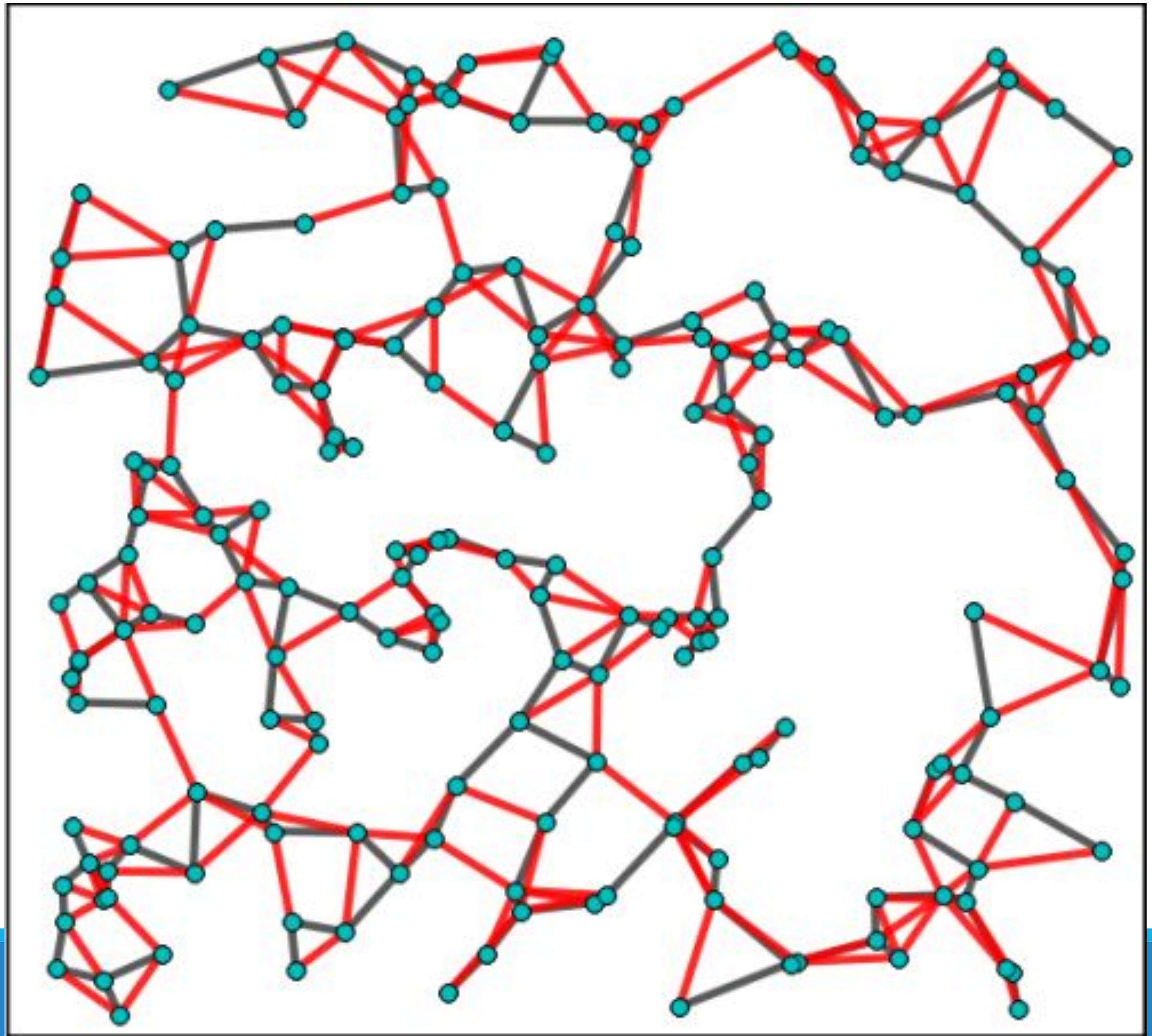
Grafos 2

Temario

- Árbol de Expansión Mínimo
- Algoritmo de Kruskal
- Algoritmo de Prim
- Algoritmo de Dijkstra

Árbol de Expansión Mínimo (MST)



Árbol de Expansión Mínimo (MST)

Entrada:

- Un grafo no dirigido (conectado) G = (V,E) con una función de pesos
  - W: E→ R

Salida:

- Un árbol de expansión T (árbol que conecta todos los nodos con un mínimo peso)

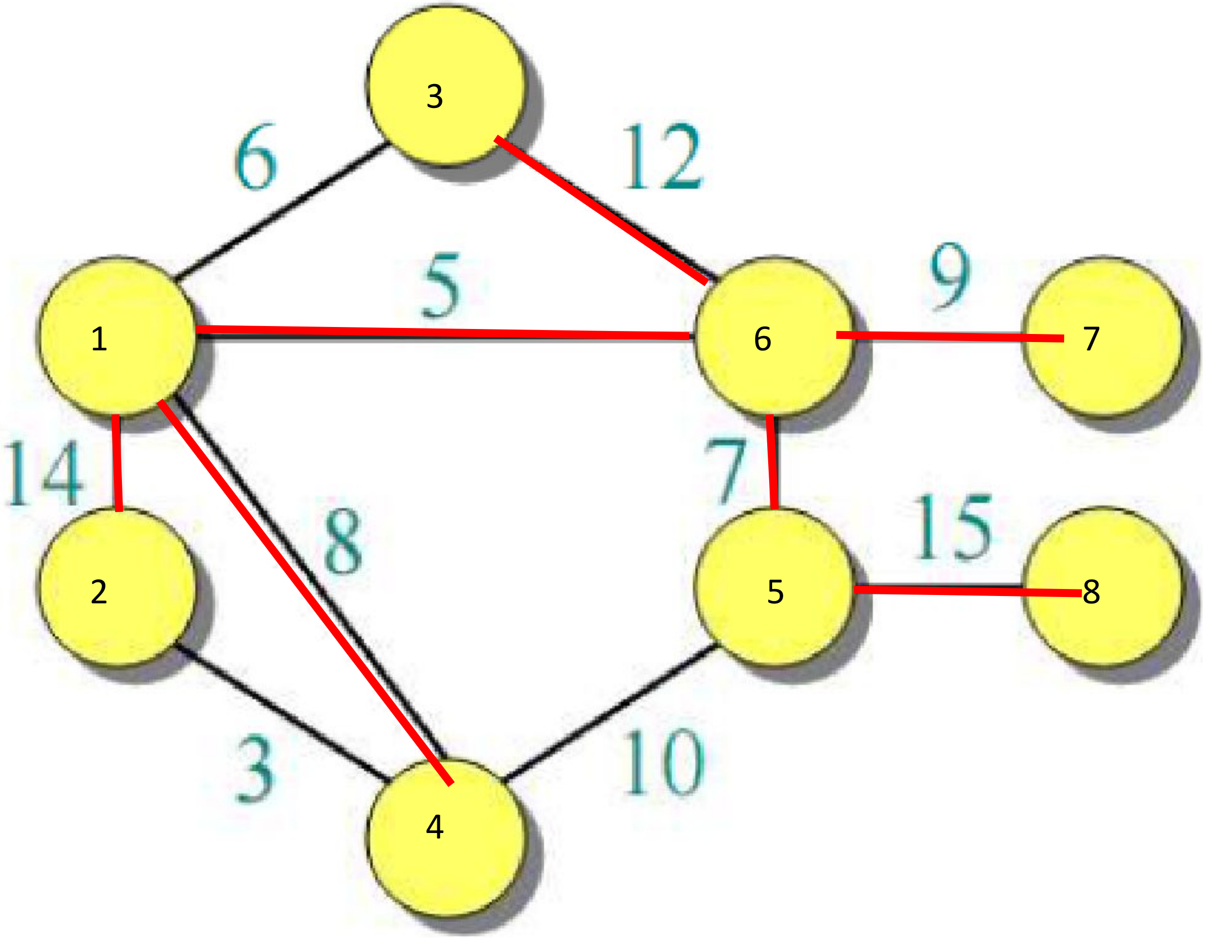
$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

Árbol de Expansión Mínimo (MST)

Ejemplo:

- **Entrada**

MST = 70

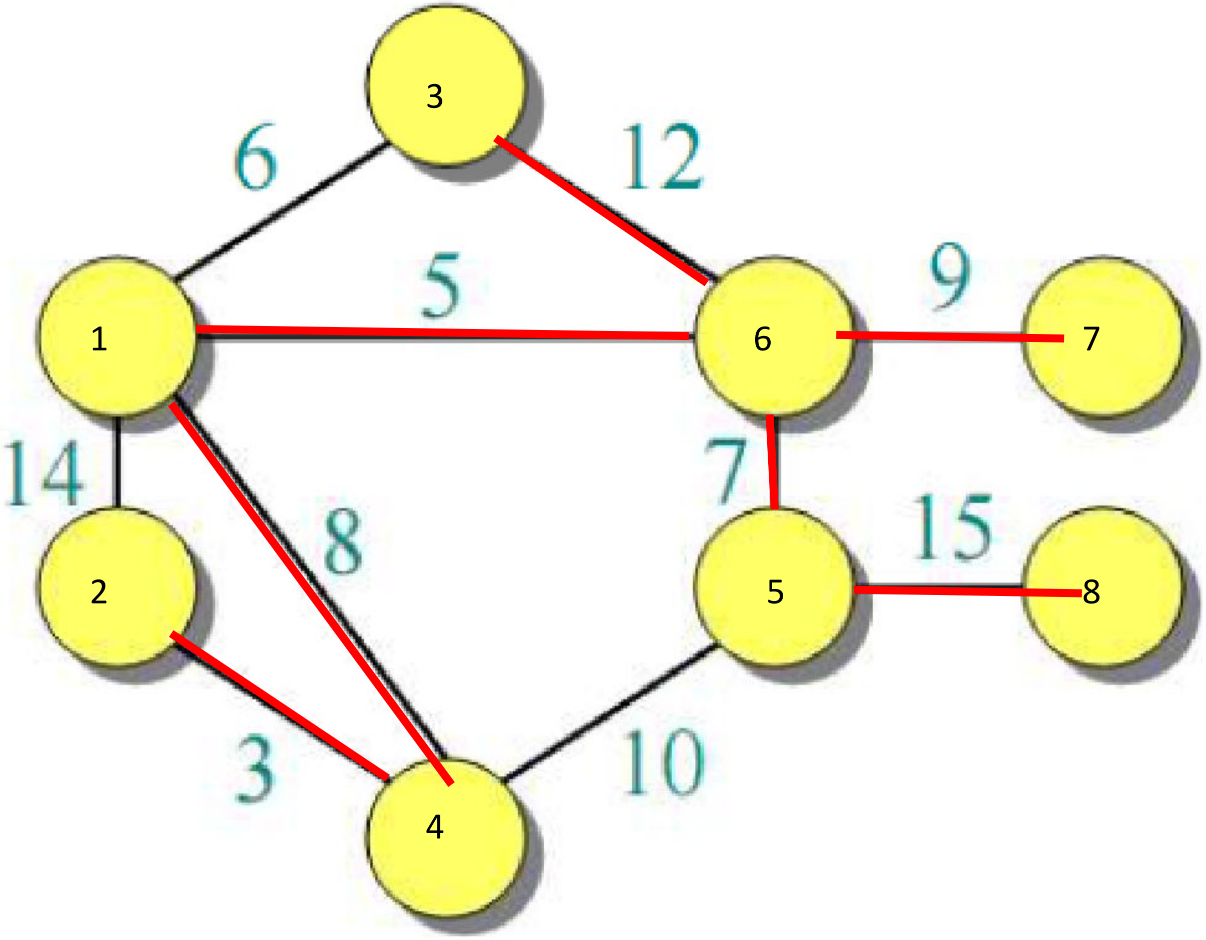


Árbol de Expansión Mínimo (MST)

Ejemplo:

- **Entrada**

MST = 59

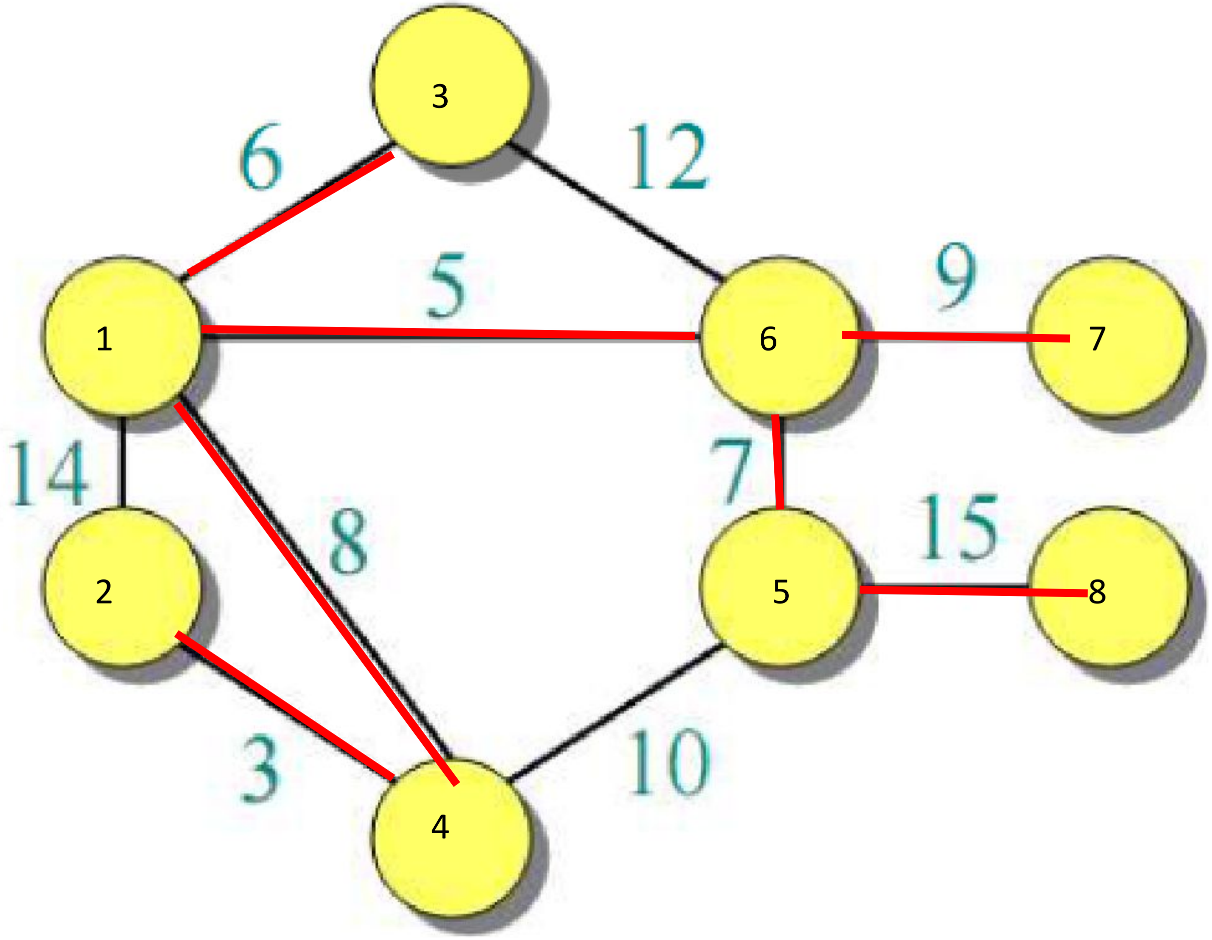


Árbol de Expansión Mínimo (MST)

Ejemplo:

- **Entrada**

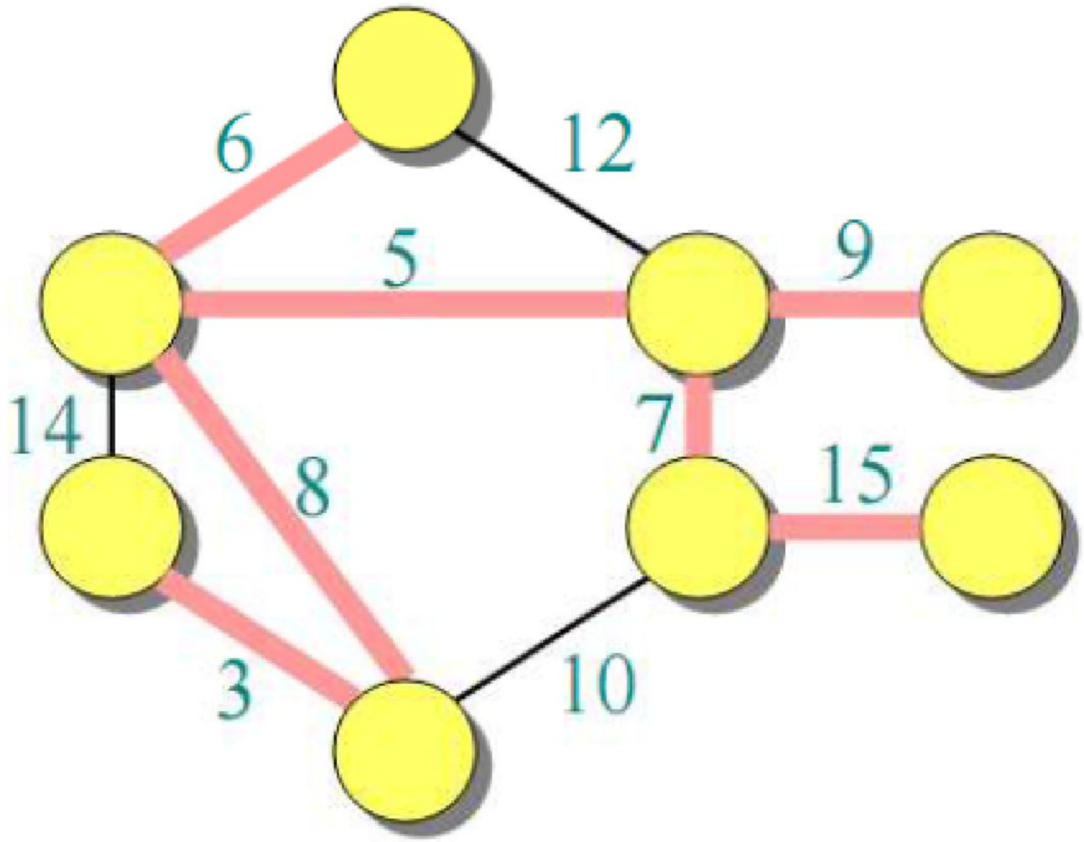
MST = 53

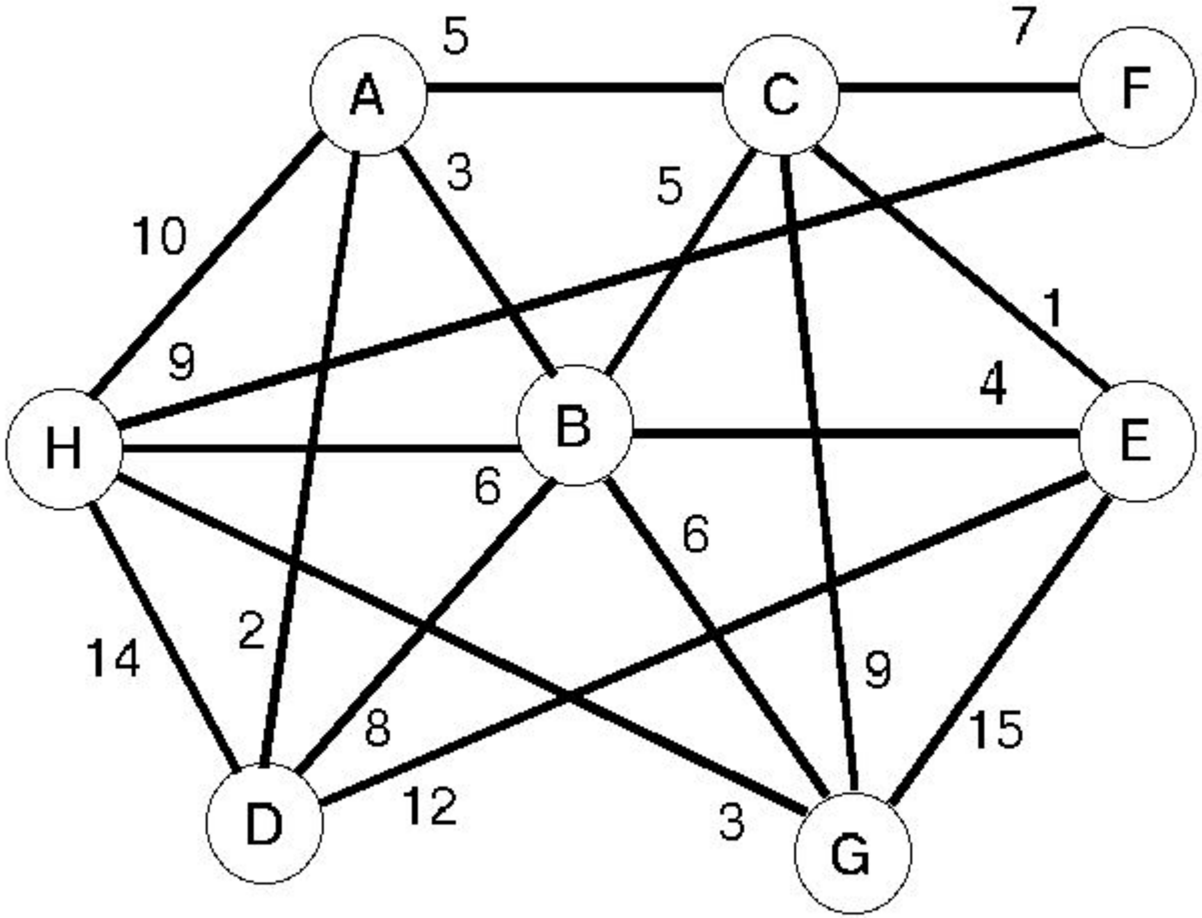


Árbol de Expansión Mínimo (MST)

Ejemplo:

- **Salida**



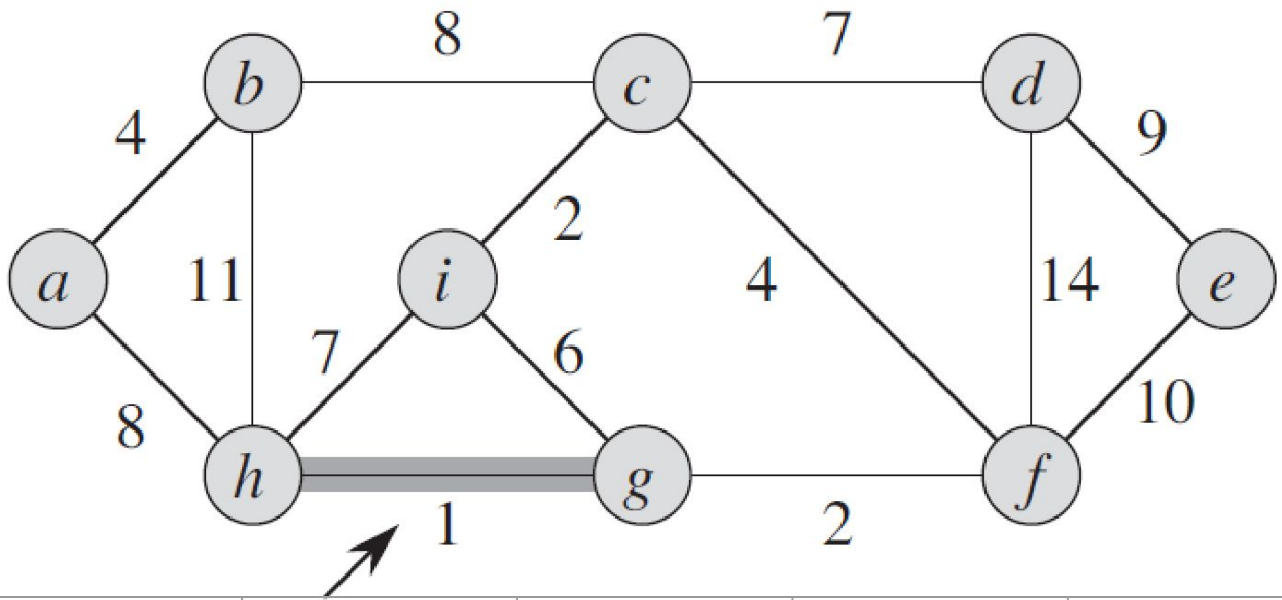

A
C
F
H
B
E
D
G
5
7
3
5
10
1
9
4
6
6
14
2
9
8
15
12
3

Algoritmo de Kruskal

```
MST-KRUSKAL(G, w)
O(1)        A = ∅
O(V)        for each vertex v ∈ G.V
                MAKE-SET(v)
O(E logE)   sort the edges of G.E into nondecreasing order by weight w
O(V logV)   for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
                if FIND-SET(u) ≠ FIND-SET(v)
                    A = A ∪ {(u, v)}
                    UNION(u, v)
            return A
```

Algoritmo de Kruskal

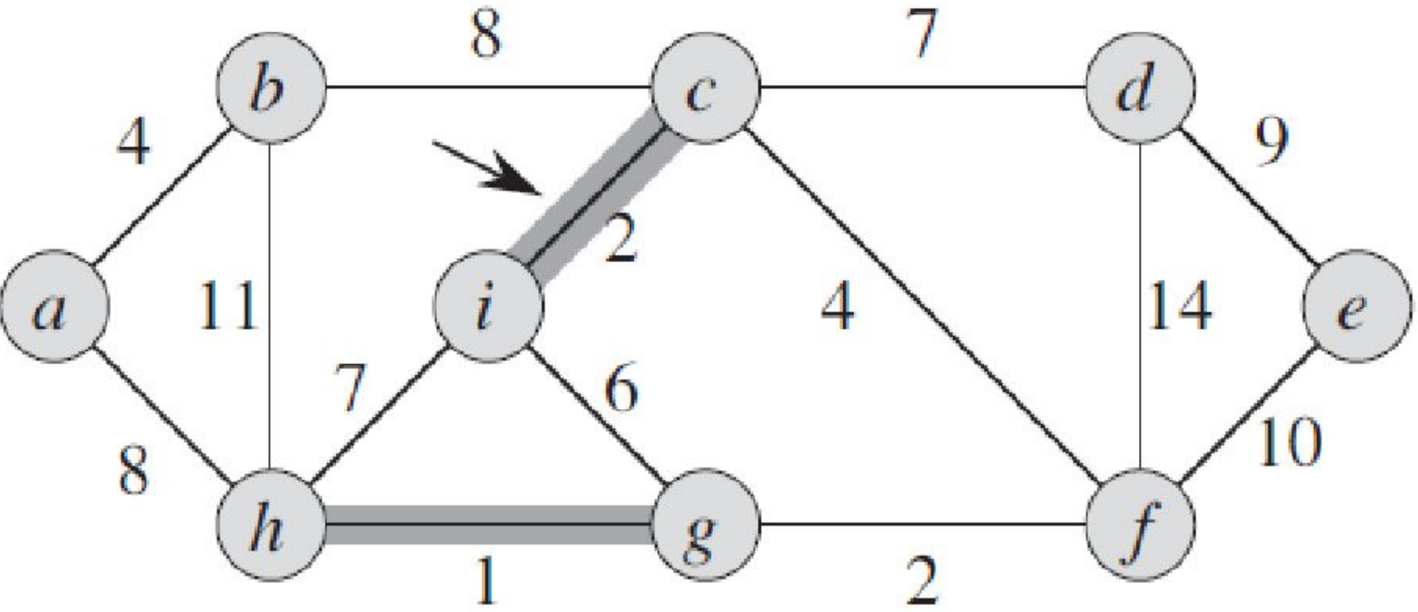
Ejemplo



hg				
1				

hg	1
gf	2
ic	2
cf	4
ab	4
ig	6
hi	7
cd	7
bc	8
ah	8
de	9
fe	10
bh	11
df	14

Algoritmo de Kruskal



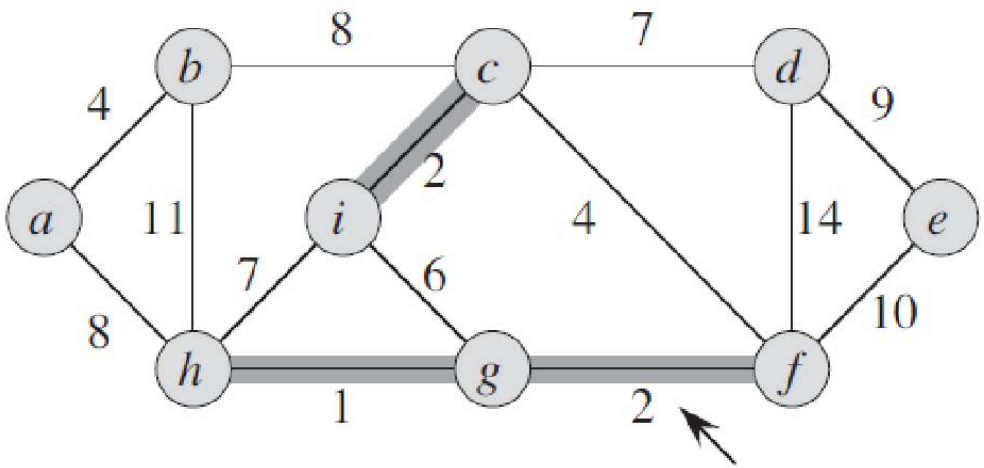
hg	1
ic	2
gf	2
cf	4
ab	4
ig	6
hi	7
cd	7
bc	8
ah	8
de	9
fe	10
bh	11
df	14

hg	ic			
1	2			

Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
hi	7
cd	7
bc	8
ah	8
de	9
fe	10
bh	11
df	14

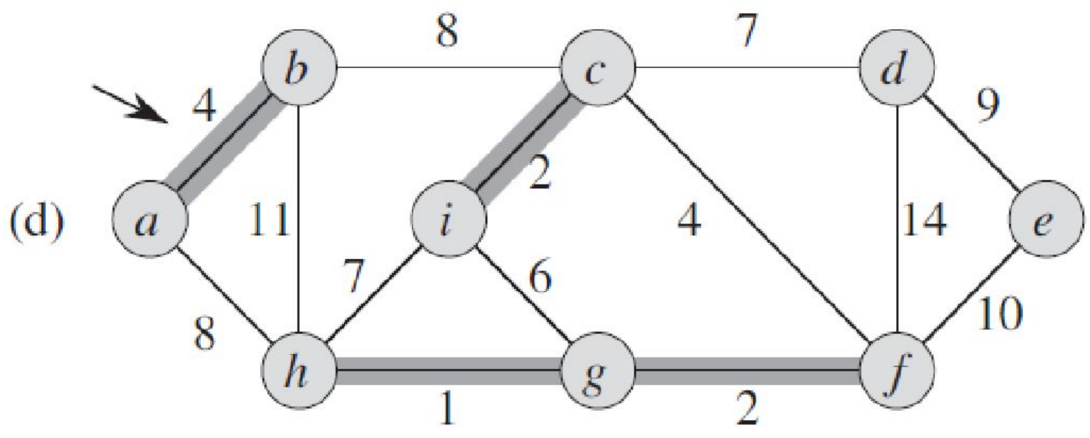
(c)



hg	ic	gf		
1	2	2		

Algoritmo de Kruskal

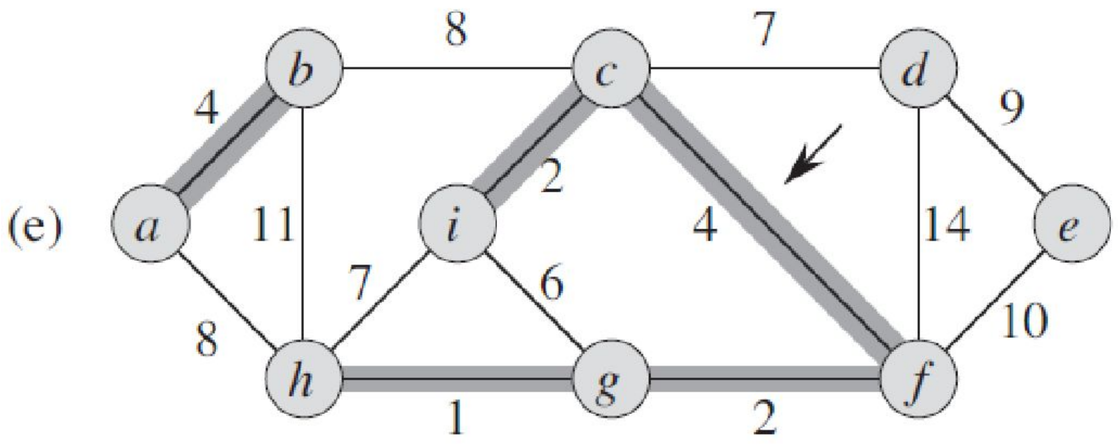
hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
hi	7
cd	7
bc	8
ah	8
de	9
fe	10
bh	11
df	14



hg	ic	gf	ab	
1	2	2	4	

Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
hi	7
cd	7
bc	8
ah	8
de	9
fe	10
bh	11
df	14

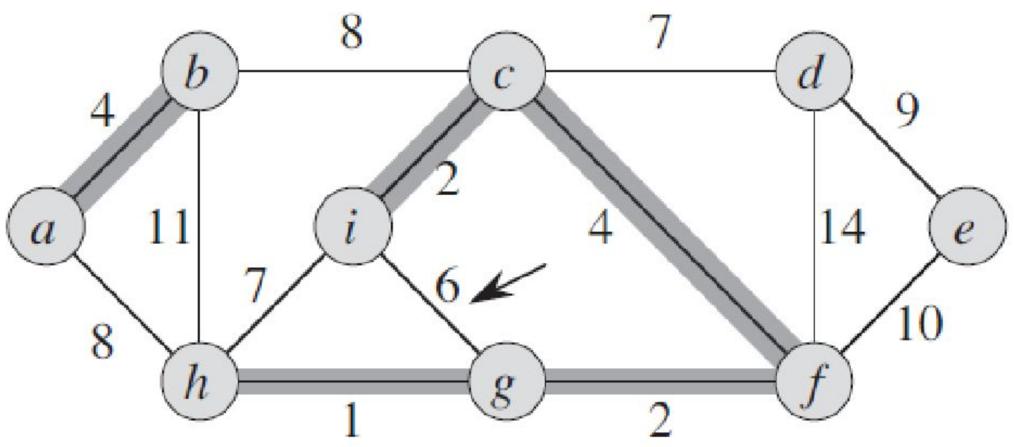


hg	ic	gf	ab	cf
1	2	2	4	4

Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14

(f)

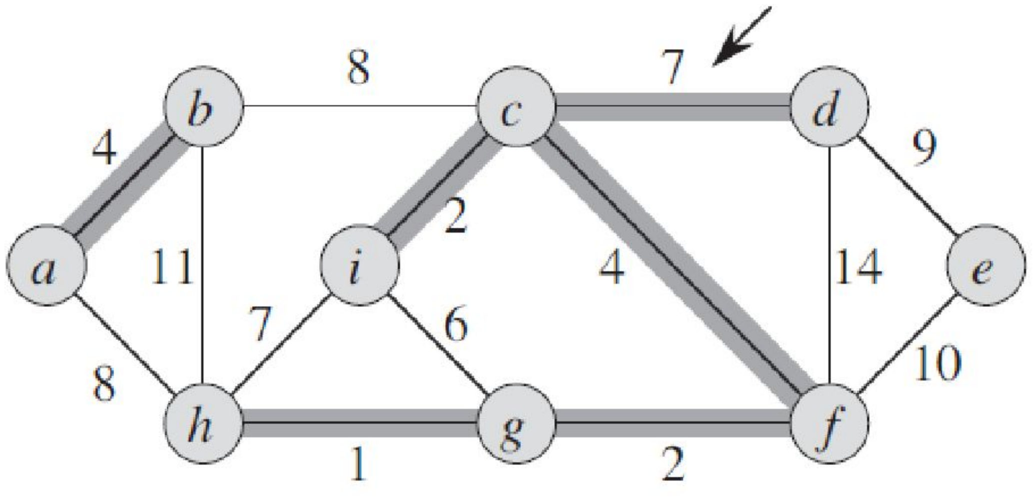


hg	ic	gf	ab	cf
1	2	2	4	4

Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14

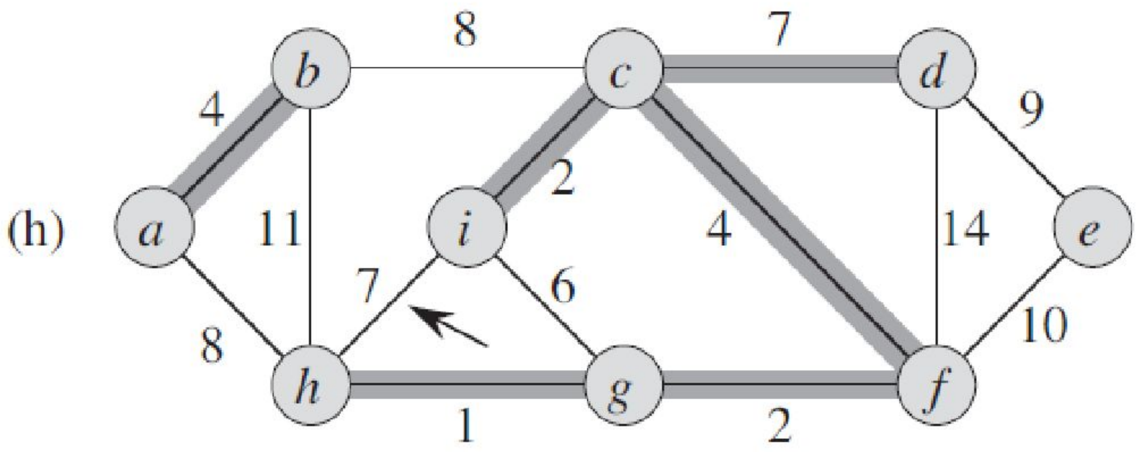
(g)



hg	ic	gf	ab	cf	cd
1	2	2	4	4	7

Algoritmo de Kruskal

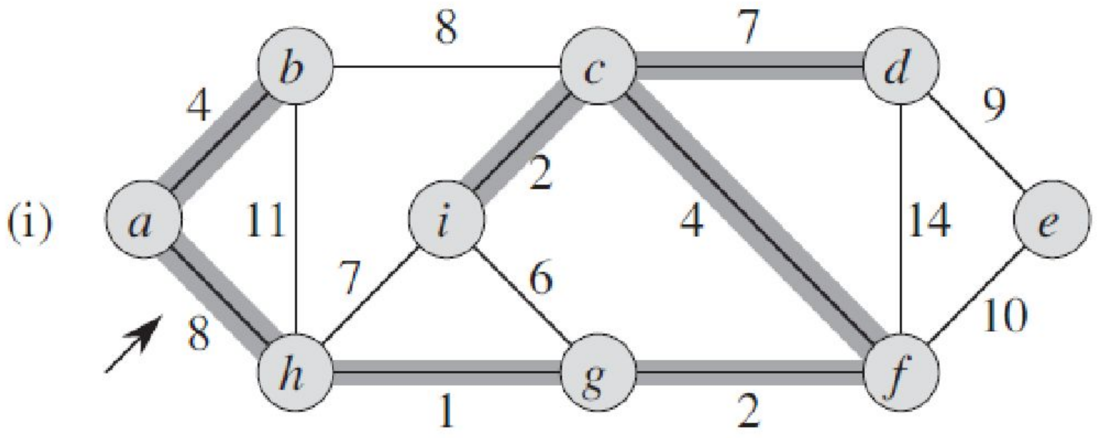
hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14



hg	ic	gf	ab	cf	cd
1	2	2	4	4	7

Algoritmo de Kruskal

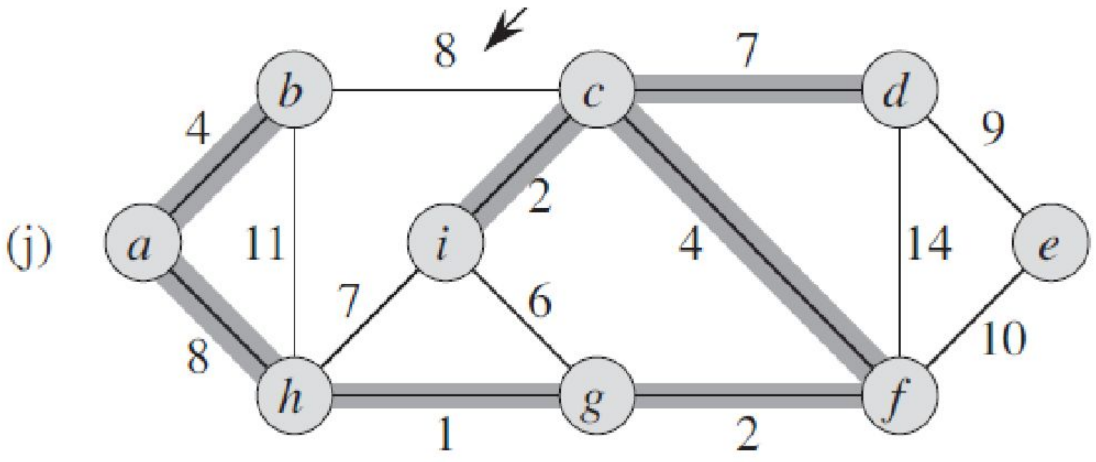
hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14



hg	ic	gf	ab	cf	cd	ah
1	2	2	4	4	7	8

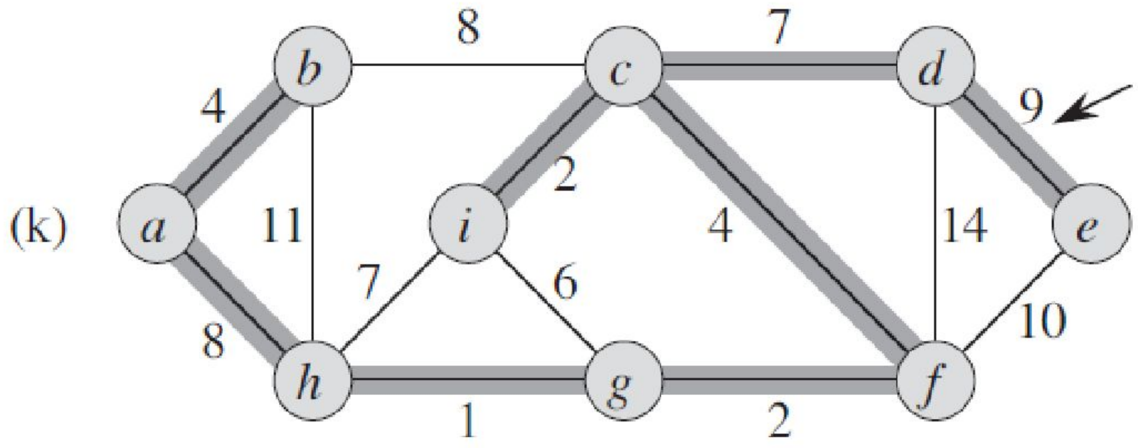
Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14



hg	ic	gf	ab	cf	cd	ah
1	2	2	4	4	7	8

Algoritmo de Kruskal



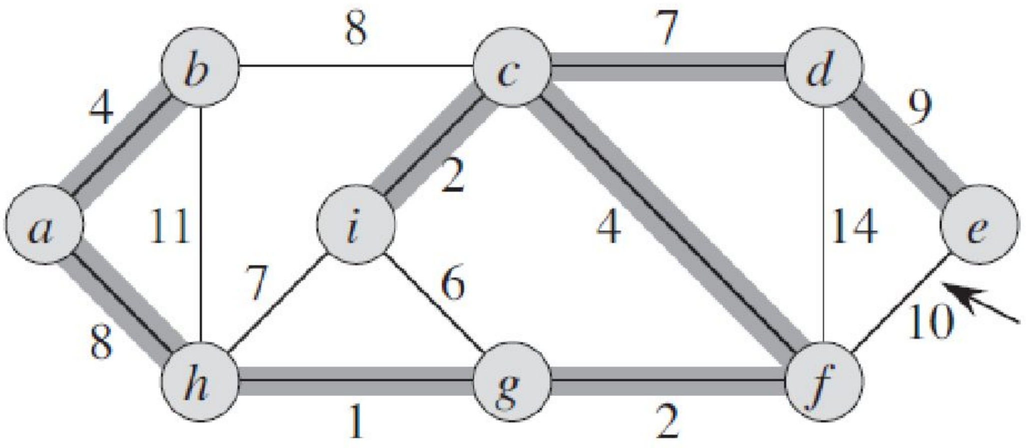
hg	ic	gf	ab	cf	cd	ah	de
1	2	2	4	4	7	8	9

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14

Algoritmo de Kruskal

hg	1
ic	2
gf	2
ab	4
cf	4
ig	6
cd	7
hi	7
ah	8
bc	8
de	9
fe	10
bh	11
df	14

(l)

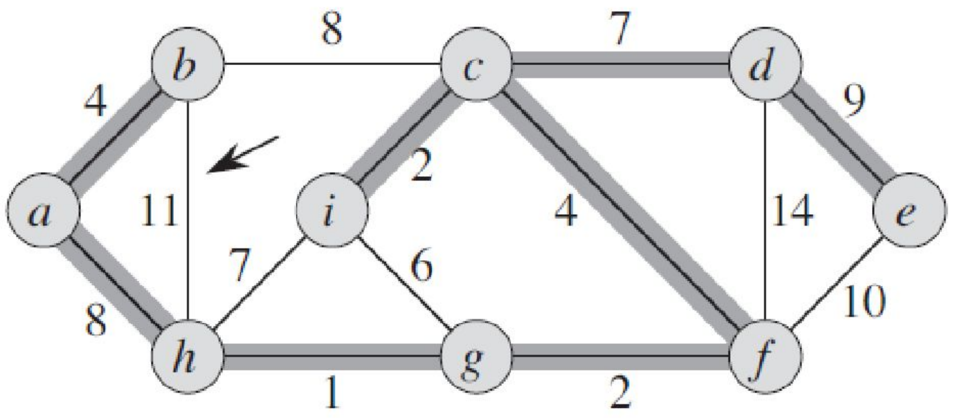


hg	ic	gf	ab	cf	cd	ah	de	Total
1	2	2	4	4	7	8	9	37

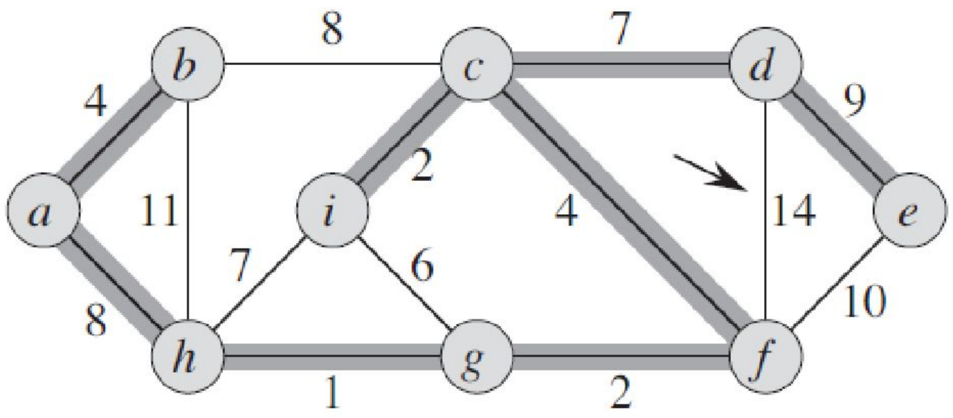
Algoritmo de Kruskal

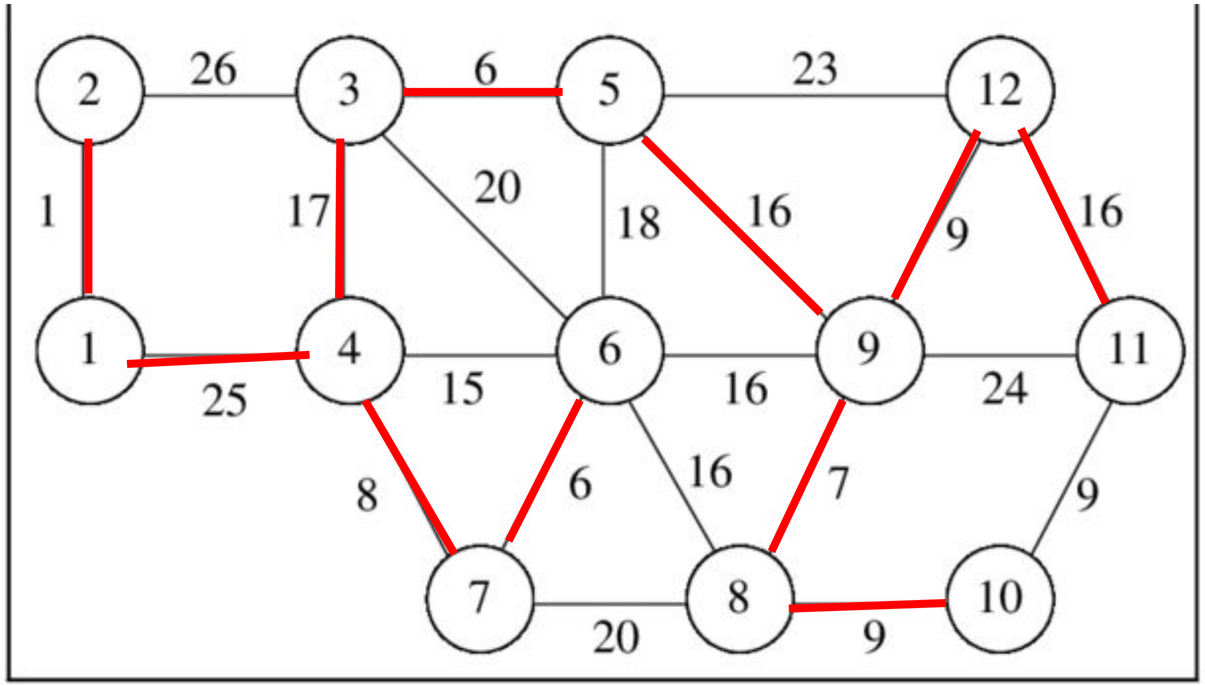
Ejemplo

(m)



(n)



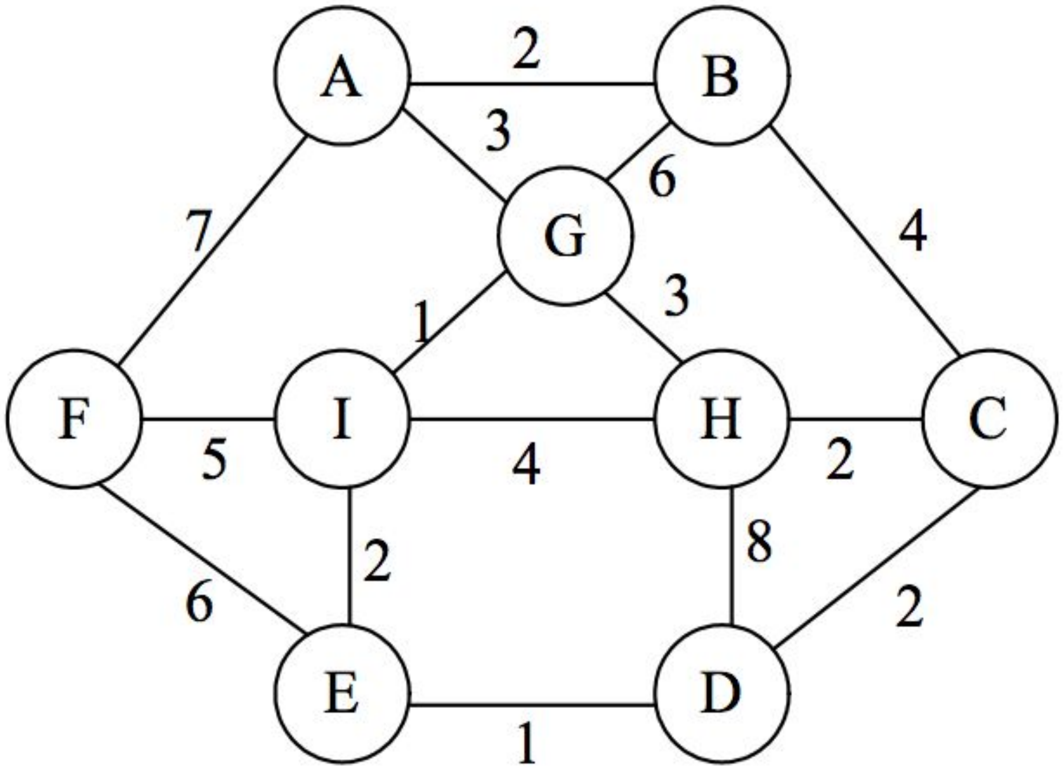


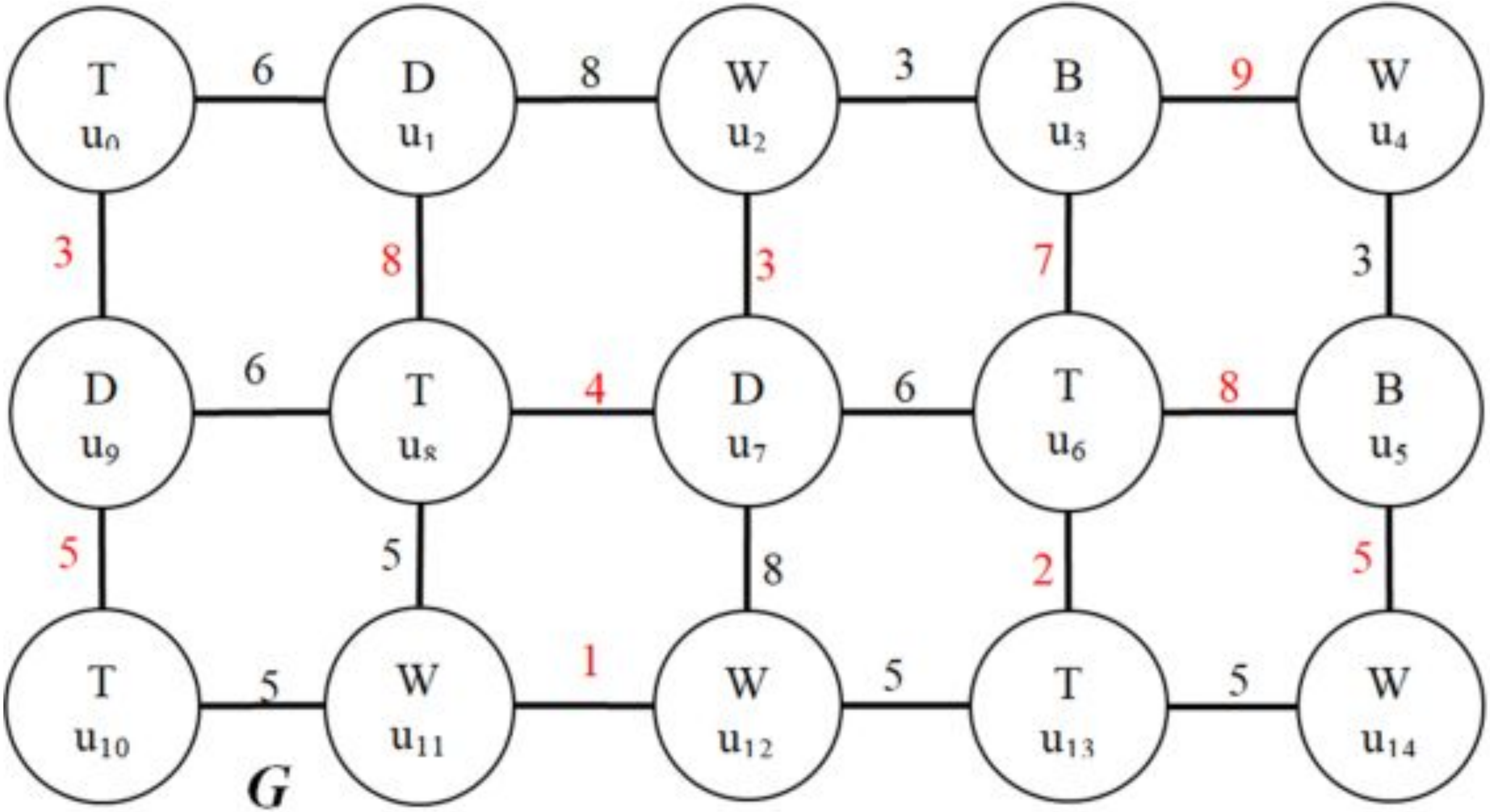
1,2	3,5	6,7	9,8	4,7	12,9	11,10	8,10
1	6	6	7	8	9	9	9

6,9	5,9	12,11	3,4	SUMA TOTAL
16	16	16	17	120

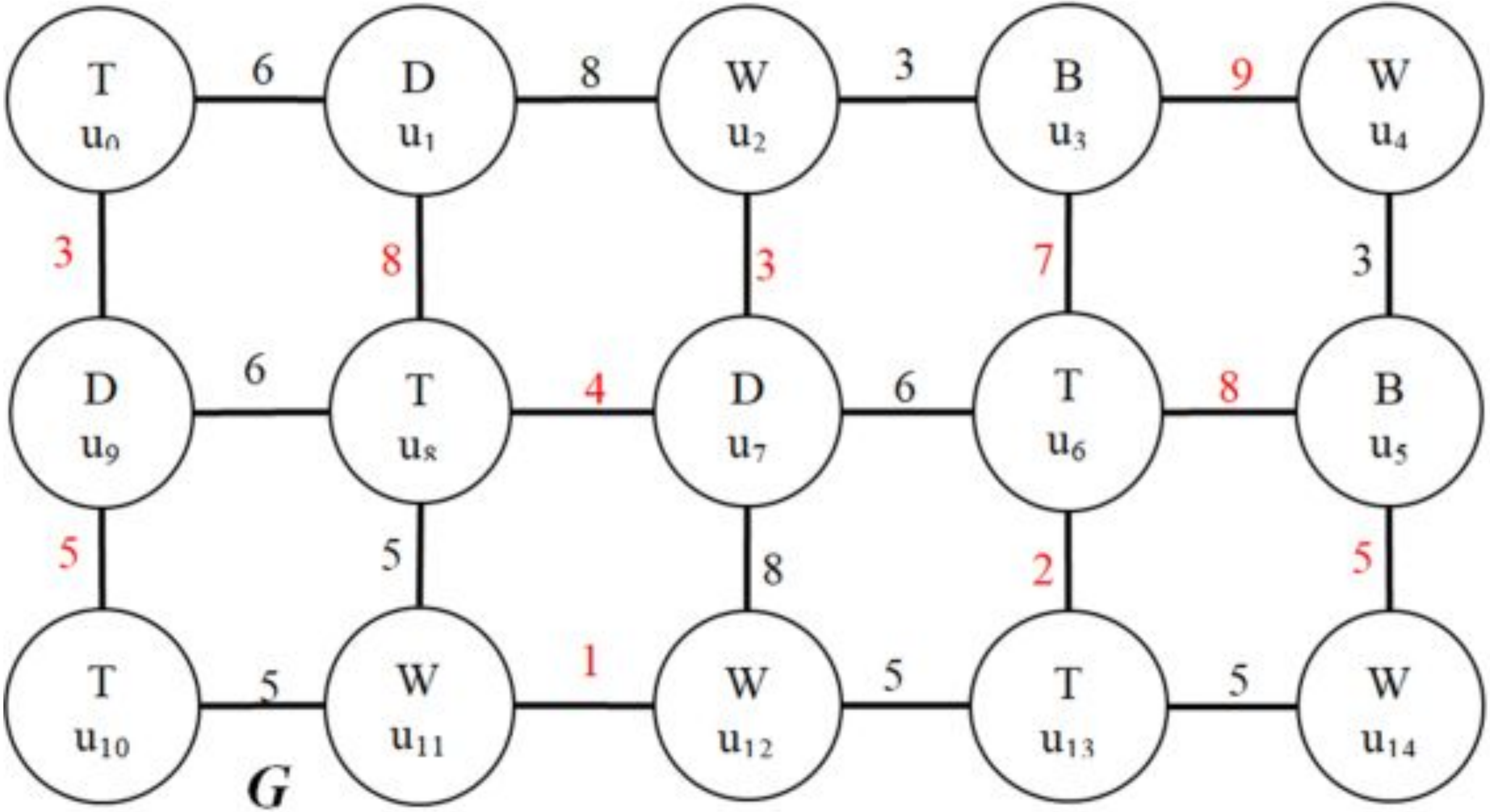
1,2	1
3,5	6
6,7	6
9,8	7
4,7	8
12,9	9
11,10	9
8,10	9
4,6	15
6,9	16
6,8	16
5,9	16
12,11	16
3,4	17

5,6	18
3,6	20
7,8	20
5,12	23
9,11	24
1,4	25
2,3	26


A
B
G
F
I
H
C
E
D
2
3
6
4
7
3
1
5
4
2
2
8
6
2
1



T
u0
6
D
u1
8
W
u2
3
B
u3
9
W
u4
3
8
3
7
3
D
u9
6
T
u8
4
D
u7
6
T
u6
8
B
u5
5
5
8
2
5
T
u10
5
W
u11
1
W
u12
5
T
u13
5
W
u14
G

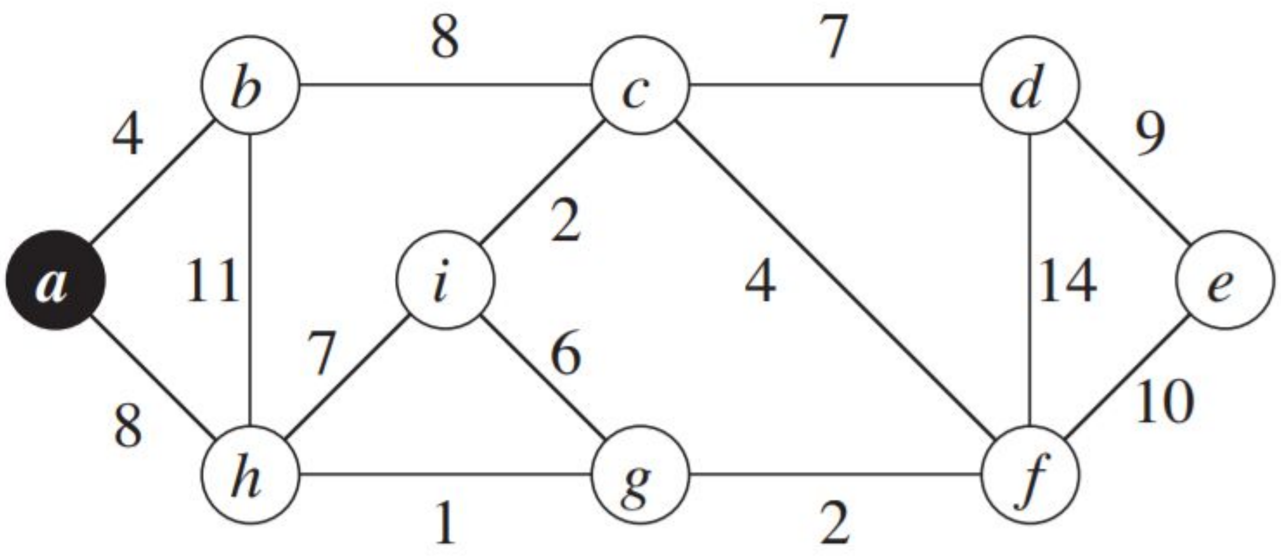

T
u_0
D
u_1
W
u_2
B
u_3
W
u_4
D
u_9
T
u_8
D
u_7
T
u_6
B
u_5
T
u_{10}
W
u_{11}
W
u_{12}
T
u_{13}
W
u_{14}
6
8
3
9
3
8
3
7
3
6
4
6
8
5
5
8
2
5
5
1
5
5
G

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-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

nodo	a	b	c	d	e	f	g	h	i
key	0	∞	∞	∞	∞	∞	∞	∞	∞
padre	-	-	-	-	-	-	-	-	-

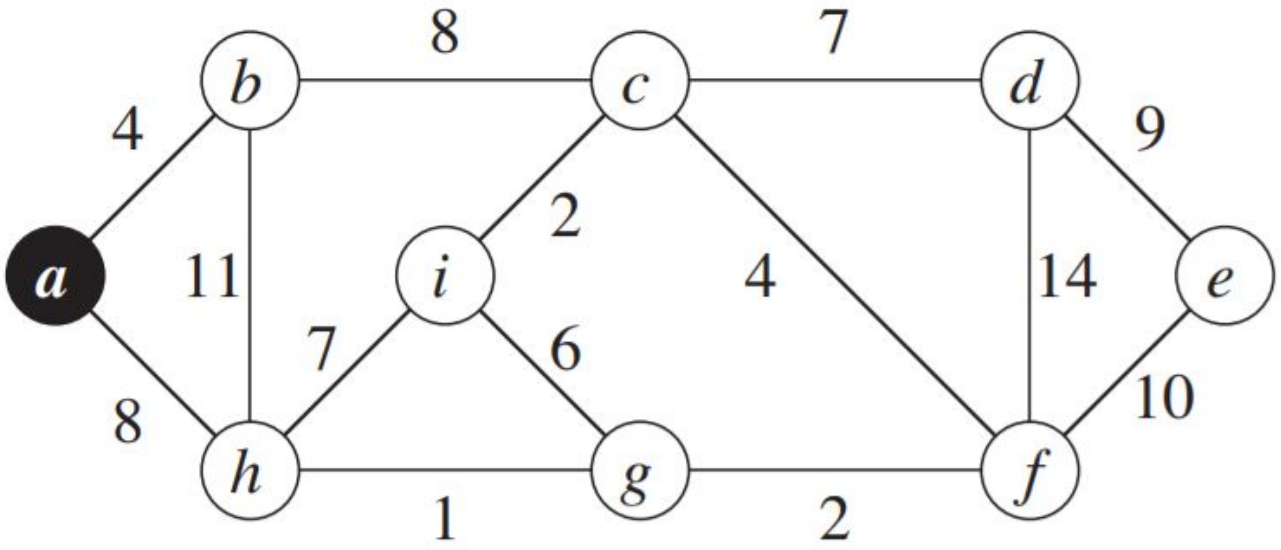
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```



nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞	∞	∞	∞	∞	∞/8	∞
padre	-	-/a	-	-	-	-	-	-/a	-

MST-PRIM(G, w, r)

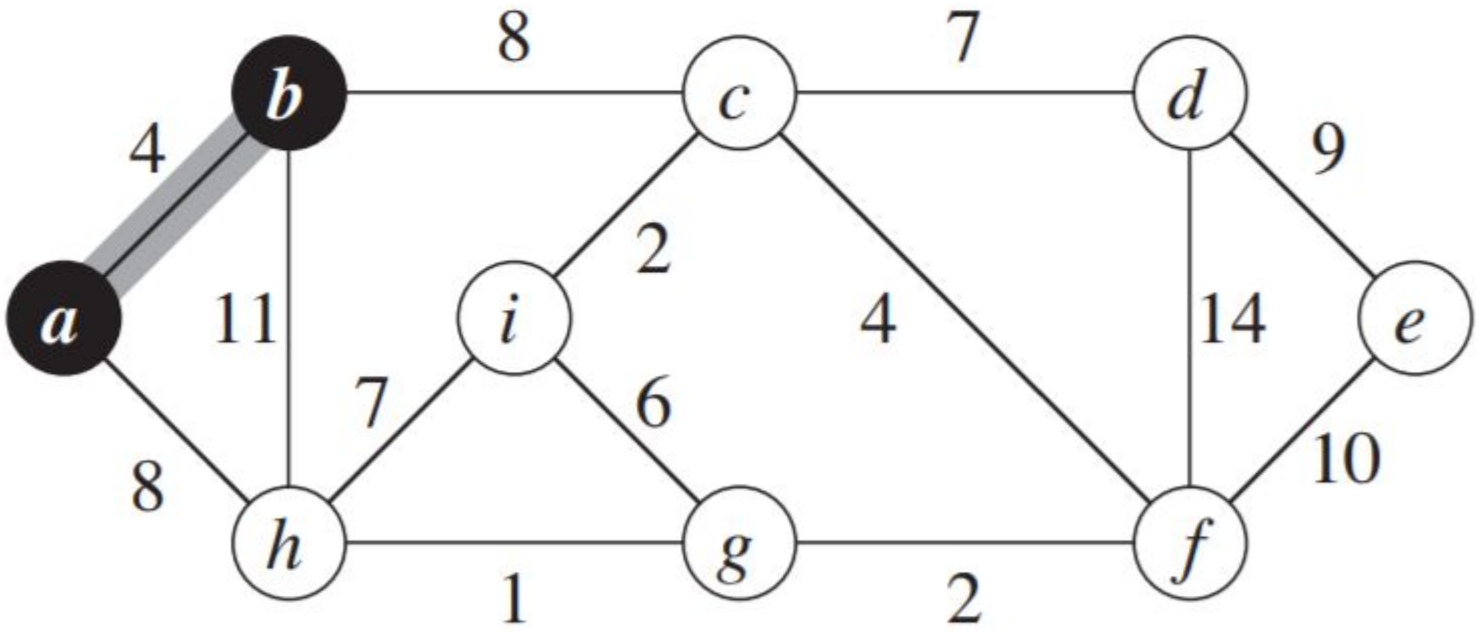
```
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```



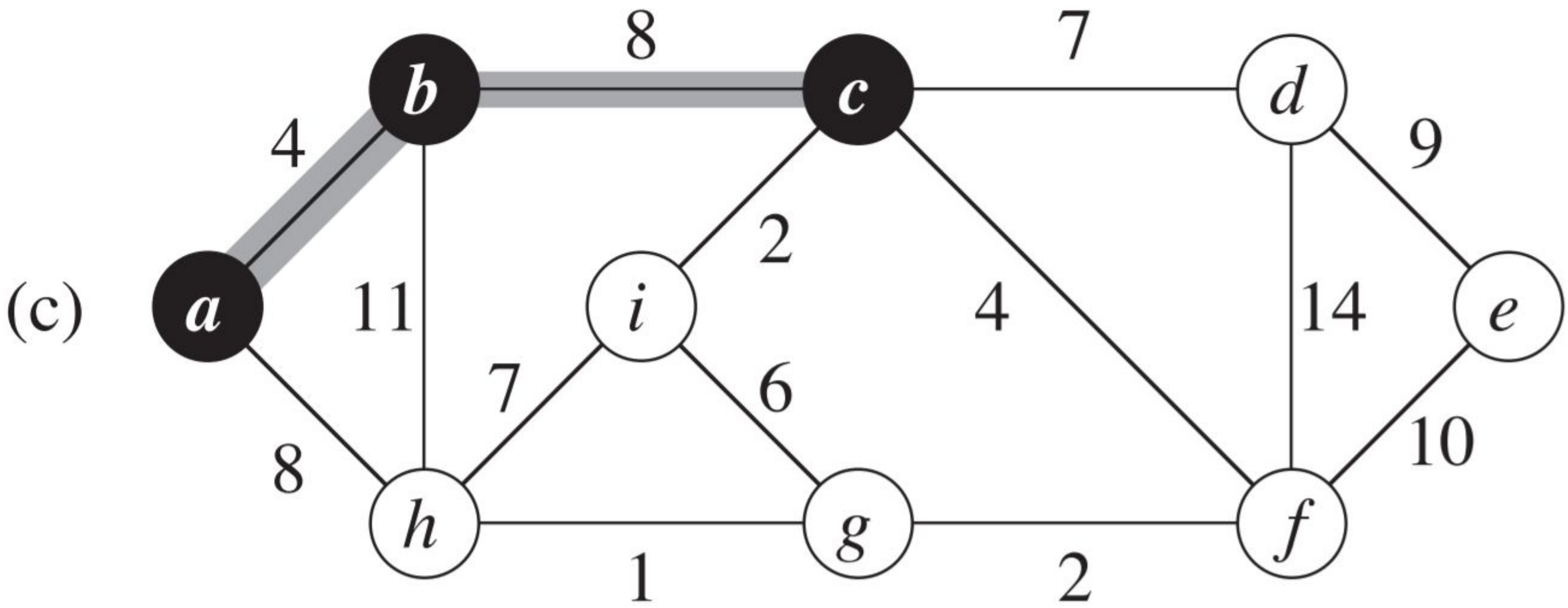
nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞	∞	∞	∞	∞/8	∞
padre	-	-/a	-/b	-	-	-	-	-/a	-

MST-PRIM(G, w, r)

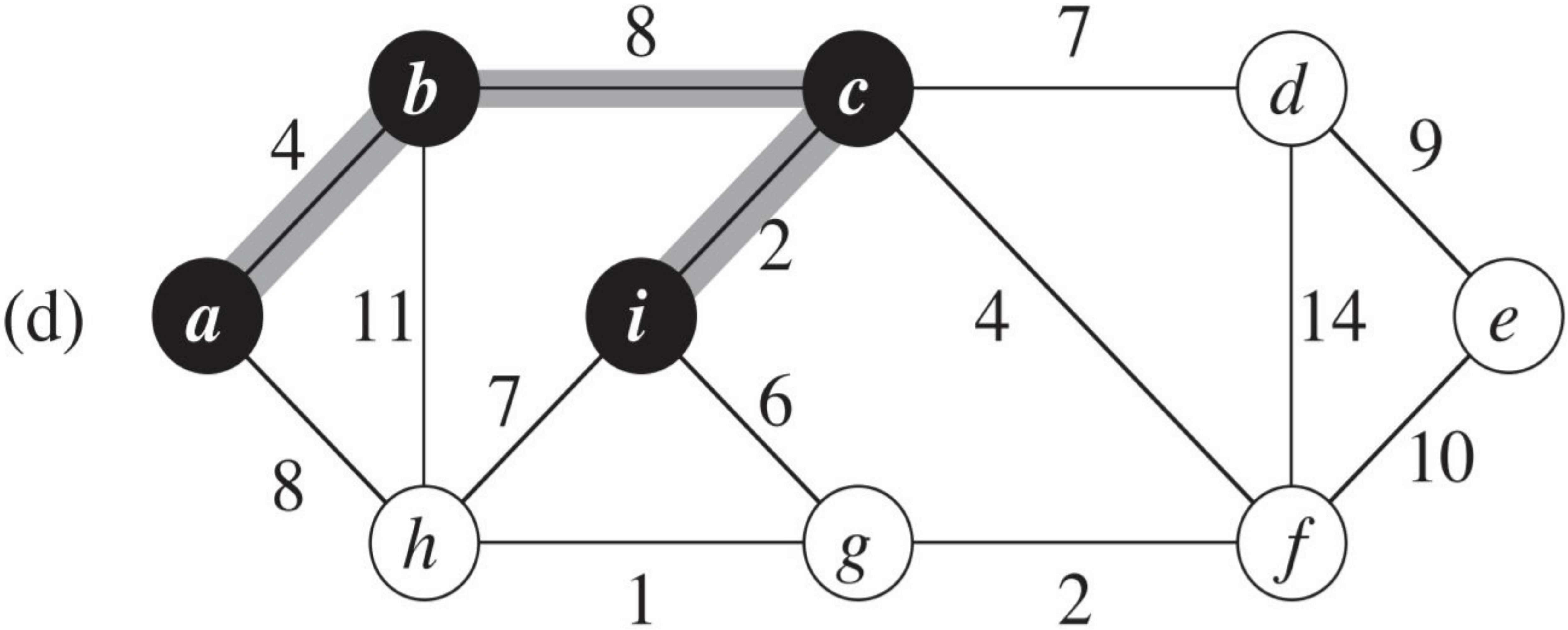
```
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```



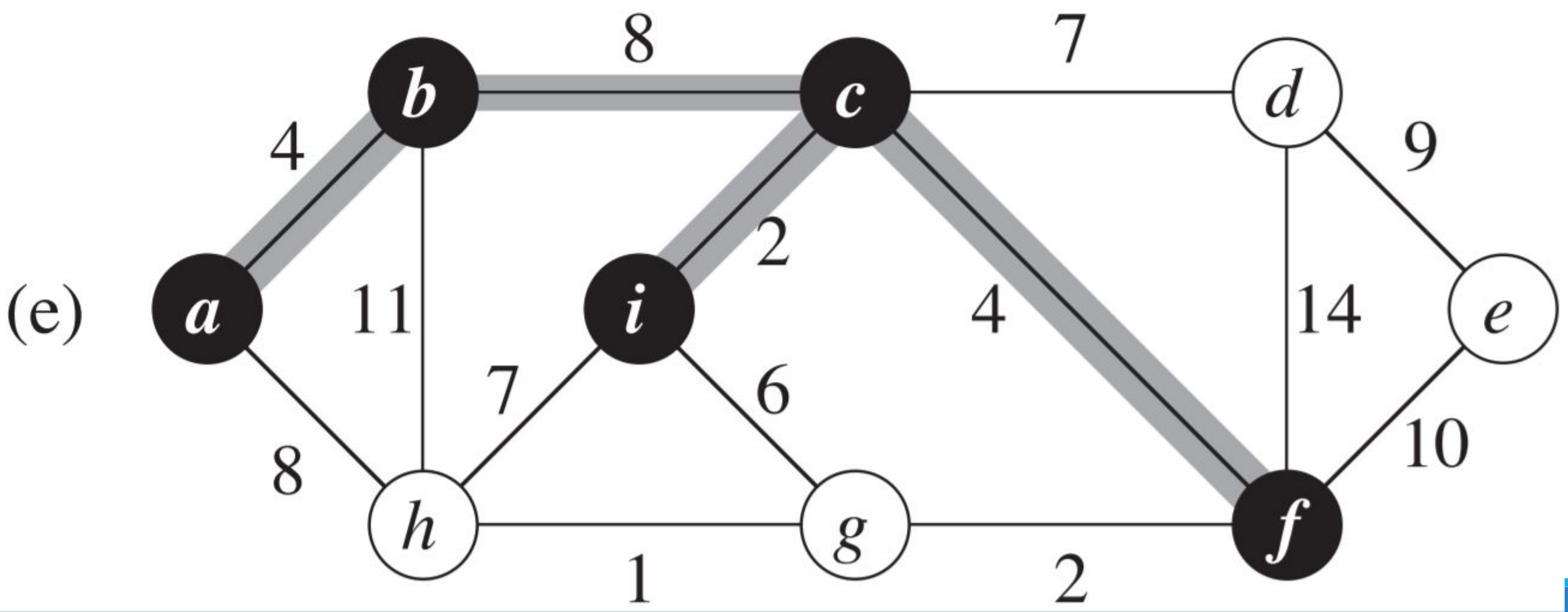
nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞	∞/4	∞	∞/8	∞/2
padre	-	-/a	-/b	-/c	-	-/c	-	-/a	-/c



nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞	∞/4	∞/6	∞/8/7	∞/2
padre	-	-/a	-/b	-/c	-	-/c	-/i	-/a/i	-/c

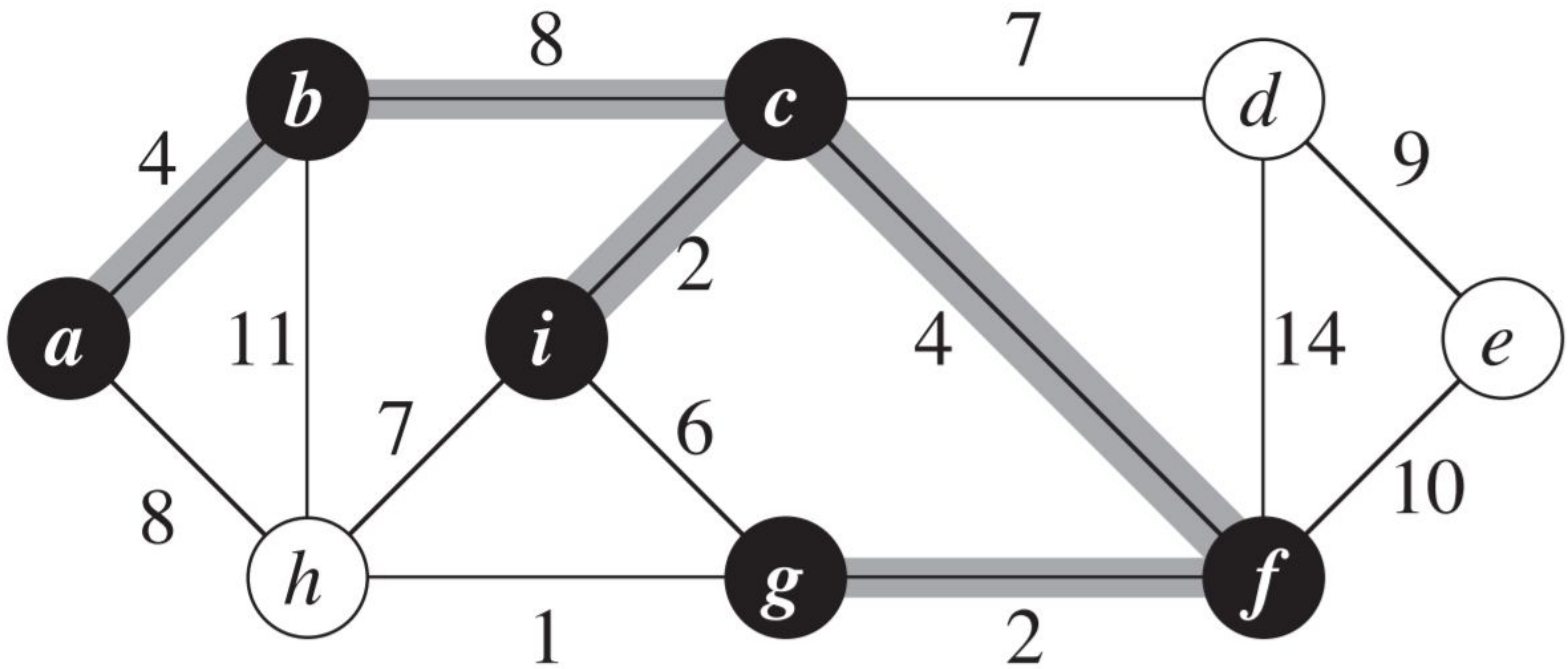


nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞/10	∞/4	∞/6/2	∞/8/7	∞/2
padre	-	-/a	-/b	-/c	-/f	-/c	-/i/f	-/a/i	-/c



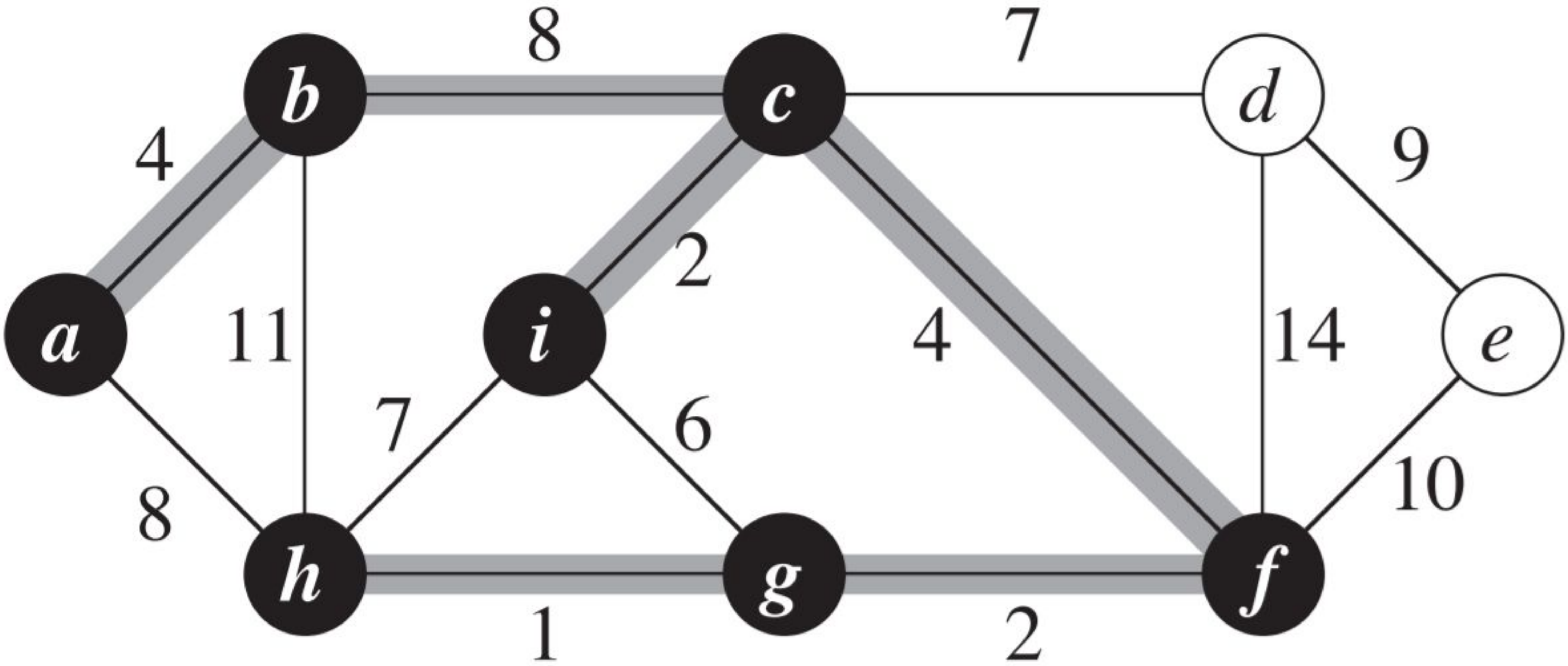
nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞/10	∞/4	∞/6/2	∞/8/7/1	∞/2
padre	-	-/a	-/b	-/c	-/f	-/c	-/i/f	-/a/i/g	-/c

(f)

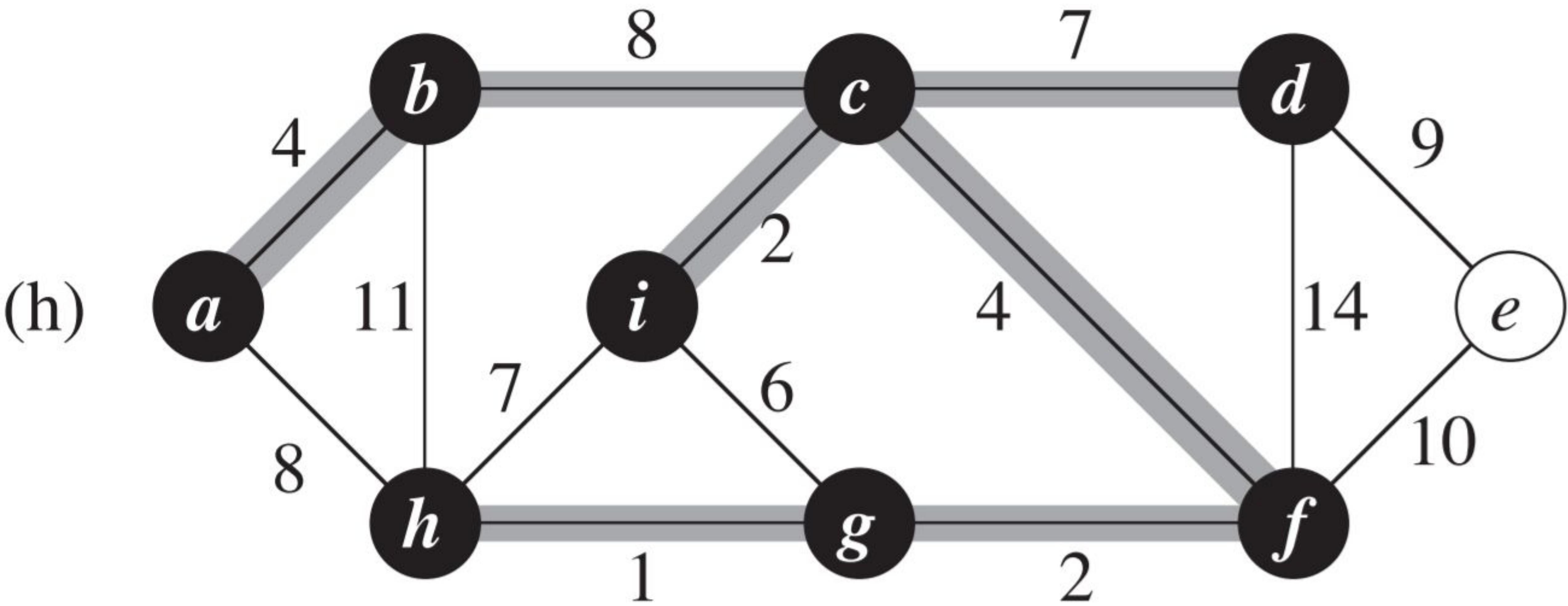


nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞/10	∞/4	∞/6/2	∞/8/7/1	∞/2
padre	-	-/a	-/b	-/c	-/f	-/c	-/i/f	-/a/i/g	-/c

(g)

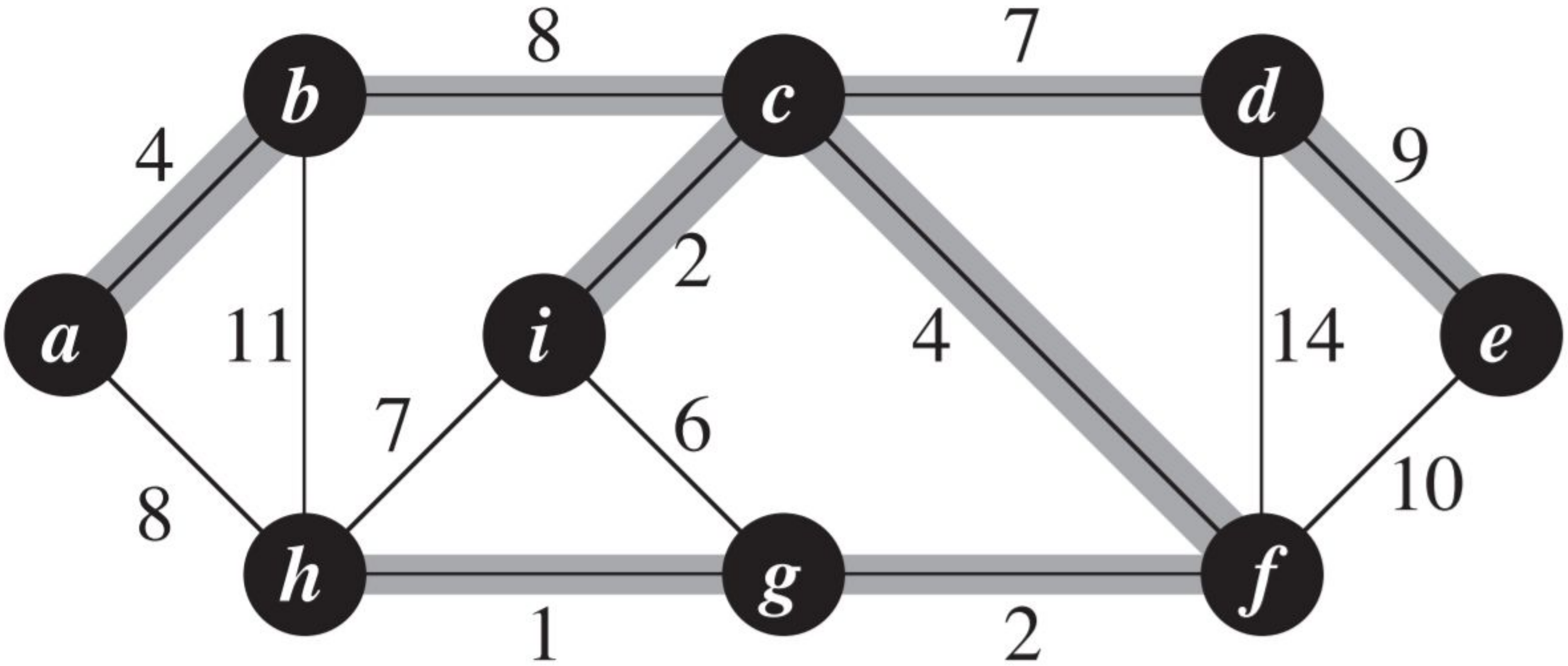


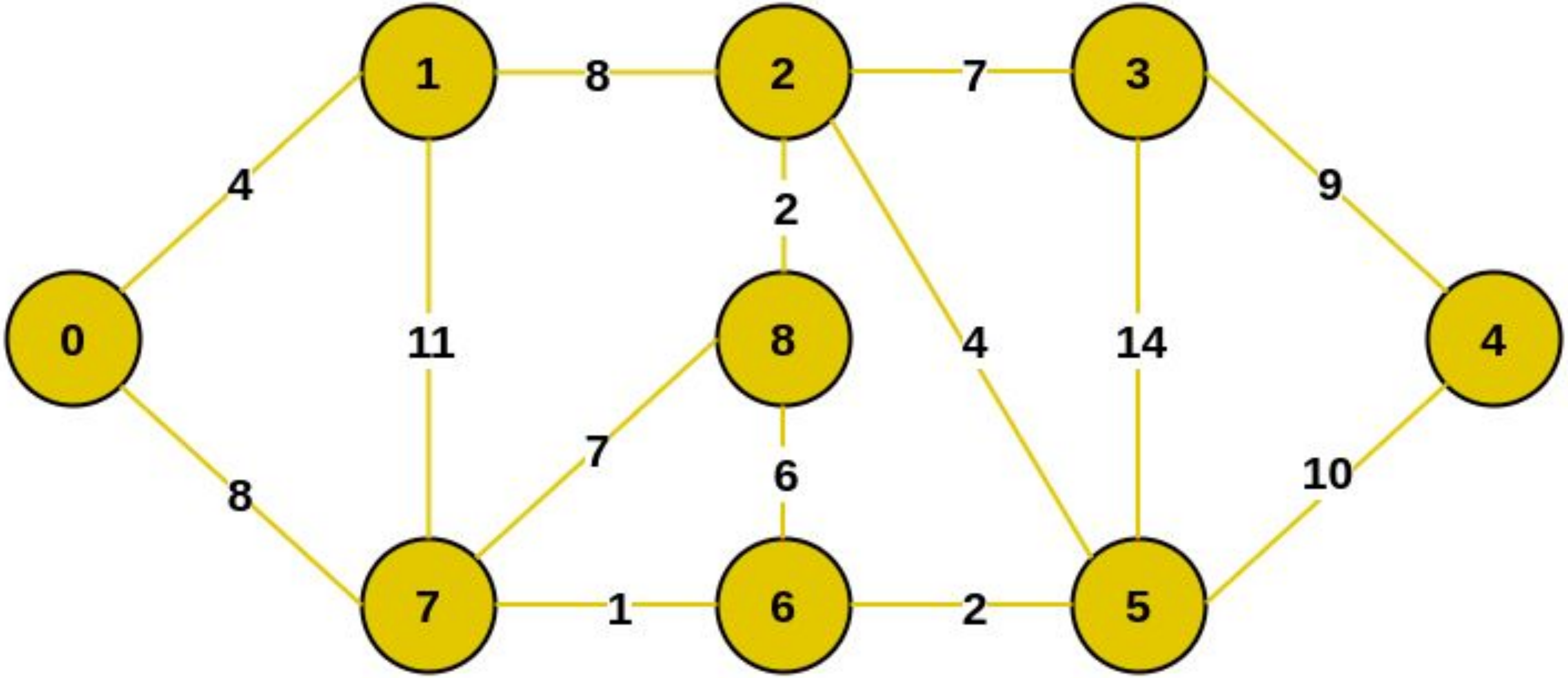
nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞/10/9	∞/4	∞/6/2	∞/8/7/1	∞/2
padre	-	-/a	-/b	-/c	-/f/d	-/c	-/i/f	-/a/i/g	-/c



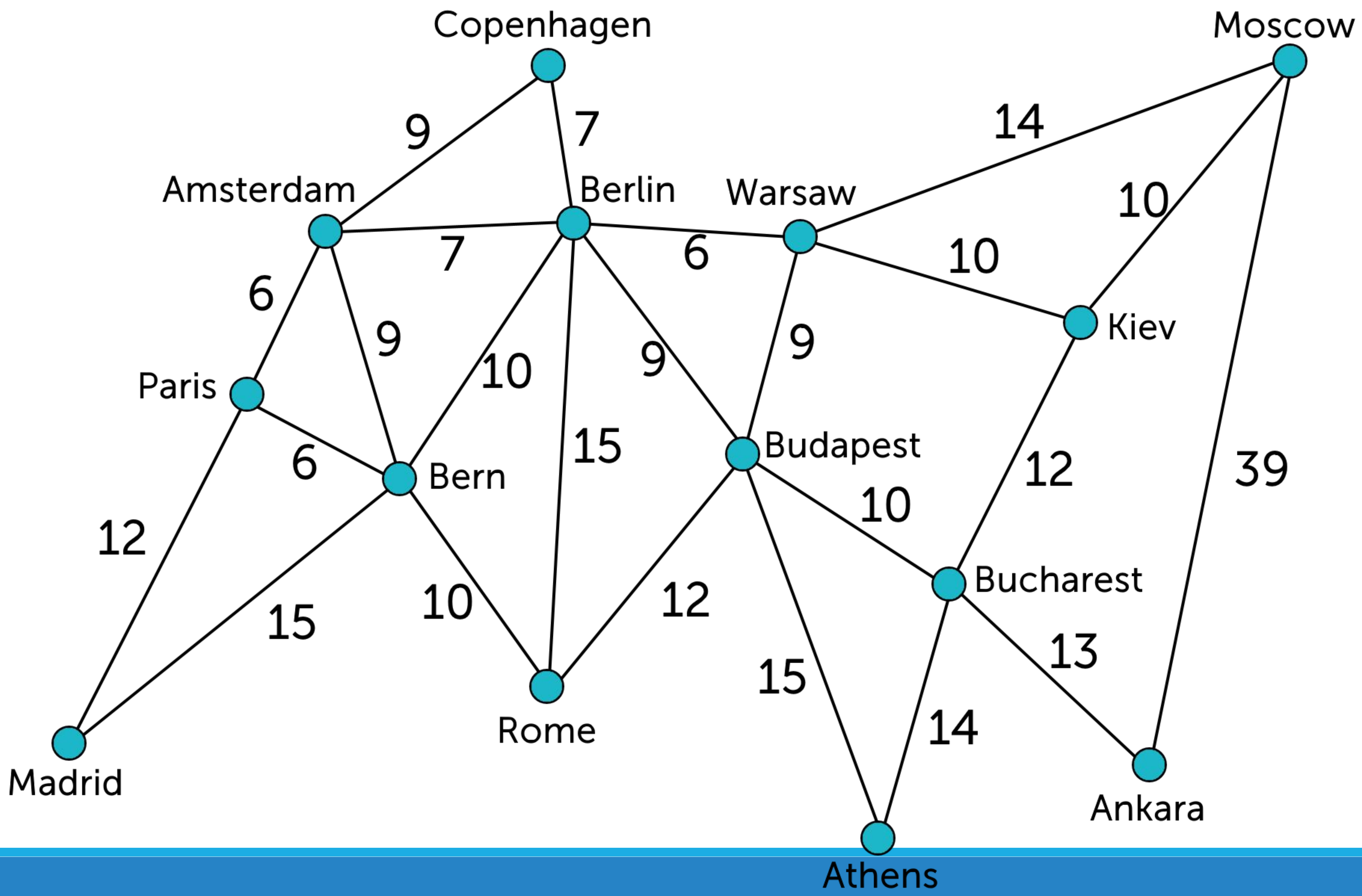
nodo	a	b	c	d	e	f	g	h	i
key	0	∞/4	∞/8	∞/7	∞/10/9	∞/4	∞/6/2	∞/8/7/1	∞/2
padre	-	-/a	-/b	-/c	-/f/d	-/c	-/i/f	-/a/i/g	-/c

(i)





Example of a Graph


Copenhagen
Moscow
9
7
14
Amsterdam
Berlin
Warsaw
10
7
6
10
6
9
9
Kiev
Paris
10
9
15
Budapest
6
Bern
12
39
10
12
Bucharest
10
15
12
13
15
Rome
14
Madrid
Ankara
Athens

Camino más corto

VARIANTES

- **Problema del camino más corto desde un nodo origen:** dado un grafo G=(V,E), se debe encontrar el camino más corto desde un vértice de origen dado $s \in V$ a cada vértice v $\in V$
- **Problema del camino más corto hacia un nodo destino:** Encontrar el camino más corto desde un vértice t hacia cada vértice v

VARIANTES

- **Problema del camino más corto entre un par de nodos:** Encontrar le camino más corto desde un vértice t hacia un vértice dado v
- **Problema del camino más corto entre todos los pares de nodos:** Encontrar el camino más corto para cada par de nodos u y v del grafo

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

- DIJKSTRA(G, w, s)
  - INITIALIZE-SINGLE-SOURCE(G, s)
  - $S \leftarrow \emptyset$
  - $Q \leftarrow V[G]$
  - **while** $Q \neq \emptyset$
  - **do** $u \leftarrow$ EXTRACT-MIN(Q)
  - $S \leftarrow S \cup \{u\}$
  - **for** each vertex $v \in Adj[u]$
  - **do** RELAX(u, v, w)

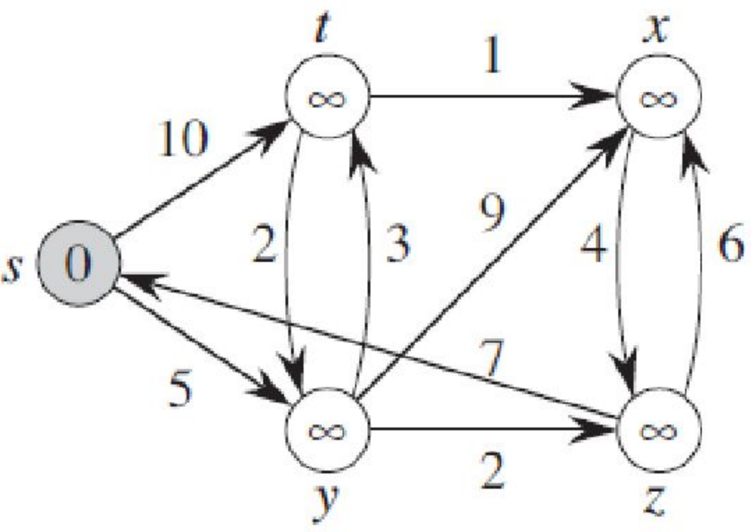
- INITIALIZE-SINGLE-SOURCE(G, s)
  - **for** each vertex $v \in V[G]$
  - **do** $d[v] \leftarrow \infty$
  - $\pi[v] \leftarrow$ NIL
  - $d[s] \leftarrow 0$

- RELAX(u, v, w)
  - **if** $d[v] > d[u] + w(u, v)$
  - **then** $d[v] \leftarrow d[u] + w(u, v)$
  - $\pi[v] \leftarrow u$

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

nodo	s	t	x	y	z
key	0	inf	inf	inf	nf
padre	-	-	-	-	-



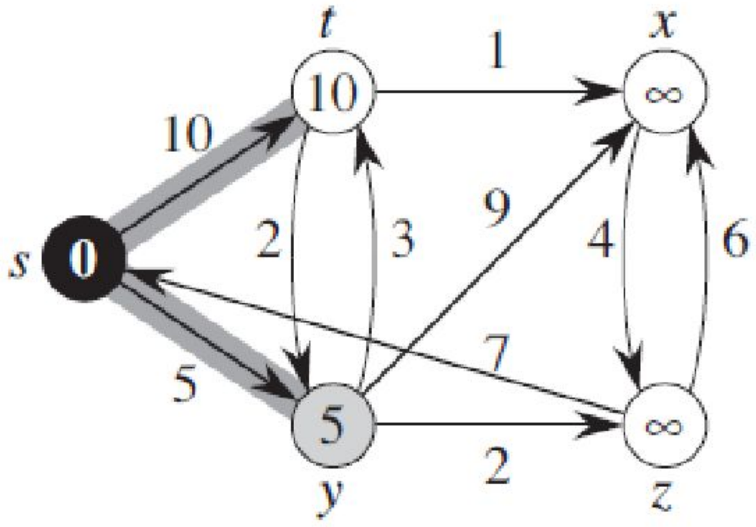
(a)

- RELAX(u, v, w)
  - **if** $d[v] > d[u] + w(u, v)$
  - **then** $d[v] \leftarrow d[u] + w(u, v)$
  - $\pi[v] \leftarrow u$

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

nodo	s	t	x	y	z
key	0	inf/10	nf	inf/5	nf
padre	-	-/s	-	-/s	-



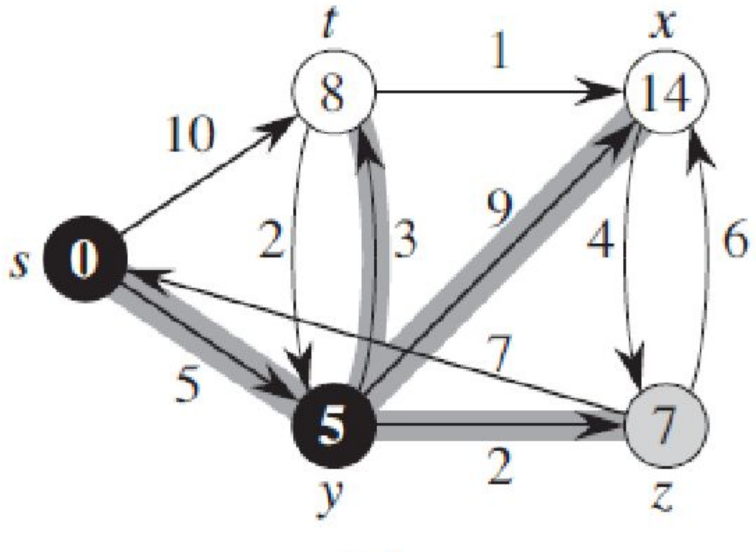
(b)

- RELAX(u, v, w)
  - 1 **if** $d[v] > d[u] + w(u, v)$
  - 2 **then** $d[v] \leftarrow d[u] + w(u, v)$
  - 3 $\pi[v] \leftarrow u$

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

nodo	s	t	x	y	z
key	0	inf/10/8	inf/14	inf/5	inf/7
padre	-	-/s/y	-/y	-/s	-/y



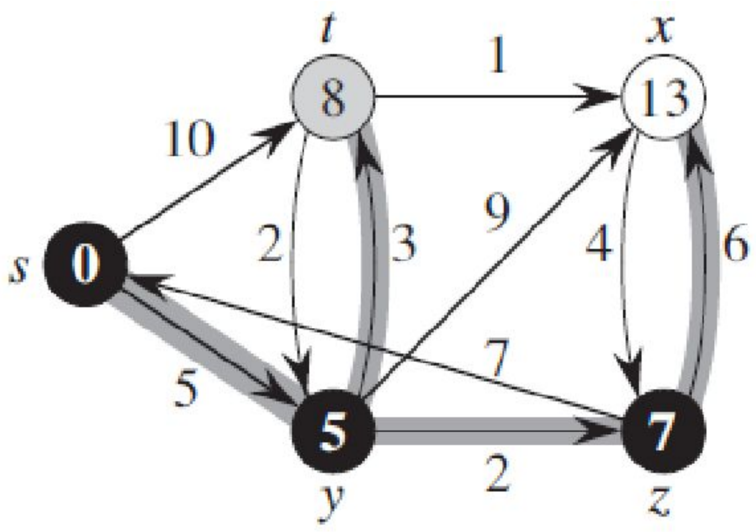
(c)

- RELAX(u, v, w)
  - 1 **if** $d[v] > d[u] + w(u, v)$
  - 2 **then** $d[v] \leftarrow d[u] + w(u, v)$
  - 3 $\pi[v] \leftarrow u$

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

nodo	s	t	x	y	z
key	0	inf/10/8	inf/14/13	inf/5	inf/7
padre	-	-/s/y	-/y/z	-/s	-/y

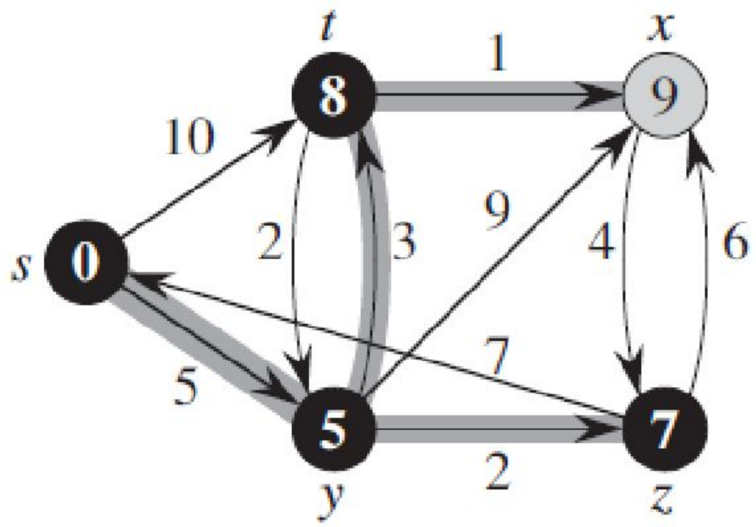


(d)

- RELAX(u, v, w)
  - 1 **if** $d[v] > d[u] + w(u, v)$
  - 2 **then** $d[v] \leftarrow d[u] + w(u, v)$
  - 3 $\pi[v] \leftarrow u$

Algoritmo de Dijsktra

Algoritmo de Dijkstra:

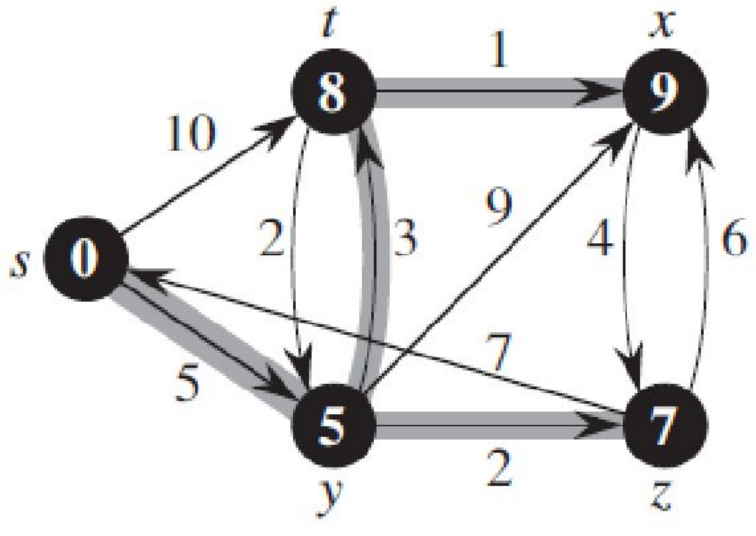


(e)

nodo	s	t	x	y	z
key	0	inf/10/8	inf/14/13/9	inf/5	inf/7
padre	-	-/s/y	-/y/z/t	-/s	-/y

Algoritmo de Dijsktra

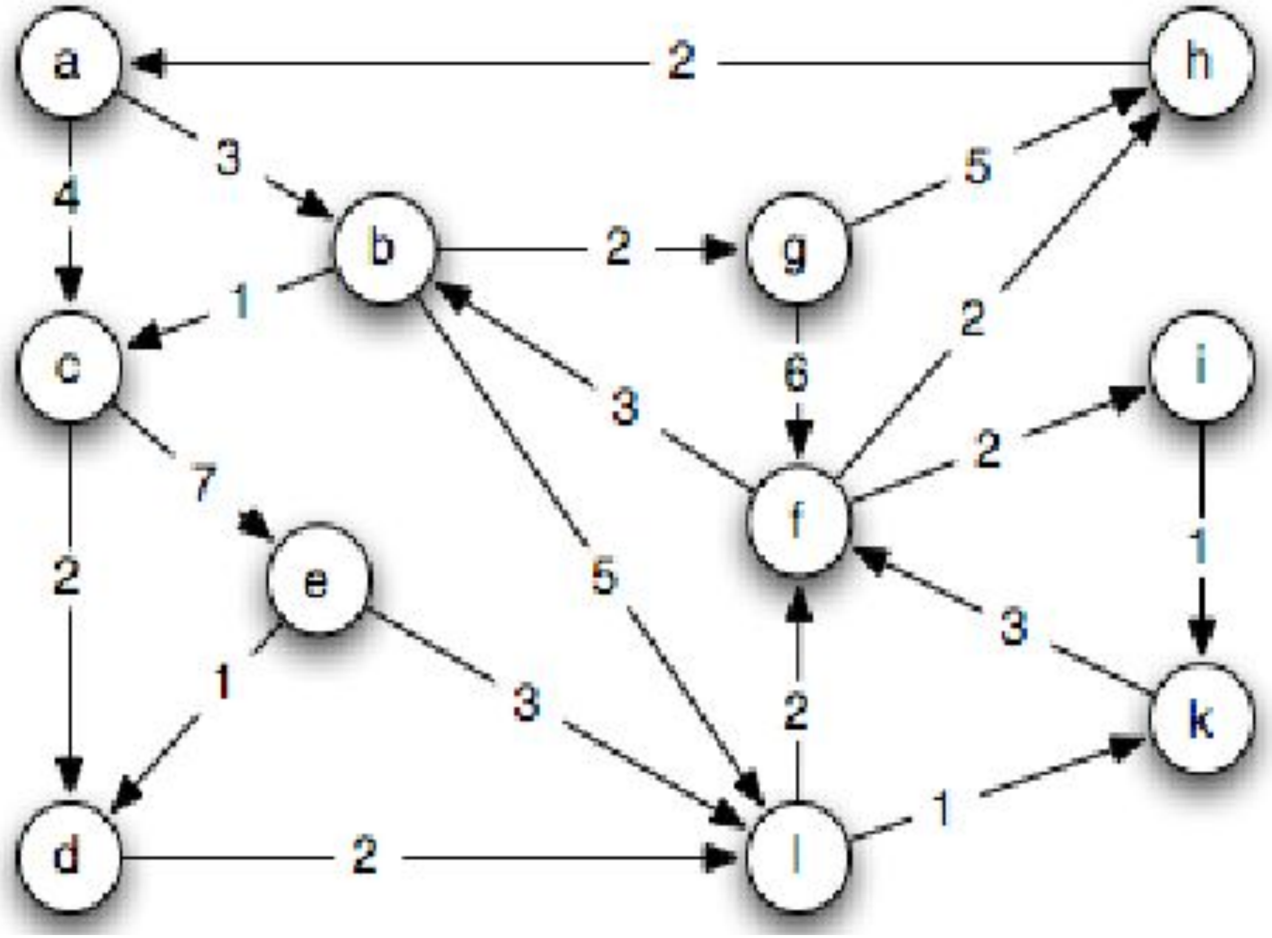
Algoritmo de Dijkstra:



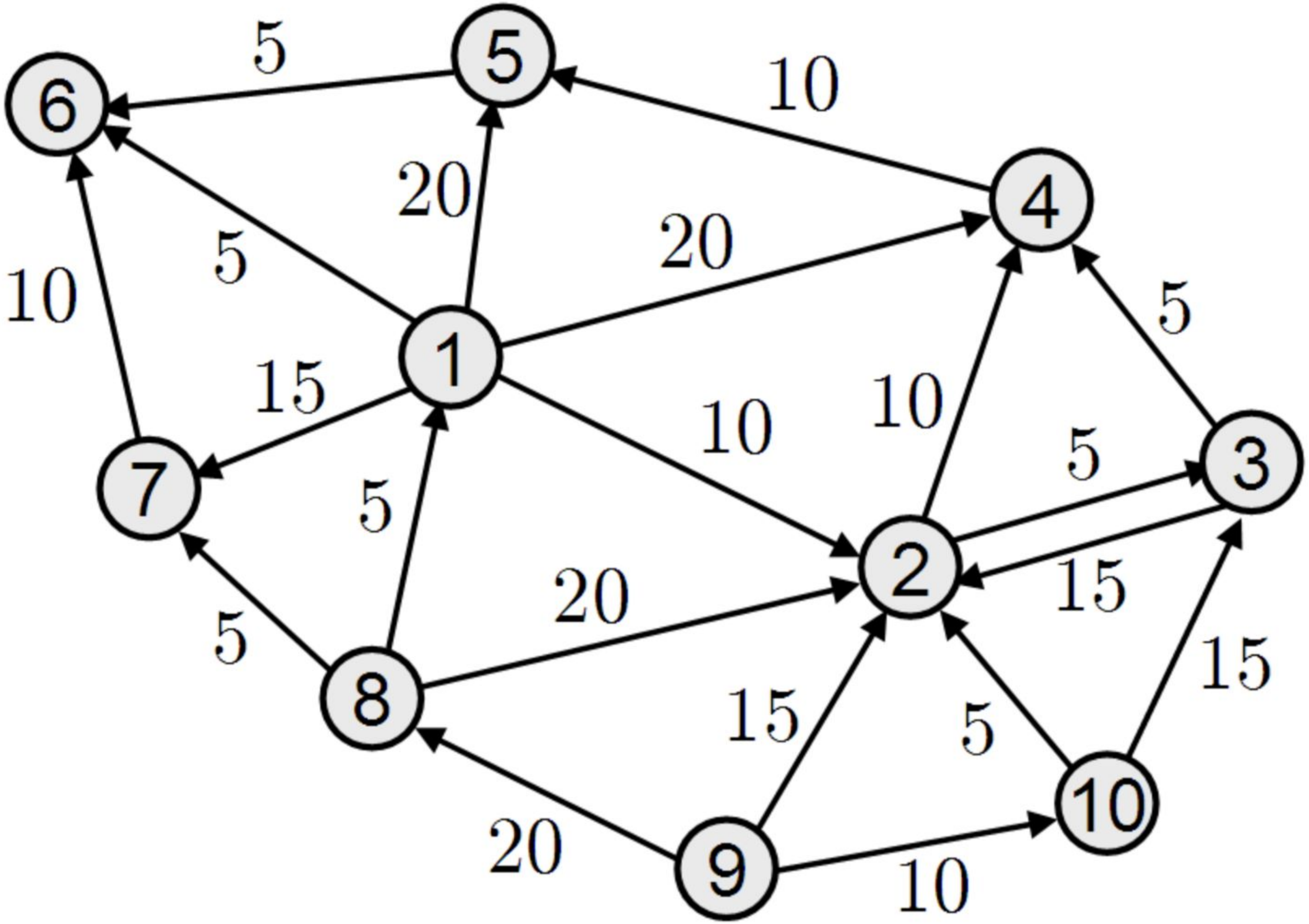
(f)

nodo	s	t	x	y	z
key	0	inf/10/8	inf/14/13/9	inf/5	inf/7
padre	-	-/s/y	-/y/z/t	-/s	-/y

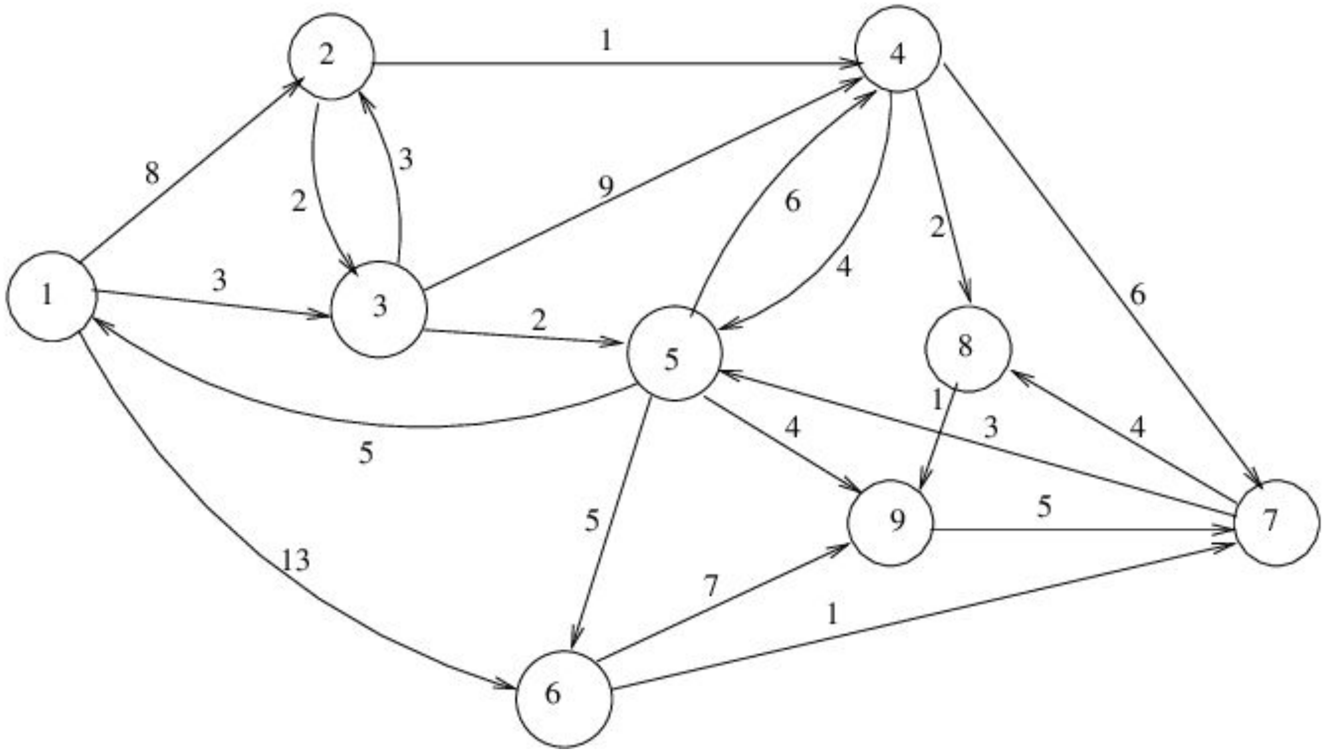
1a, 1g, 1b



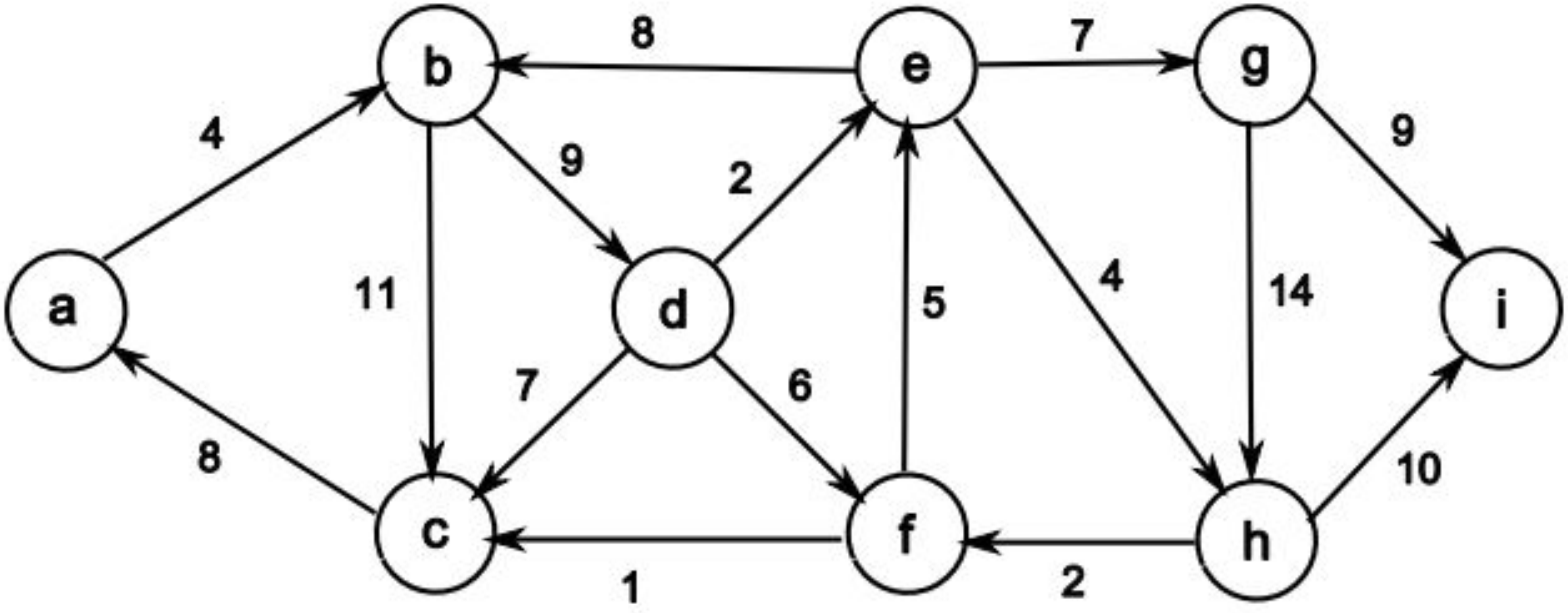
2-1,2-8,2-3



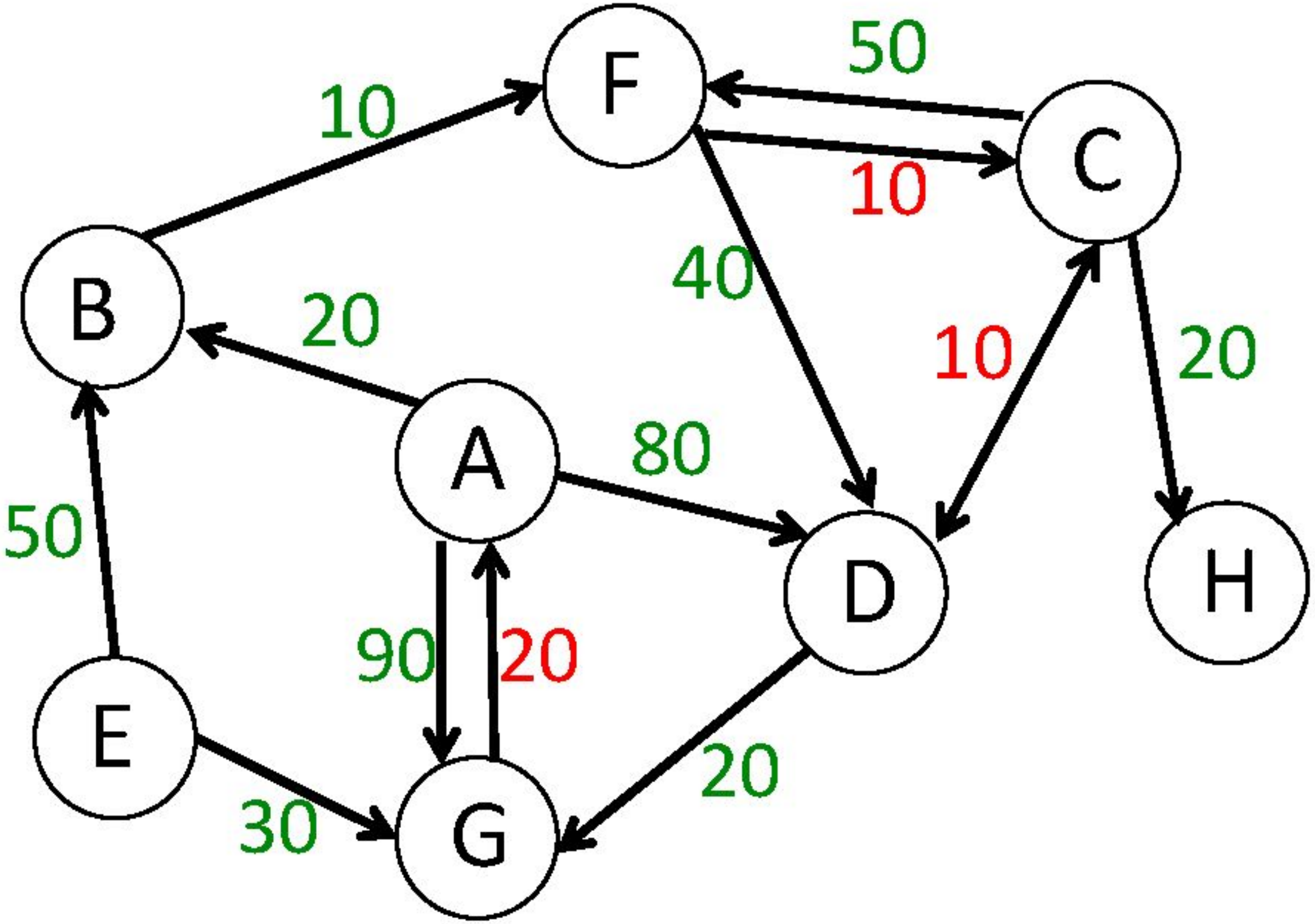
3-1, 3-5



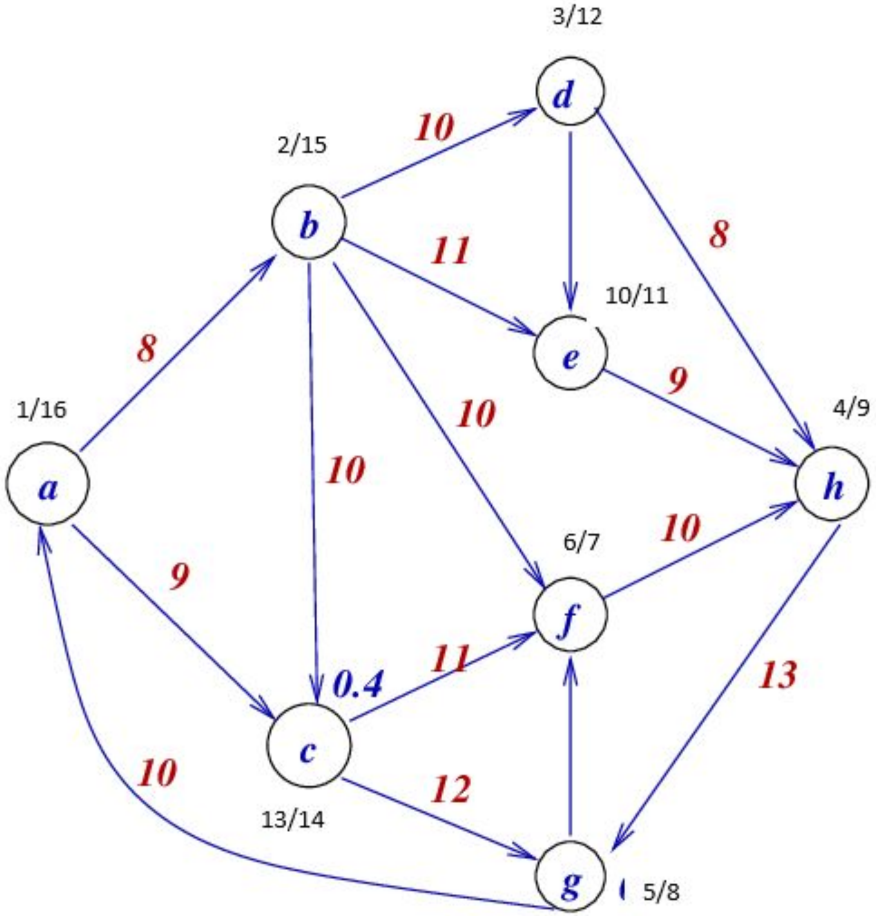
4a , 4e



5a



6a,



3/12
d
10
2/15
b
11
8
10/11
e
8
9
1/16
10
4/9
a
10
h
10
6/7
9
f
11
0.4
13
c
10
12
13/14
g
5/8