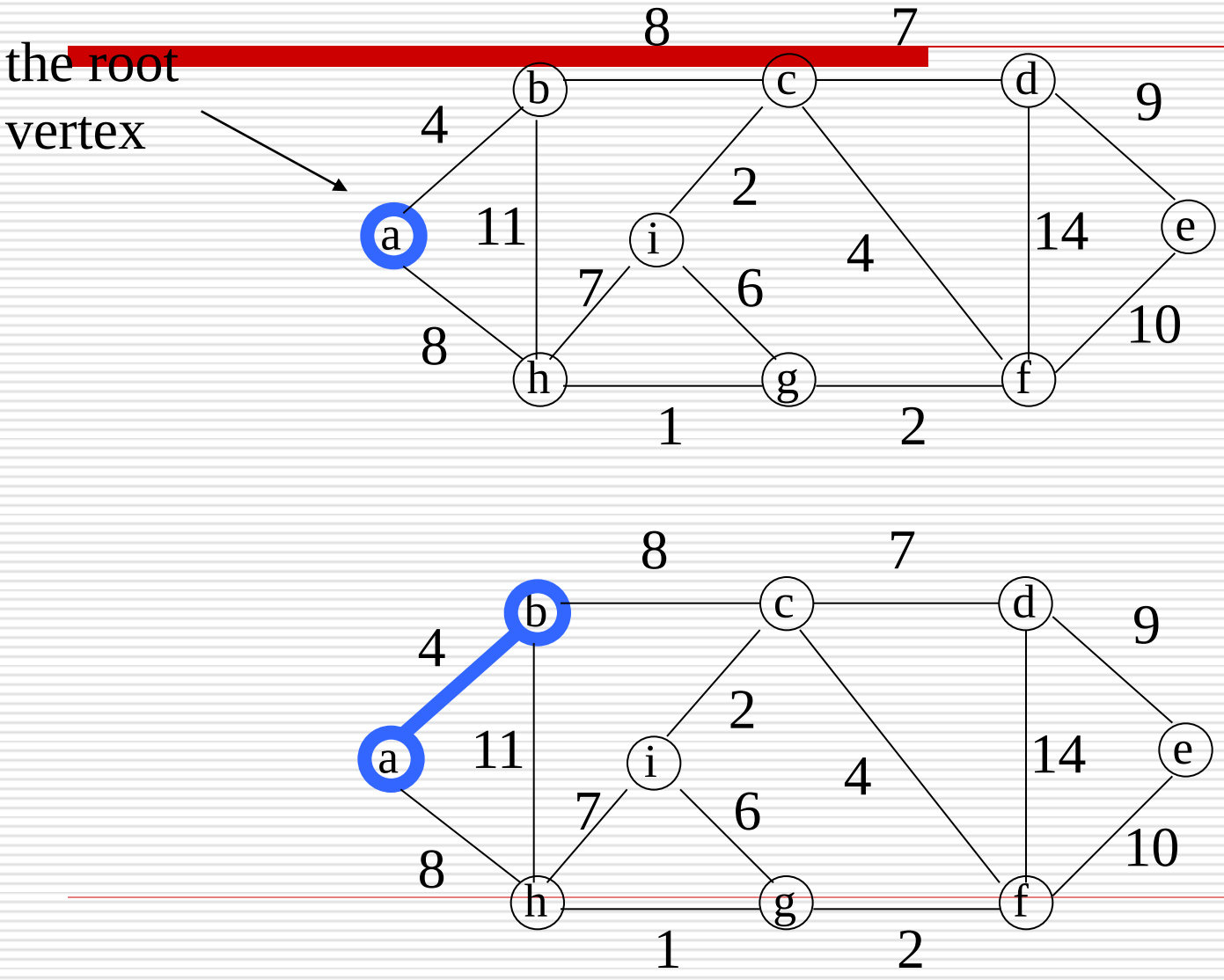
Prim's algorithm(basic part)

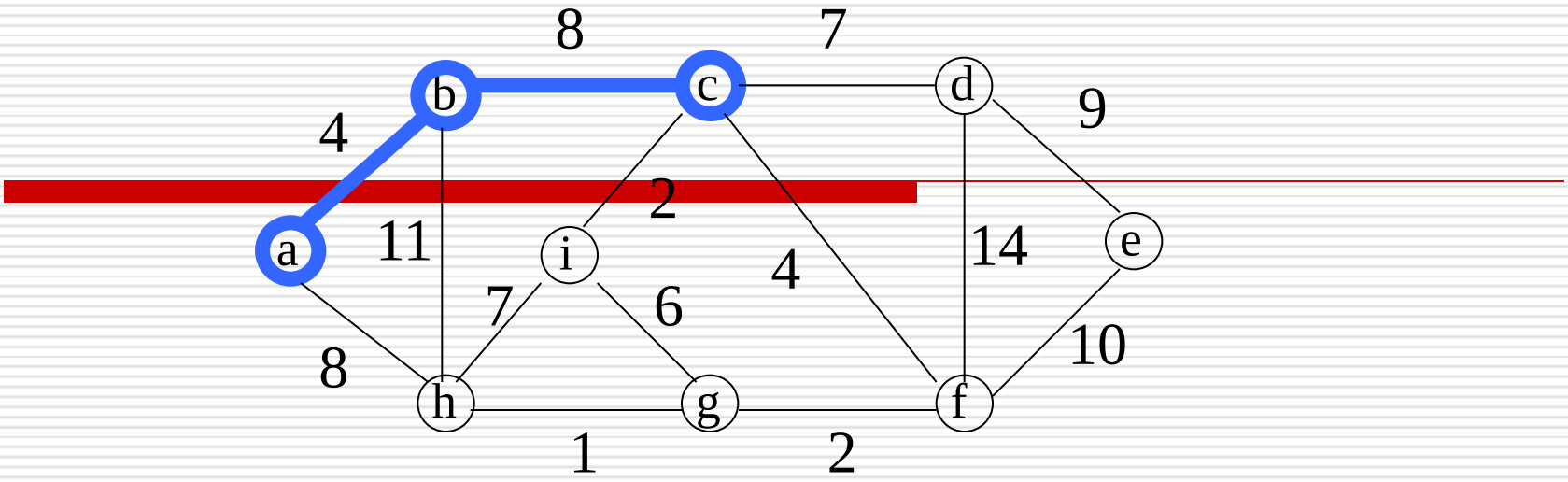
MST_PRIM(G,w,r)

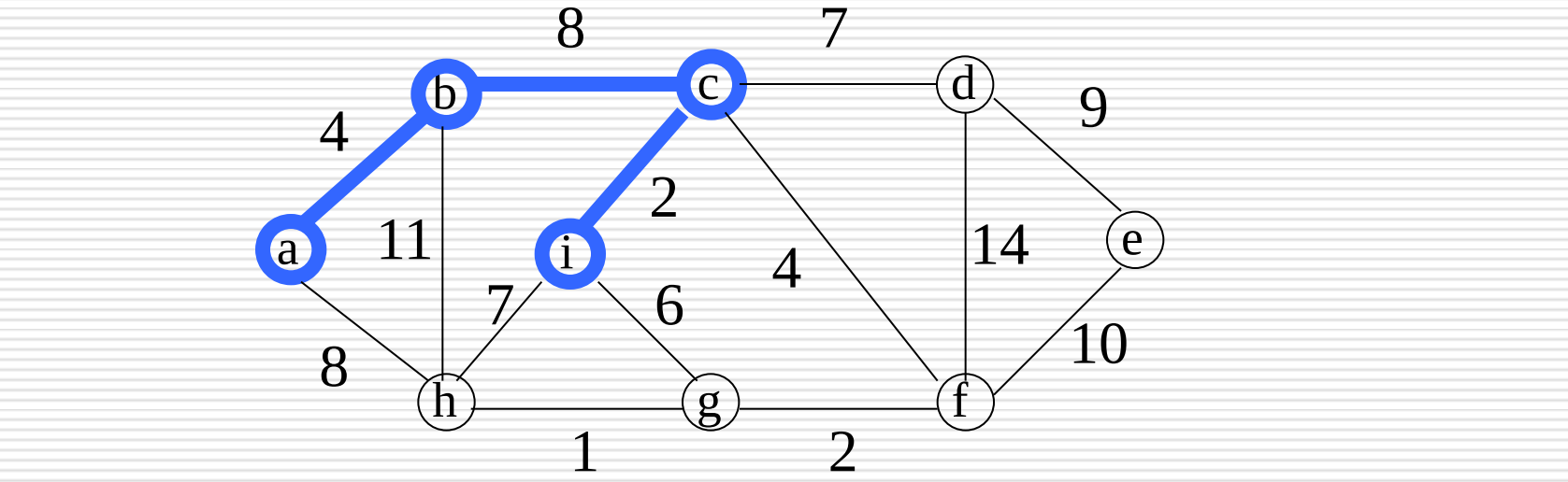
1. A={}
2. S:={r} (r is an arbitrary node in V)
3. Q=V-{r};
4. **while** Q is not empty **do {**
5. take an edge (u, v) such that **(1) u** $\in$**S and v** $\in$ **Q (v**$\notin$ **S)** and
   (u, v) is the shortest edge satisfying (1)
6. add (u, v) to A, add v to S and delete v from Q
   }

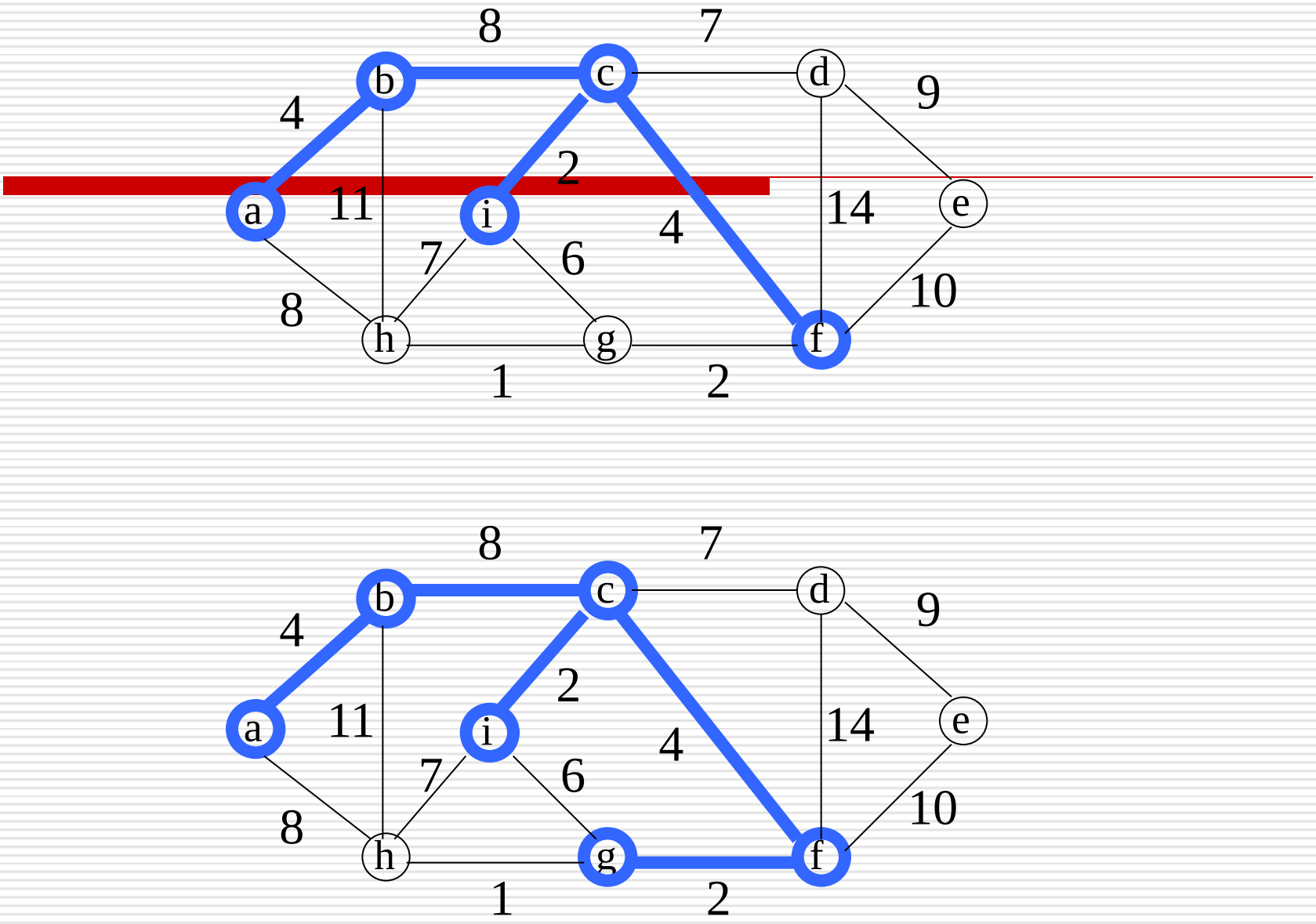
- Grow the minimum spanning tree from the root vertex r.
- Q is a priority queue, holding all vertices that are not in the tree now.
- key[v] is the minimum weight of any edge connecting v to a vertex in the tree.
- parent[v] names the parent of v in the tree.
- When the algorithm terminates, Q is empty; the minimum spanning tree A for G is thus $A=\{(v,parent[v]):v\in V-\{r\}\}$.
- Running time: $O(|E|+|V|\lg |V|)$. (Analysis is not required)(Fibonacci heap: decreas-key in O(1) time)

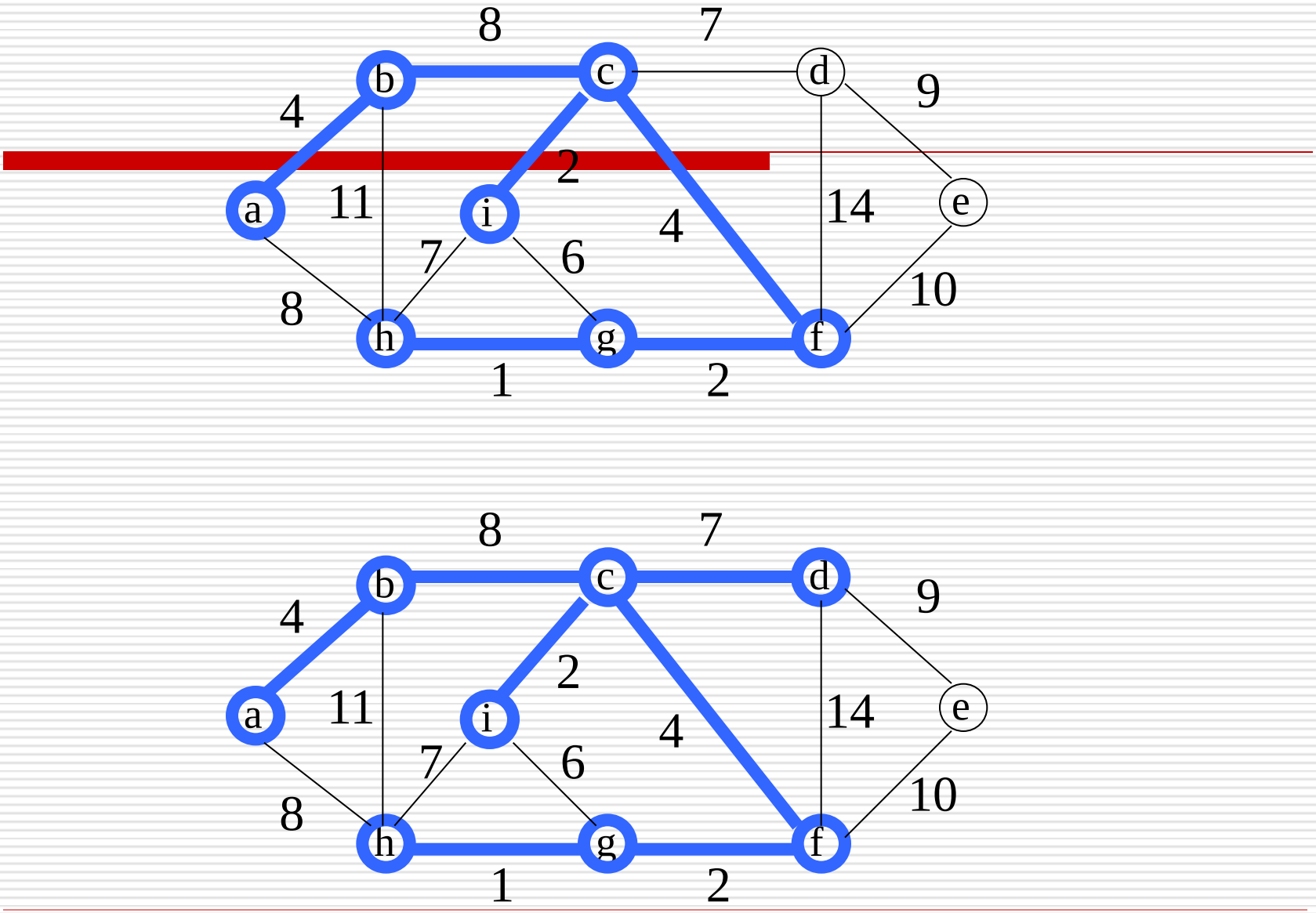
The execution of Prim's algorithm(moderate part)

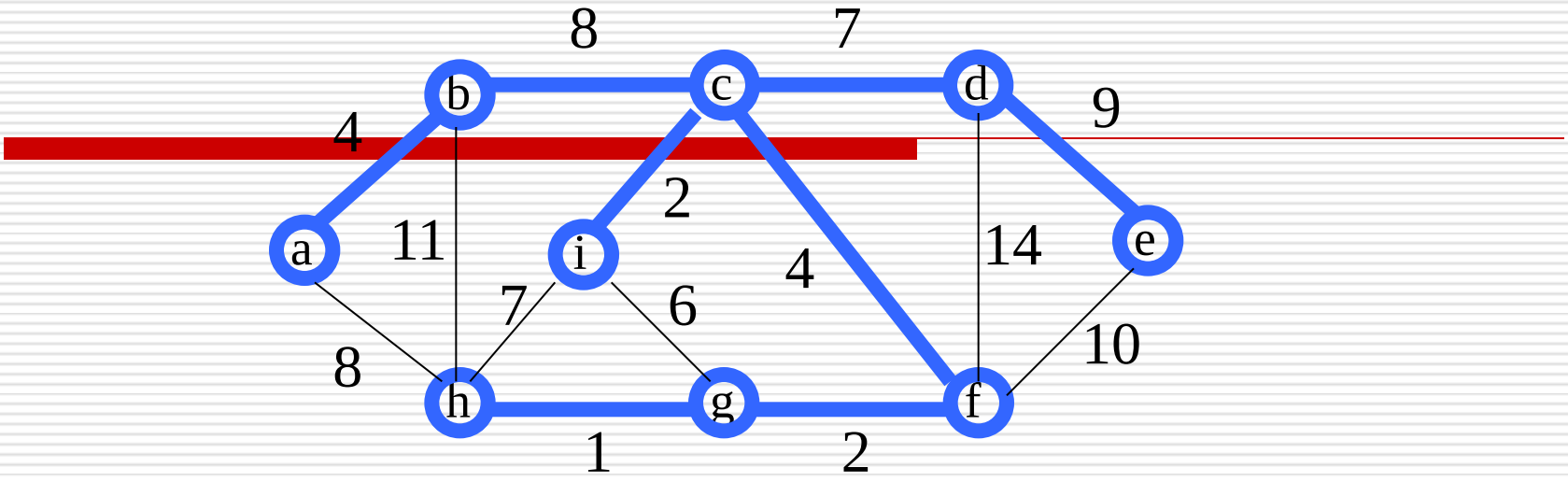



8
7
b
c
d
9
4
2
a
11
i
14
e
4
7
6
8
10
h
g
f
1
2


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


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


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

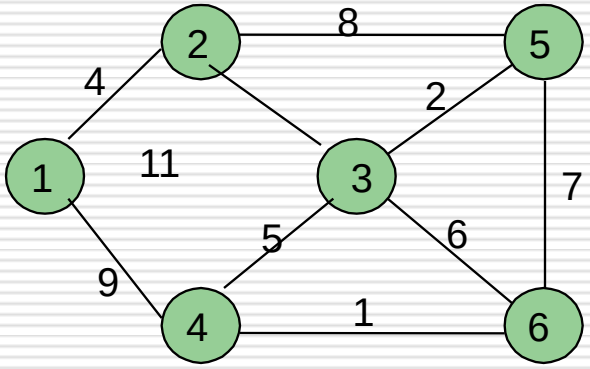


Bottleneck spanning tree: A spanning tree of G whose largest edge weight is minimum over all spanning trees of G. The value of the bottleneck spanning tree is the weight of the maximum-weight edge in T.

Theorem: A minimum spanning tree is also a bottleneck spanning tree. **(Challenge problem)**

soal

- Cari minimum spanning tree dengan menggunakan algoritma prim dan kruskal !



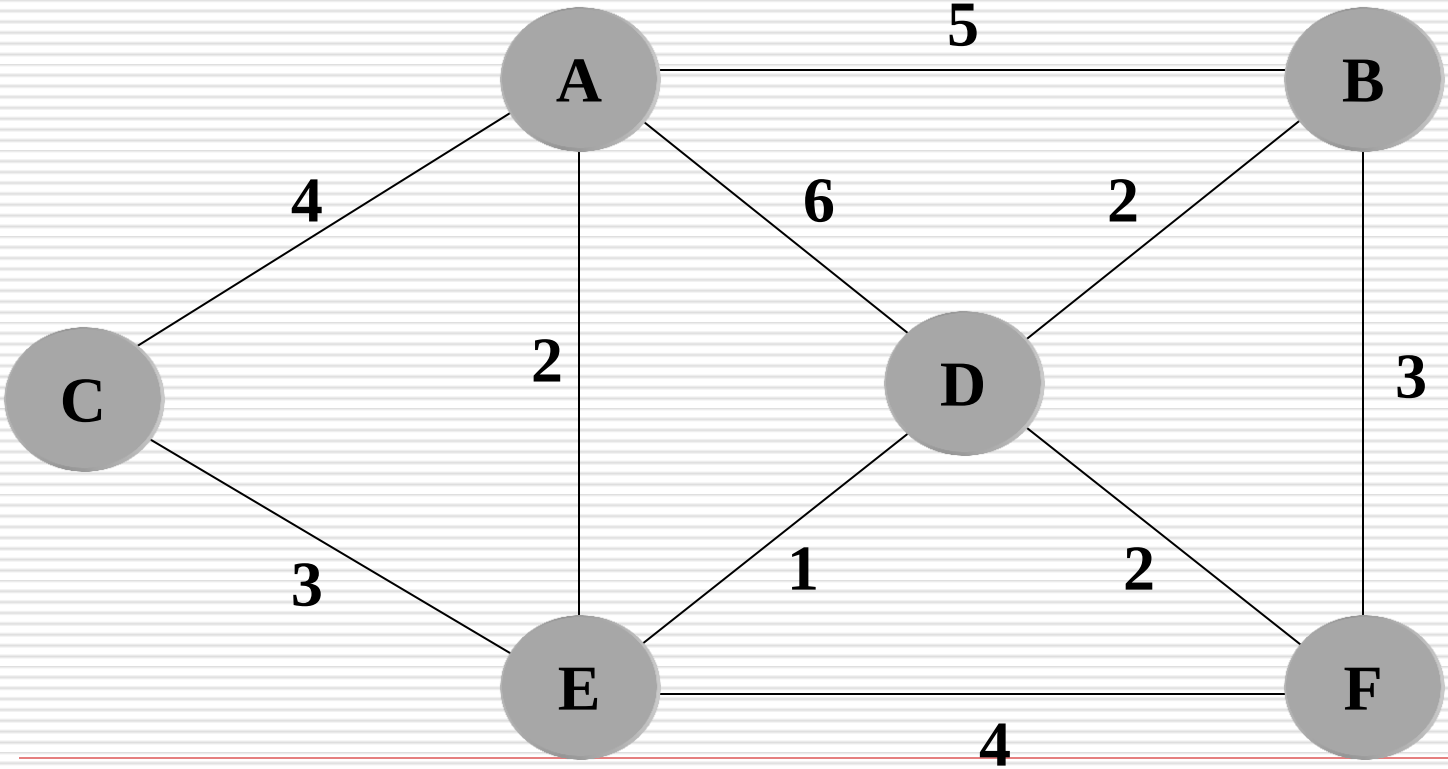
Barůvka's Algorithm

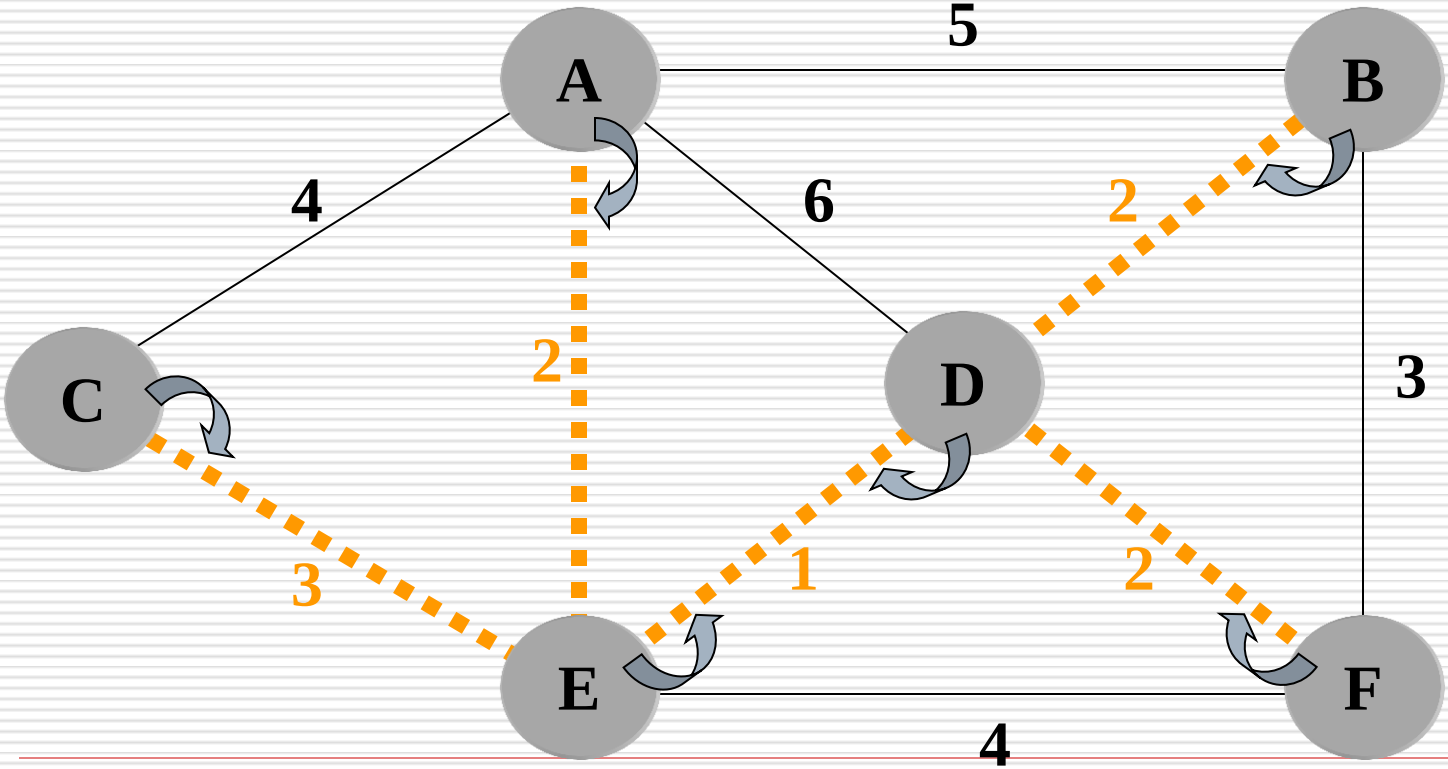
Barůvka's Algorithm

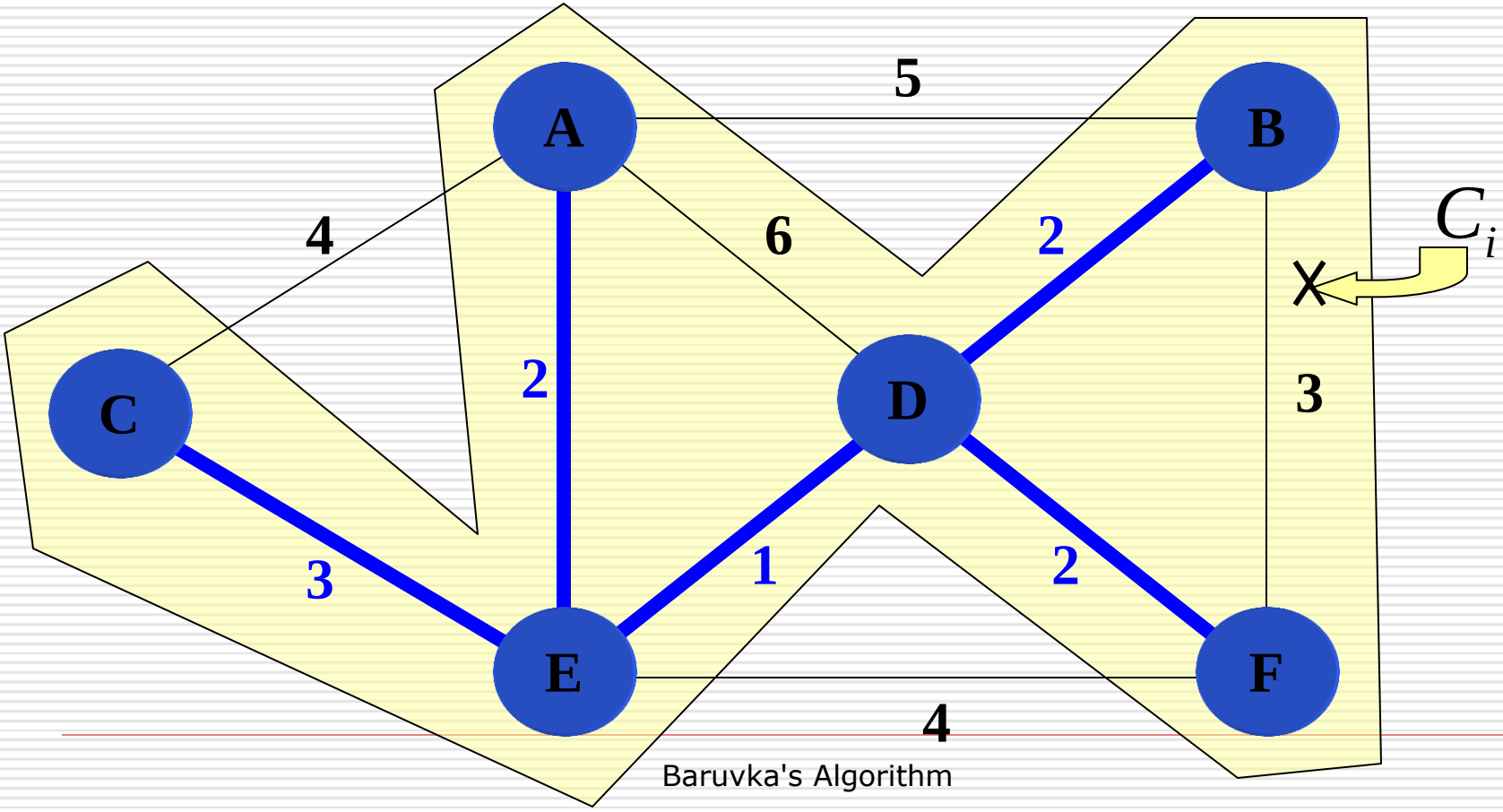
1. For all vertices search the edge with the smallest weight of this vertex and mark these edges

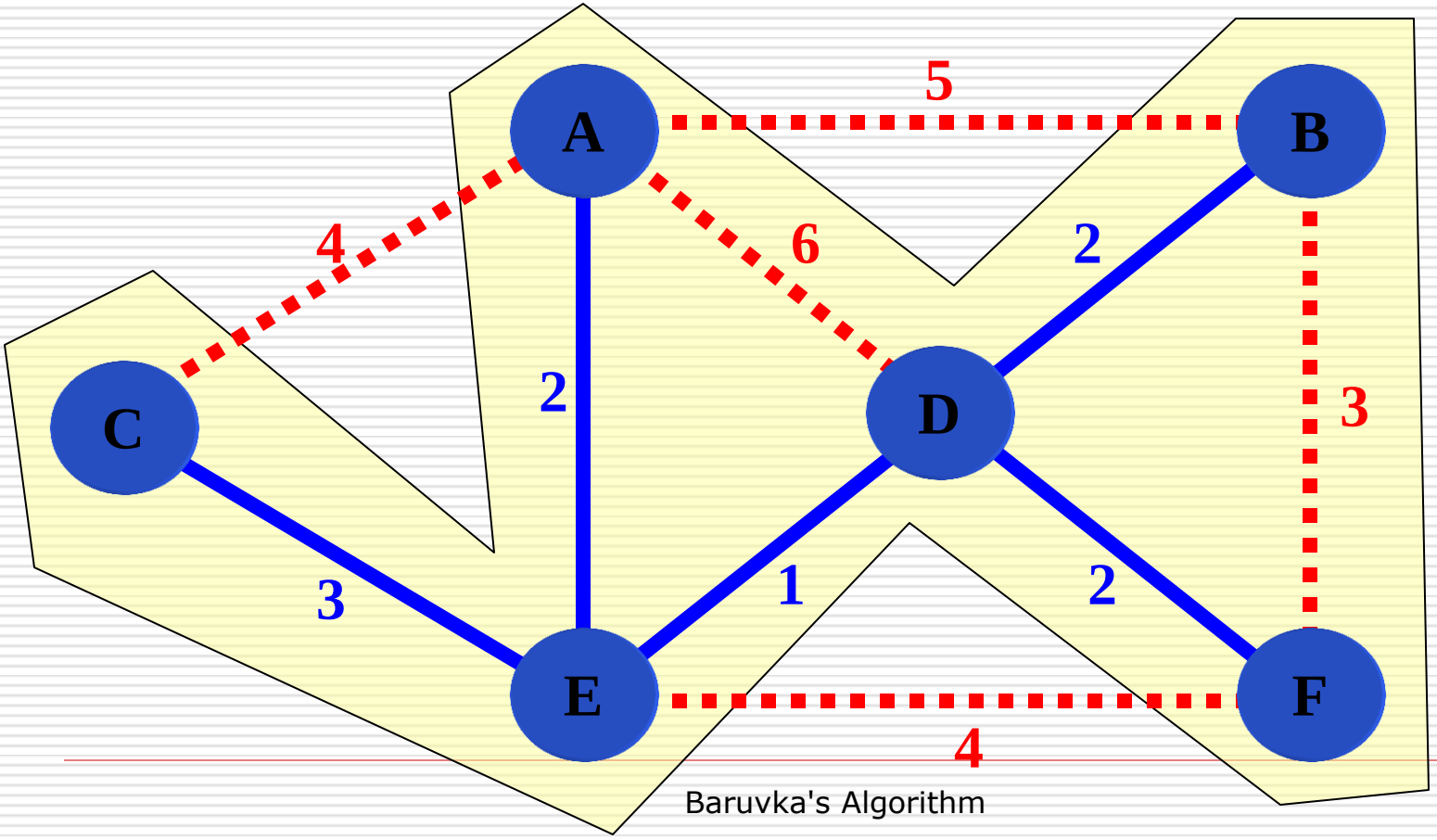
2. Search connected vertices (clusters) and replace them by a "new" vertex C_i (cluster)

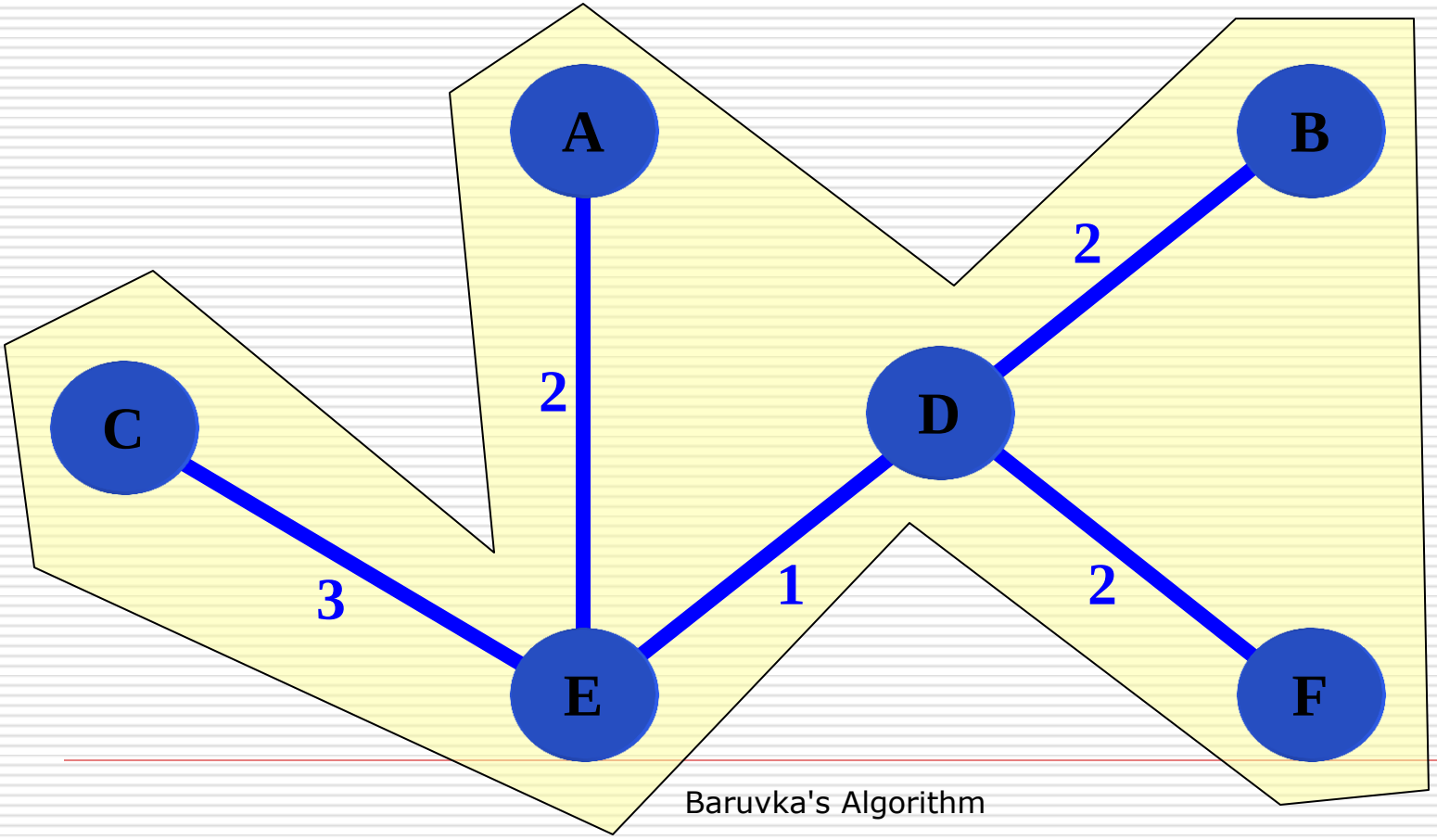
3. Remove the cycles and, if two vertices are connected by more than one edge, delete all edges except the "cheapest"


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4

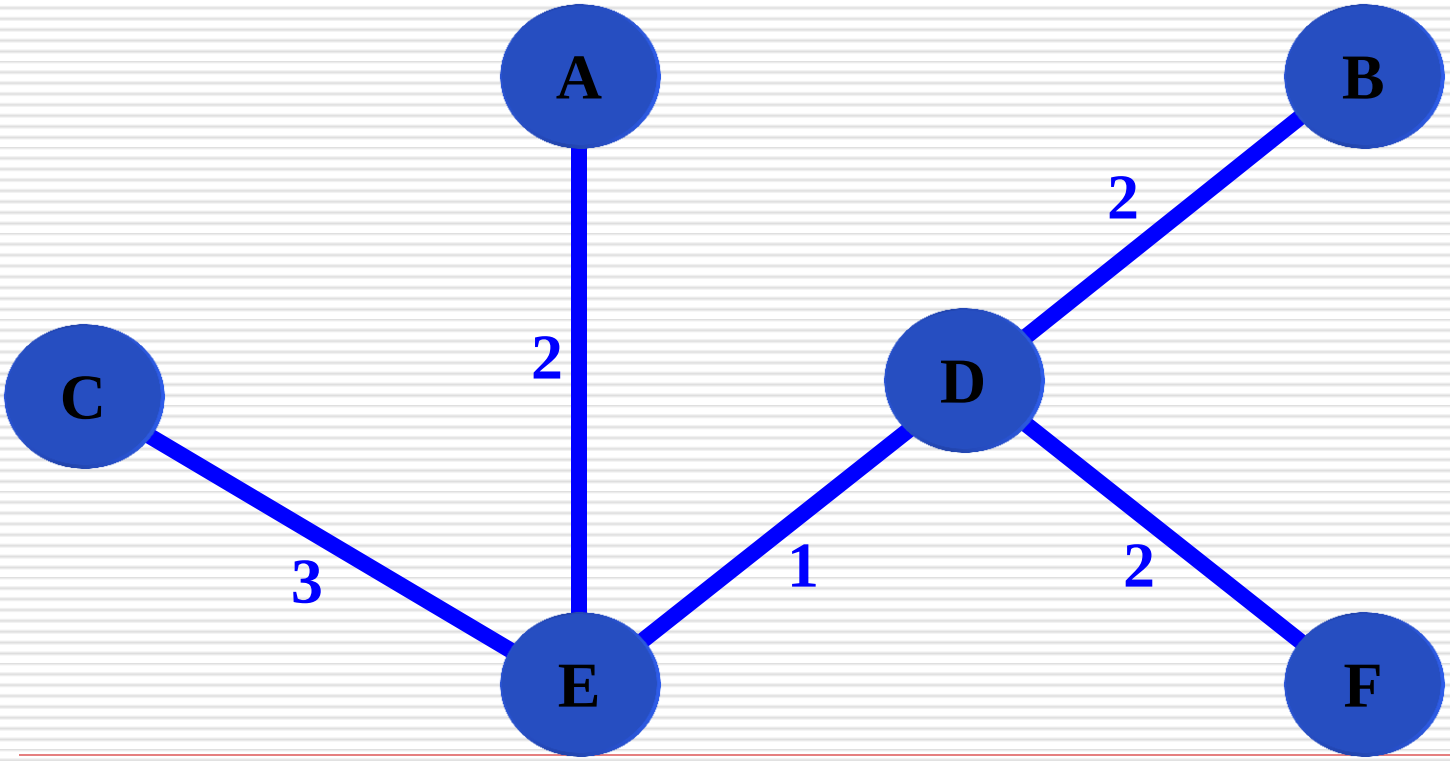

A
B
C
D
E
F
5
4
6
2
3
2
1
2
3
4


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4
C_i
X


A
B
C
D
E
F
5
4
6
2
2
3
3
1
2
4


A
B
C
D
E
F
2
2
3
1
2

minimum- spanning tree



soal

- Tentukan minimum spanning tree dengan menggunakan algoritma kruskal, baruvka dan prim