Grafovski algoritmi

Minimalno povezujuće stablo

Razapinjuće stablo

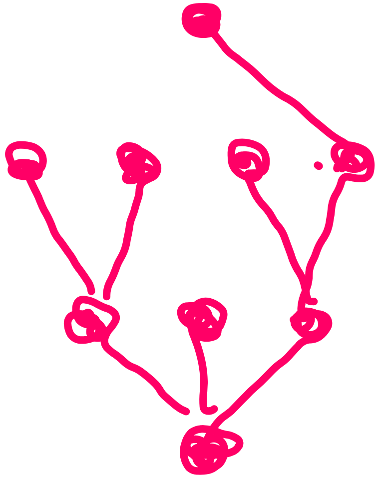
Uvod

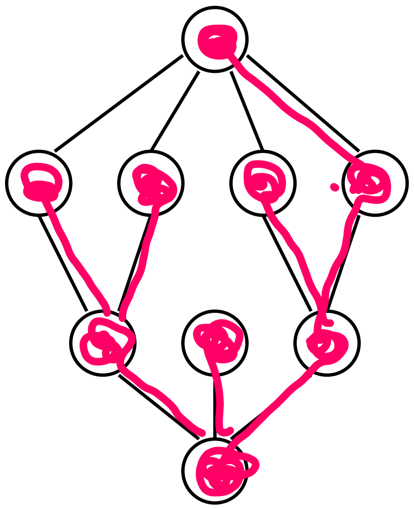
- Primena u elektronici, računarstvu, medicini, industriji, biologiji, poljoprivredi,...
  - Odrediti najjeftiniju realizaciju kućne elektroinstalacije,
  - Izgradnja putne mreže između više gradova, planiranje saobraćaja
  - Mrežno povezivanje računara
- Problem se definiše za težinske grafove
  - Težine su pridružene granama

Razapinjuće (povezujuće) stablo

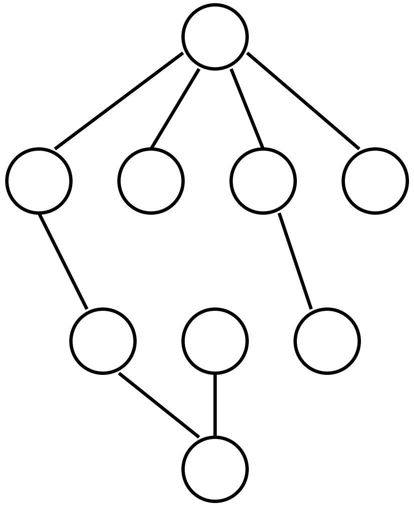
- Za dati povezan graf G(V,E) se definiše razapinjuće stablo T(V’,E’) kao:
  - T je podgraf od G, tj. V’ $\subseteq$ V, E’ $\subseteq$ E.
  - T povezuje sve čvorove grafa G (V’ = V)
  - T je stablo (bez ciklusa) |E’| = |V| -1

- Ukoliko je inicijalni graf nepovezan onda se može problem razmatrati na svim povezanim komponentama odvojeno

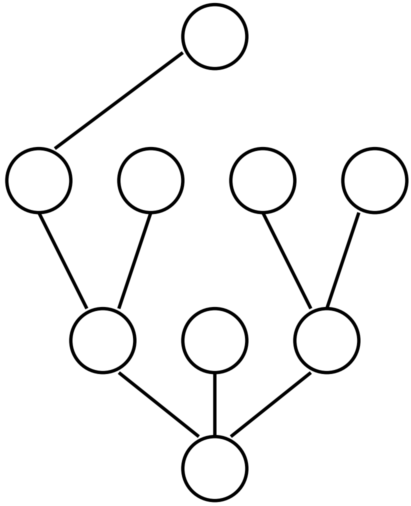




Neusmeren graf



Jedan mogući rezultat BFS početni čvor - vrh

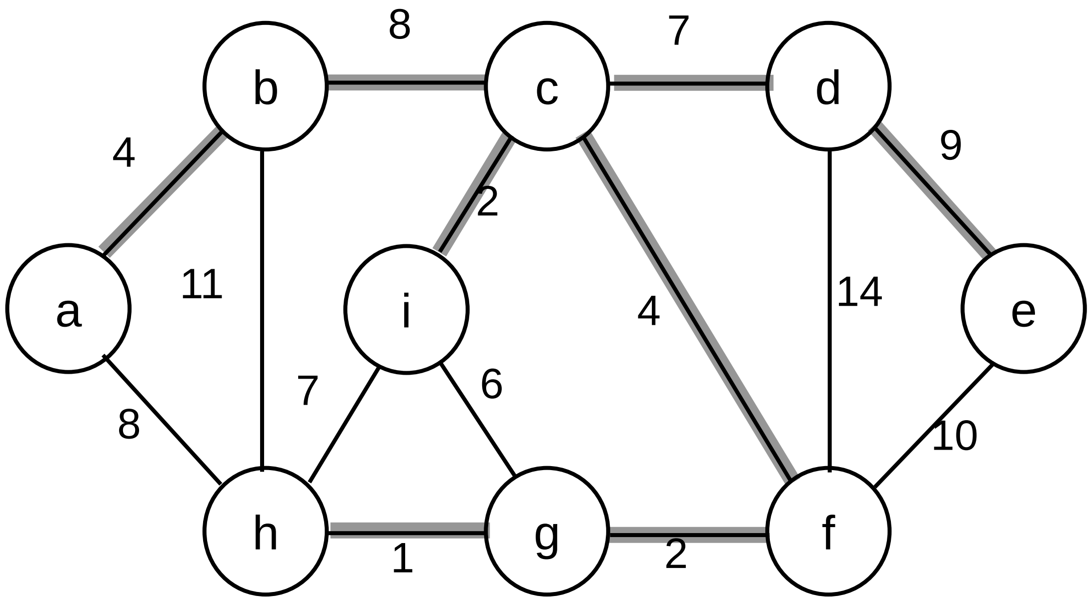


Jedan mogući rezultat DFS početni čvor - vrh

Minimalno razapinjuće stablo

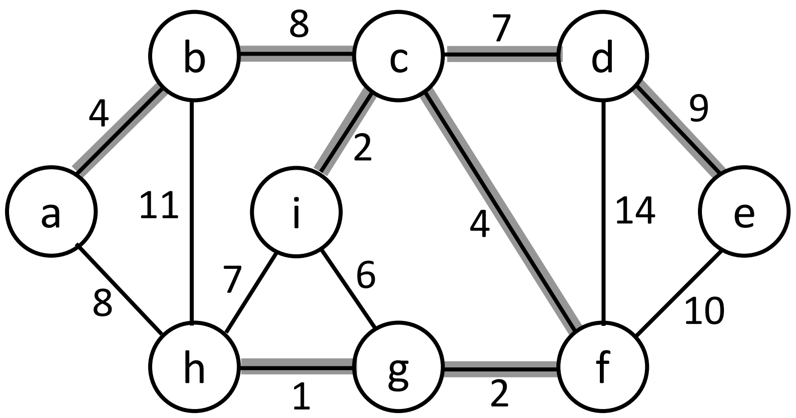
Neka je zadat neusmeren graf G(V,E) gde je V skup čvorova, a E skup grana (mogućih veza između čvorova) takvih da je za svaku granu $(\mathrm{u}, \mathrm{v}) \in E$ data težina $\mathrm{w}(\mathrm{u}, \mathrm{v})$ koja se posmatra kao cena. Potrebno je naći podgraf $T \subseteq E$ koji povezuje sve čvorove grafa E i čija je ukupna težina $\mathrm{w}(\mathrm{T}) = \sum_{u,v \in T} \mathrm{w}(\mathrm{u}, \mathrm{v})$ minimalna.

- rešenje ne mora biti jedinstveno



Nastajanje minimalnog razapinjućeg stabla

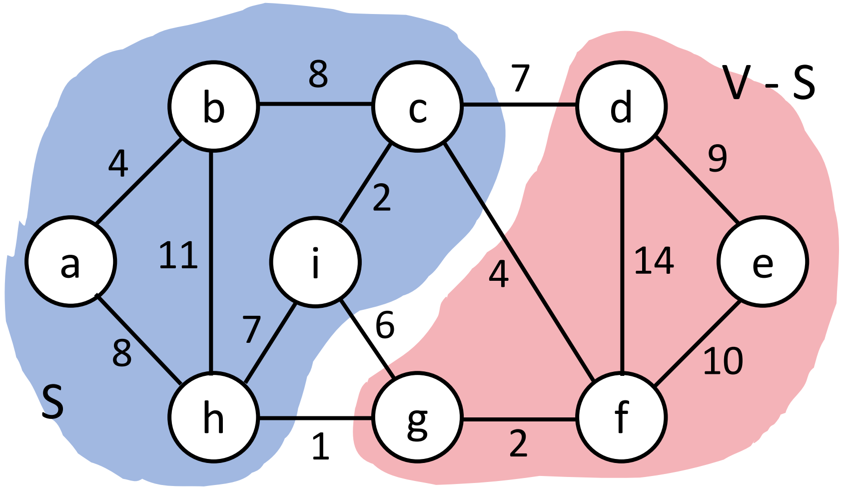
- Iterativno dodavanje grana u skup A (inicijalno prazan) da se dobije MST (Minimum Spanning Tree)
- Ideja: Dodaju se samo „sigurne“ (*safe*) grane
  - U svakoj iteraciji A je podskup nekog MST-a




```
GENERIC-MST(G,w)
A ← ∅
while A is not a spanning tree
    find an edge (u, v) that is safe for A
    A ← A ∪ {(u, v)}
return A
```

Kako pronaći „sigurne“ grane?

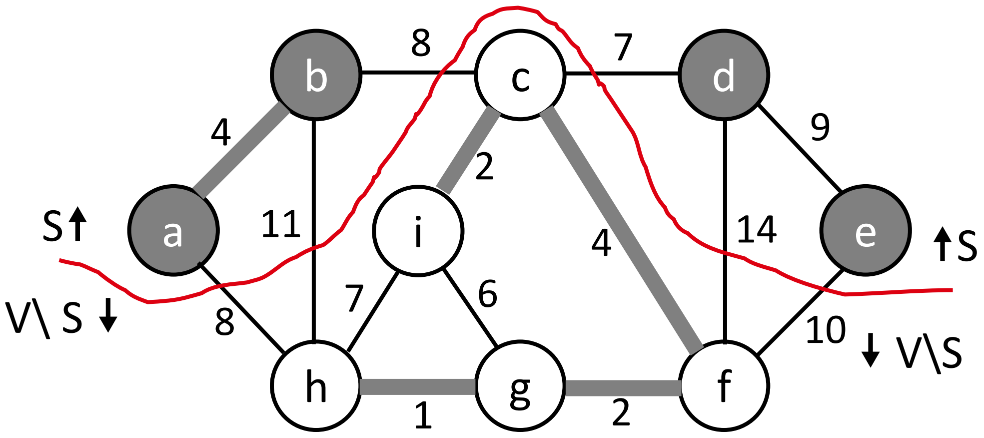
Pronalaženje „sigurnih“ grana



- Izabrati inicijalnu granu za graf A?
  - Grana sa najmanjom težinom (h, g)?
- Nakon toga, iterativno:
  - Neka je S $\subset$ V skup čvorova koji sadrži *h* ne sadrži *v* ($g \in V \backslash S$)
  - U bilo kom MST, ima bar jedna koja povezuje podgrafove S i $V \backslash S$
  - Zašto ne treba uvek birati granu sa minimalnom težinom (h,g)?

Definicije

- **Rez** - **cut** (S, V\S) je skup čvorova koji je podskup skupa V koji se nalaze da različitih strana skupova S i V\S
- **Prelazne grane** (**crosses**) reza (S, V \ S) imaju čvorove na različitim stranama (jedan u S, drugi u V\S)
- Rez **uvažava** (**respects**) skup grana $A \Leftrightarrow$ nema prelaznih grana iz A za rez
- Neka grana je **laka prelazna grana** (**light edge**) za rez $\Leftrightarrow$ njena težina je najmanja od svih prelaznih grana reza
  - Za dati rez može postojati više lakih prelaznih grana

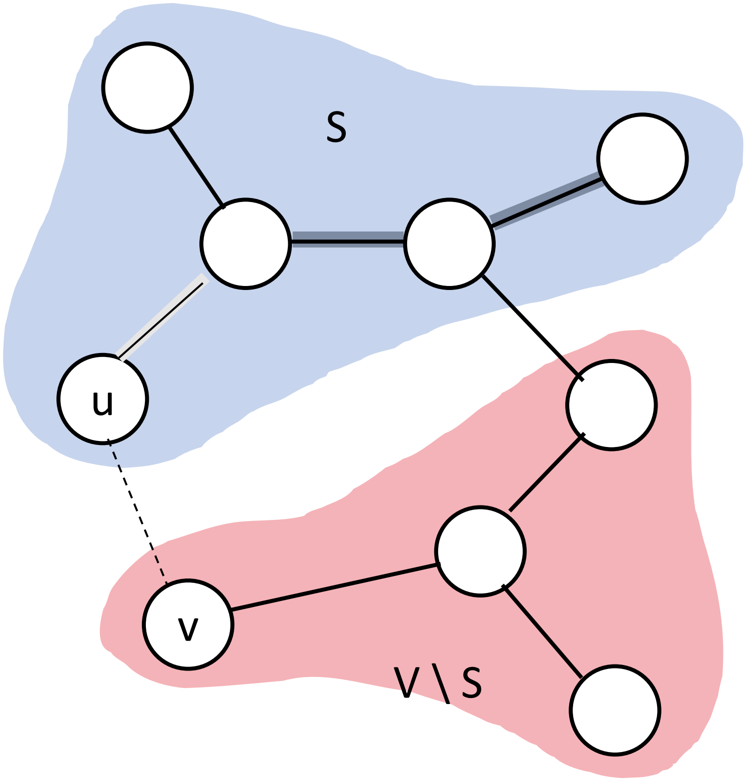


Pravilo

- Neka je $\mathcal{A}$ podskup nekog MST (T), (S, V \ S) postoji rez koji uvažava A
- Ako je (u, v) laka prelazna grana za (S, V\S) ⇒ (u, v) sigurna grana za $\mathcal{A}$.

Dokaz:

- Neka je T jedan MST koji sadrži A
  - Grane u A su zasenčene
- <u>Slučaj 1:</u> Ako T sadrži (u,v), tada ona može biti sigurna za A
- <u>Slučaj 2:</u> Pretpostavimo da T ne sadrži granu (u, v)
- **Ideja**: Konstruisati drugi MST T’ koji sadrži A ∪ {(u, v)}



Dokaz teoreme

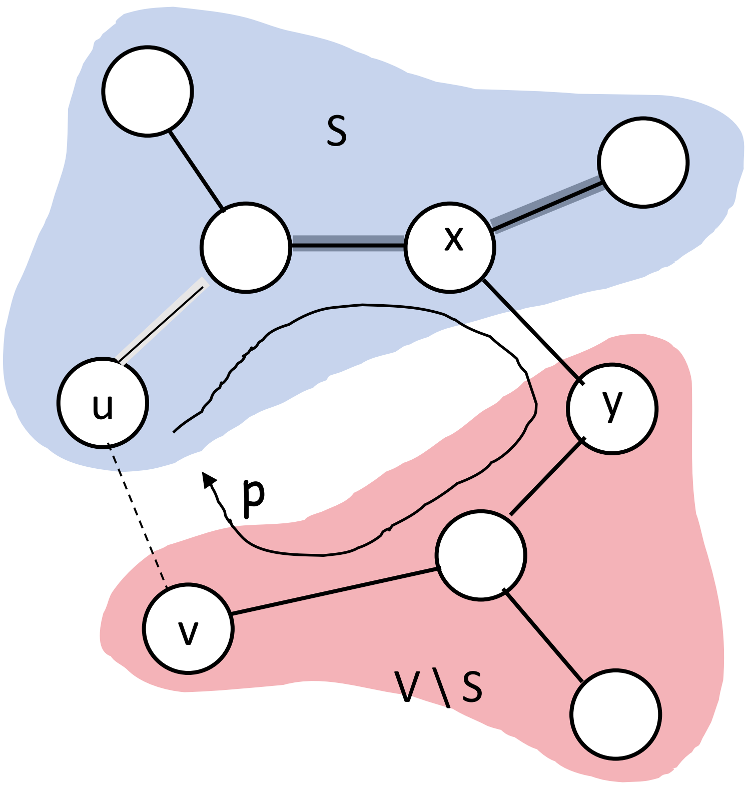
- T sadrži jedinstvenu putanju između p i v
- Putanja p mora da prelazi rez (S, V \ S) bar jednom:

  neka je (x, y) prelazna grana

Ako se ukloni (x,y) $\Rightarrow$ **deli se T na dve komponente**

- Ako se potom doda (u, v) povezuju se komponente

$$T' = T - \{(x, y)\} \cup \{(u, v)\}$$



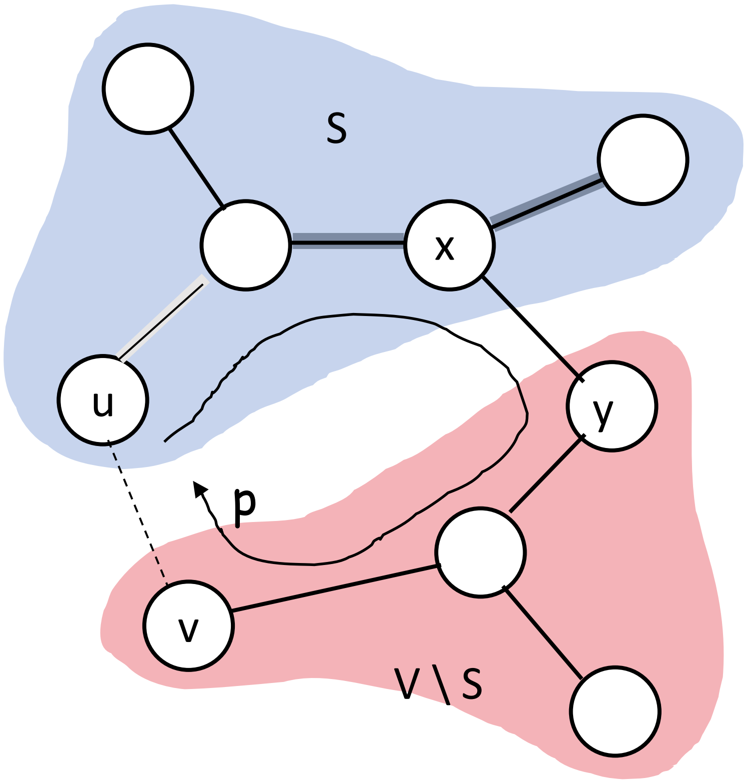
Dokaz teoreme

$T' = T - \{(x, y)\} \cup \{(u, v)\}$

Treba dokazati da je T’ jedan MST:

- (u, v) je laka grana $\Rightarrow w(u, v) \leq w(x, y)$
- $w(T') = w(T) - w(x, y) + w(u, v) \leq w(T)$
- Pošto je T razapinjuće stablo

  $w(T) \leq w(T') \Rightarrow$ T’ mora biti jedan MST

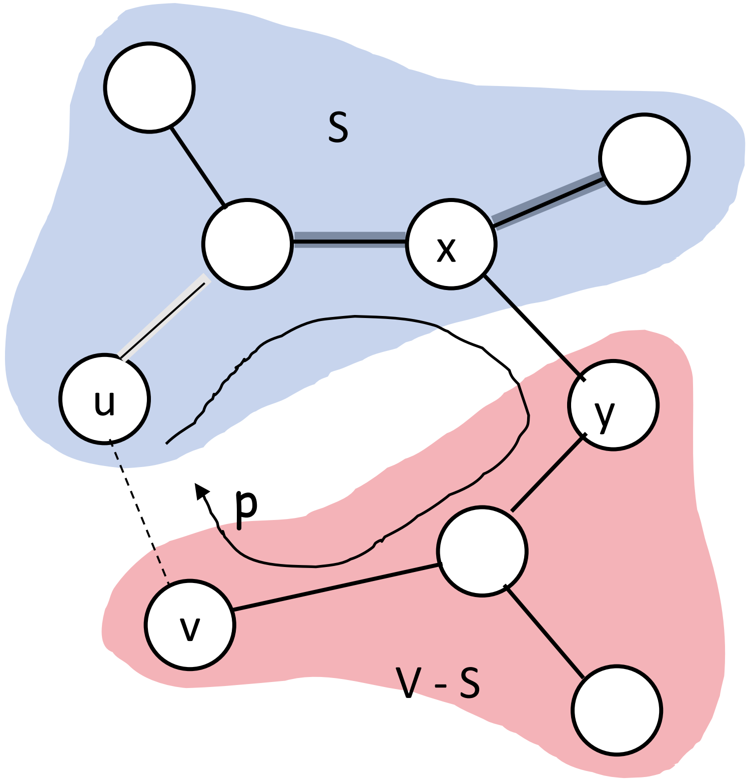


Dokaz teoreme

Treba pokazati da je (u, v) bezbedna za A:

Npr. (u, v) može biti deo nekog MST

- $A \subseteq T$ i $(x, y) \notin T \Rightarrow (x, y) \notin A \Rightarrow A \subseteq T'$
- $A \cup \{(u, v)\} \subseteq T'$
- Pošto je T' MST $\Rightarrow$ (u, v) je bezbedna za A



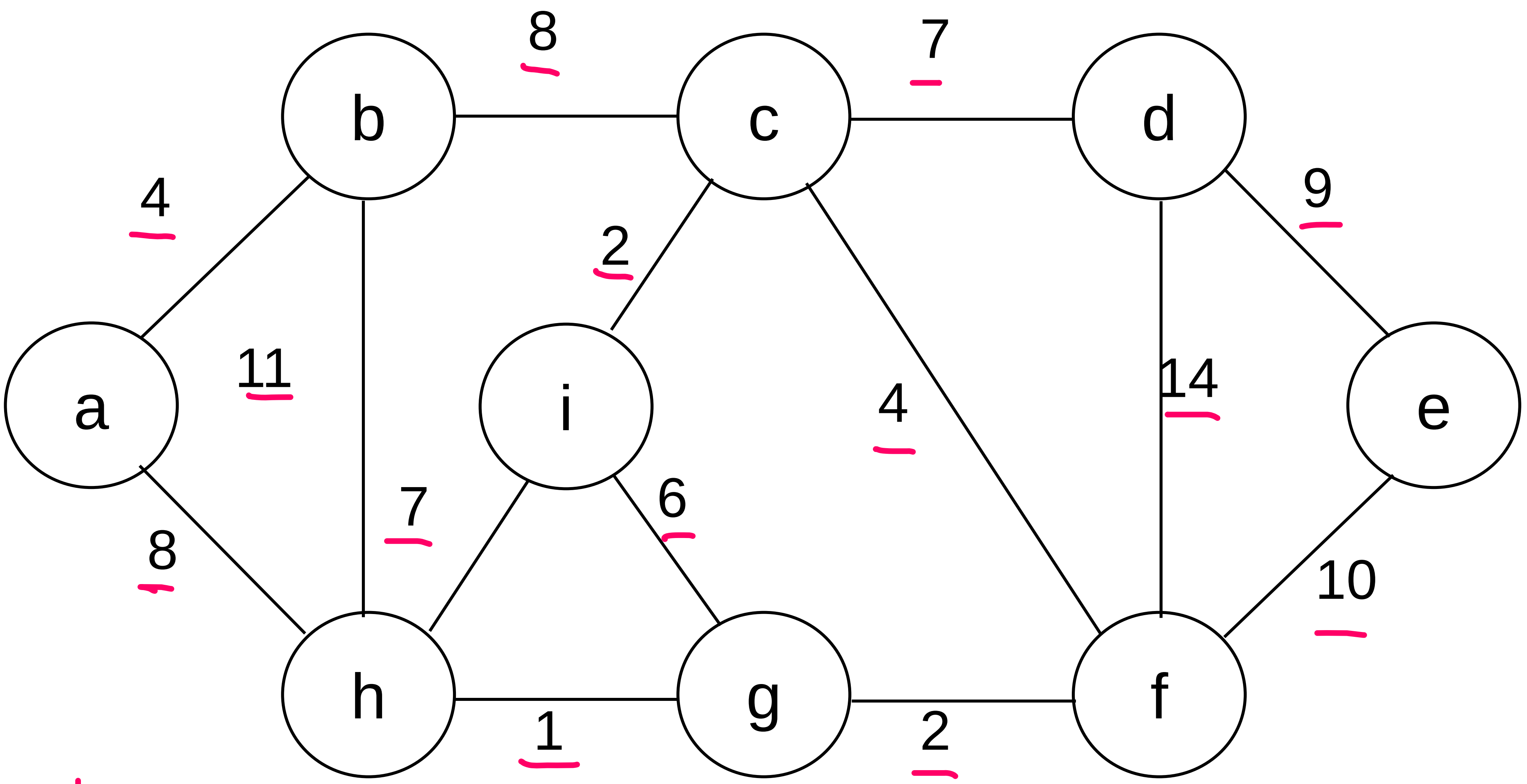
Osnovni algoritmi

- Kruskalov algoritam
  - Polazi od praznog skupa grana i iterativno dodaje najjeftiniju granu koja ne stvara ciklus.
    - Razmatraju se samo grane.

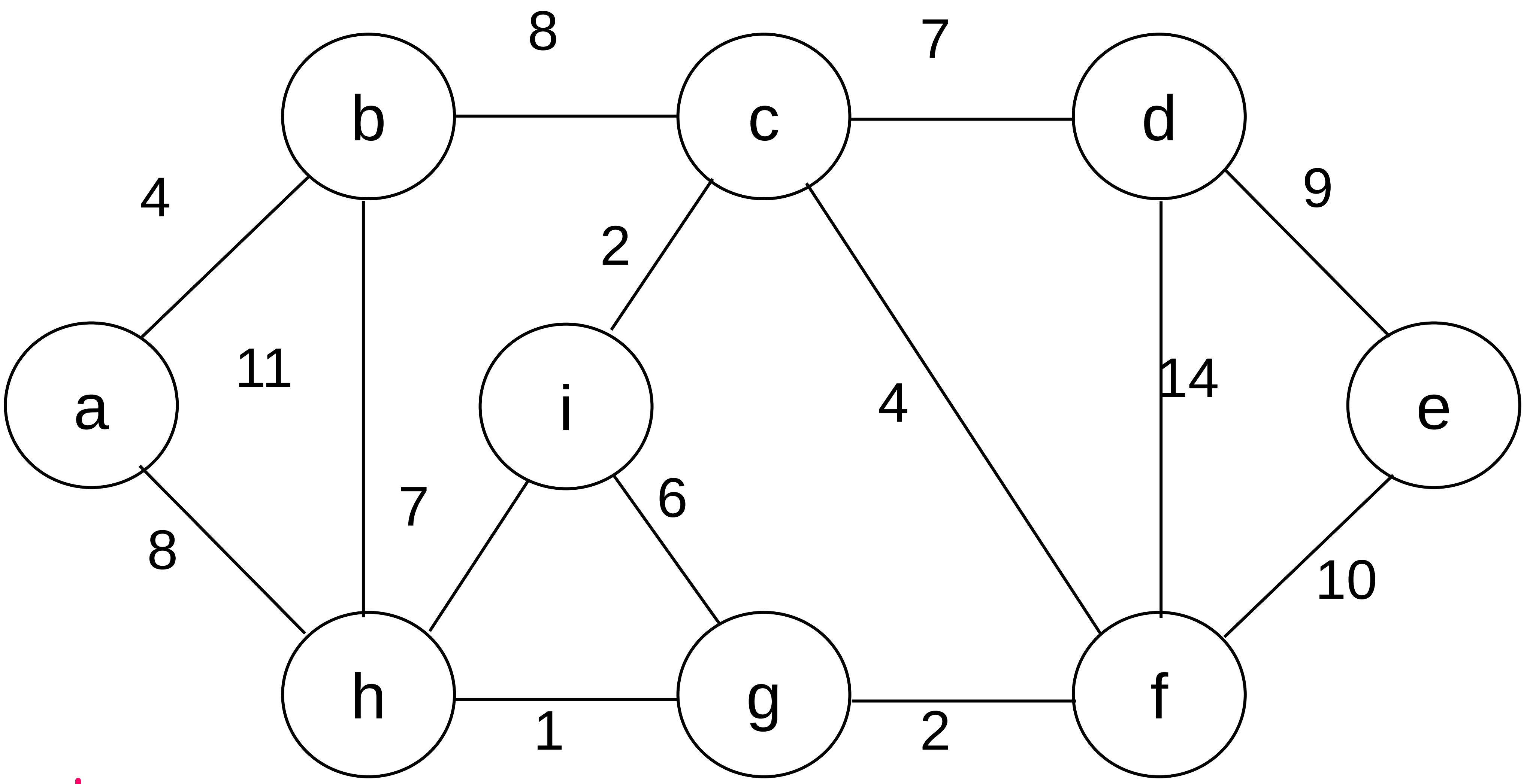
- Primov algoritam
  - Kreće od jednog čvora i iterativno dodaje njemu incidentnu granu i preko nje susedni čvor ako čvor nije dodat u stablo.
    - Razmatraju se i grane i čvorovi.
- Oba algoritma su greedy („pohlepni“) algoritmi.

Kruskalov algoritam

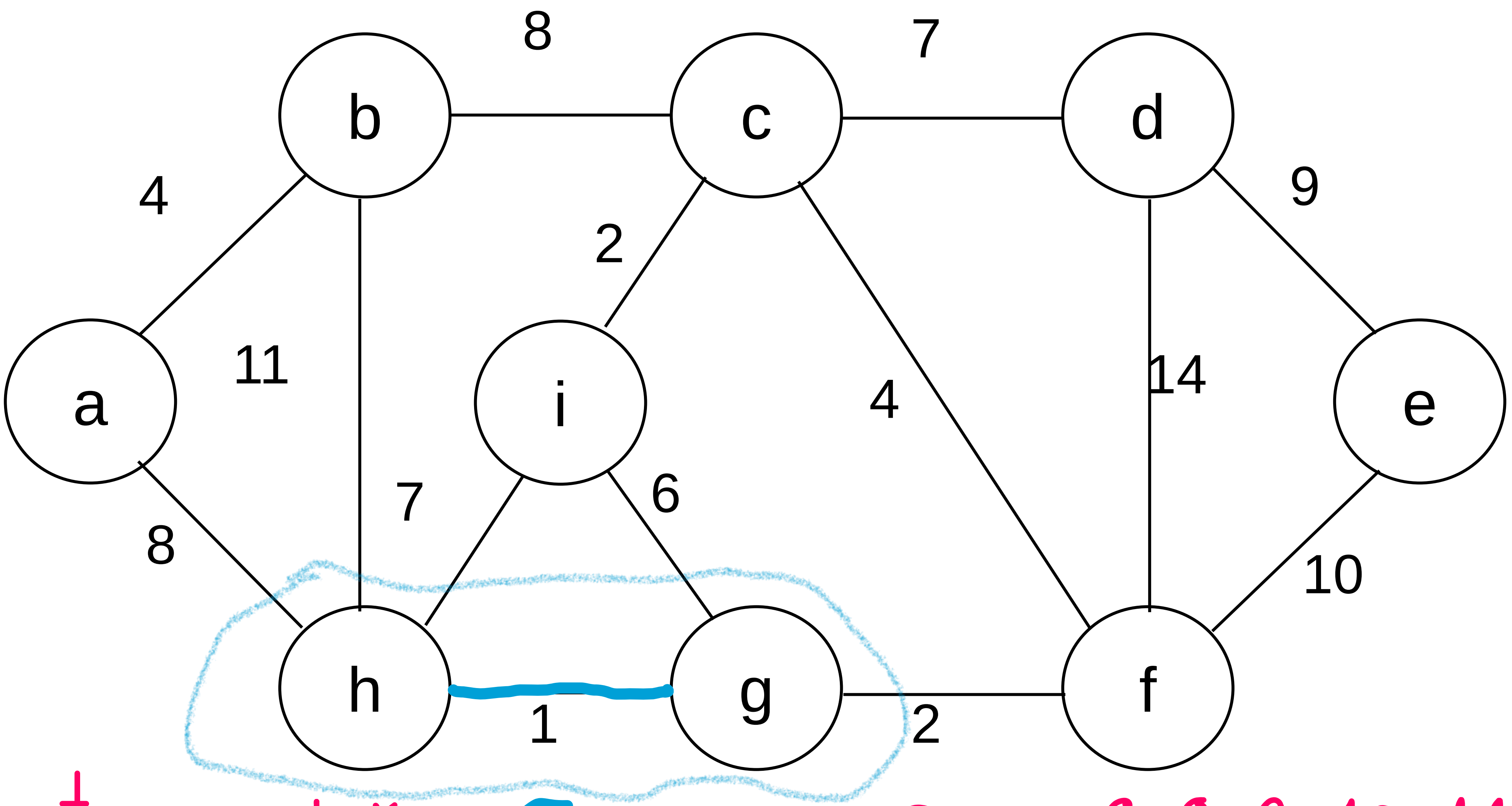
```
MST_KRUSKAL(G, w)
A = ∅
for each v ∈ G.V
  MAKE_SET(v)
S = SORT(G.E, w)     // sortirati grane u neopadajućem red. po w
for each (u,v) ∈ S   // uzimati redom grane iz S
  if FIND_SET(u) ≠ FIND_SET(v)
    A = A U {(u,v)}
    UNION(u,v)
return A
```



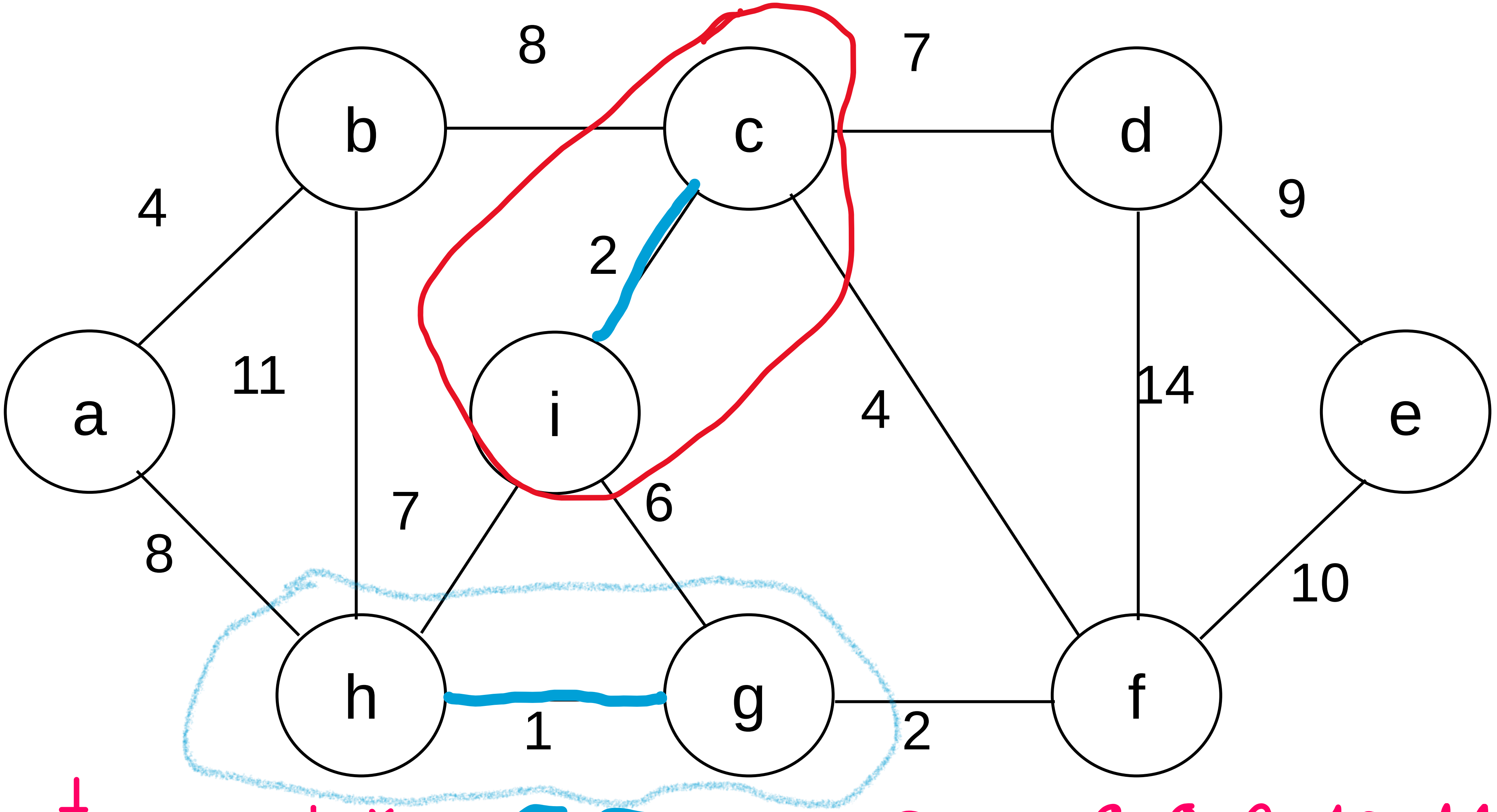
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



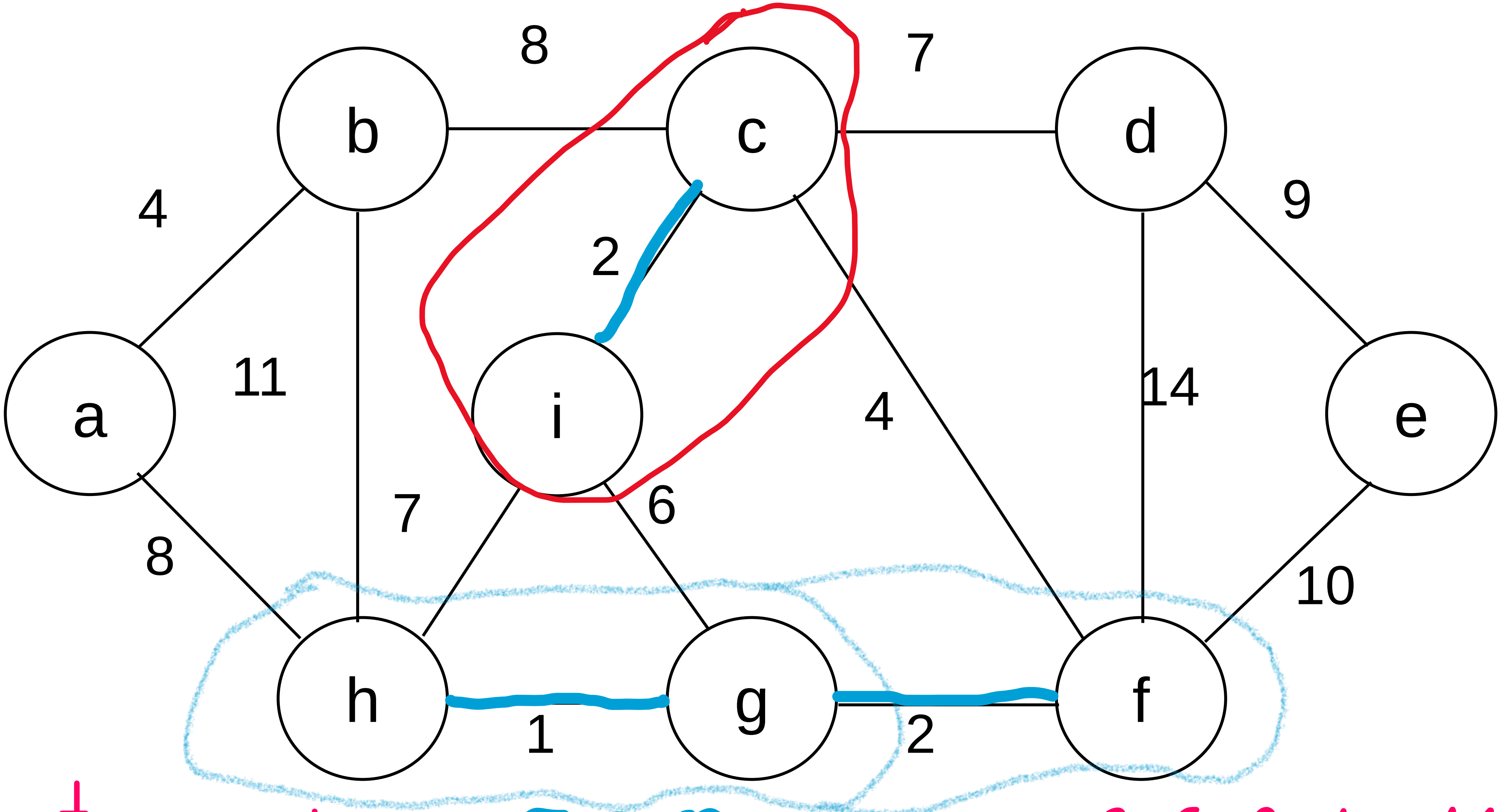
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



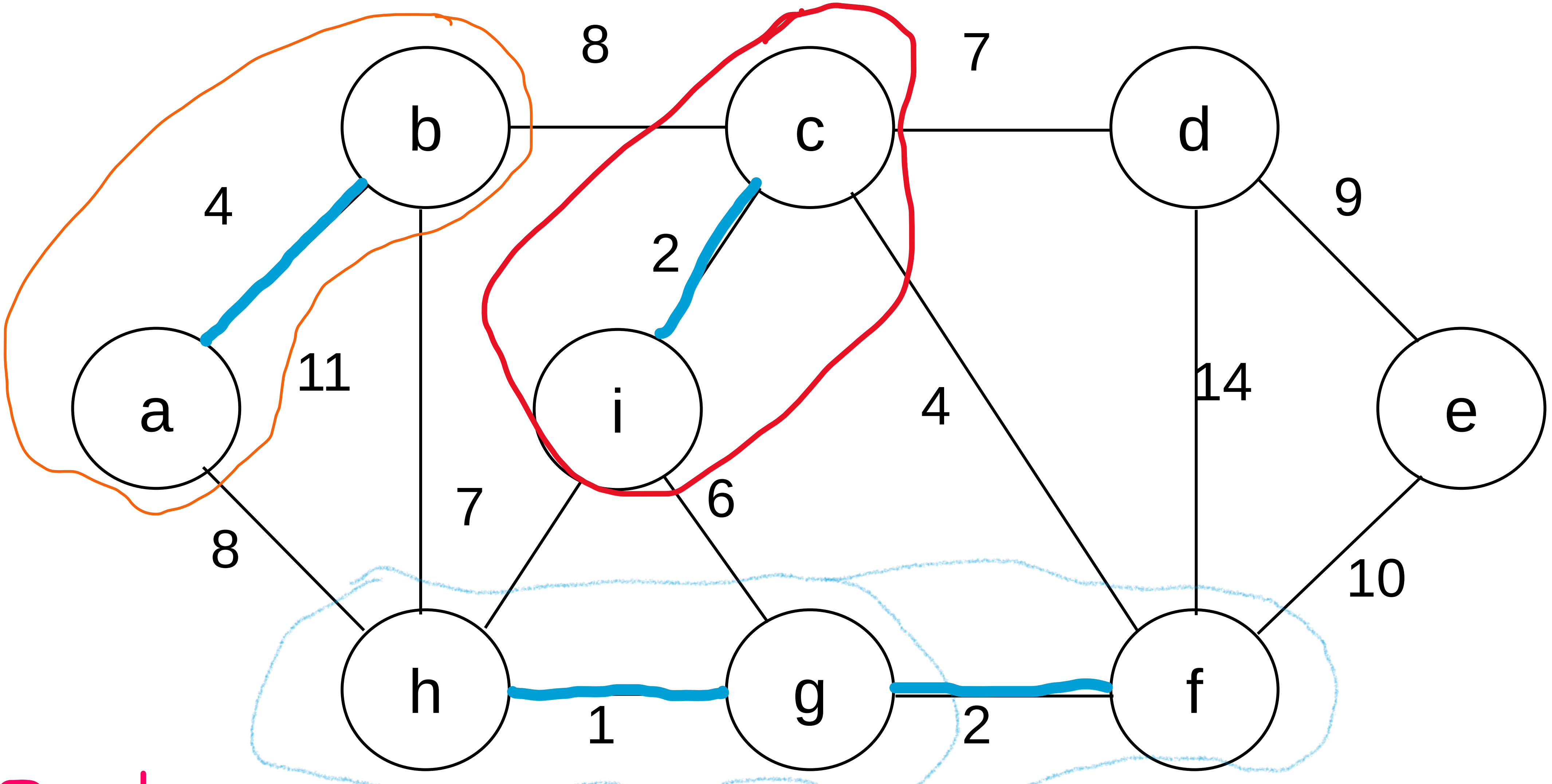
Sortirane težine: 1 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



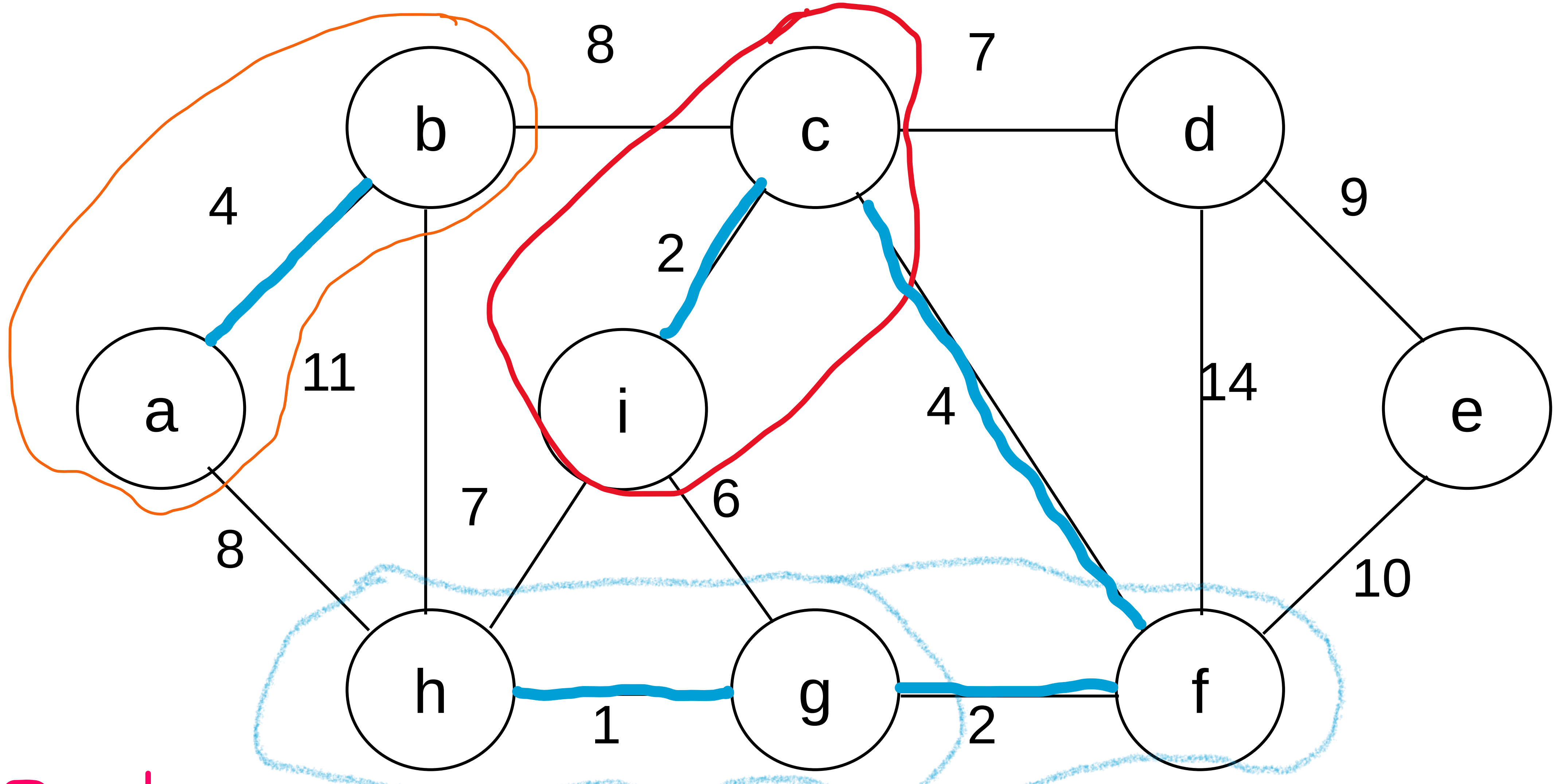
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



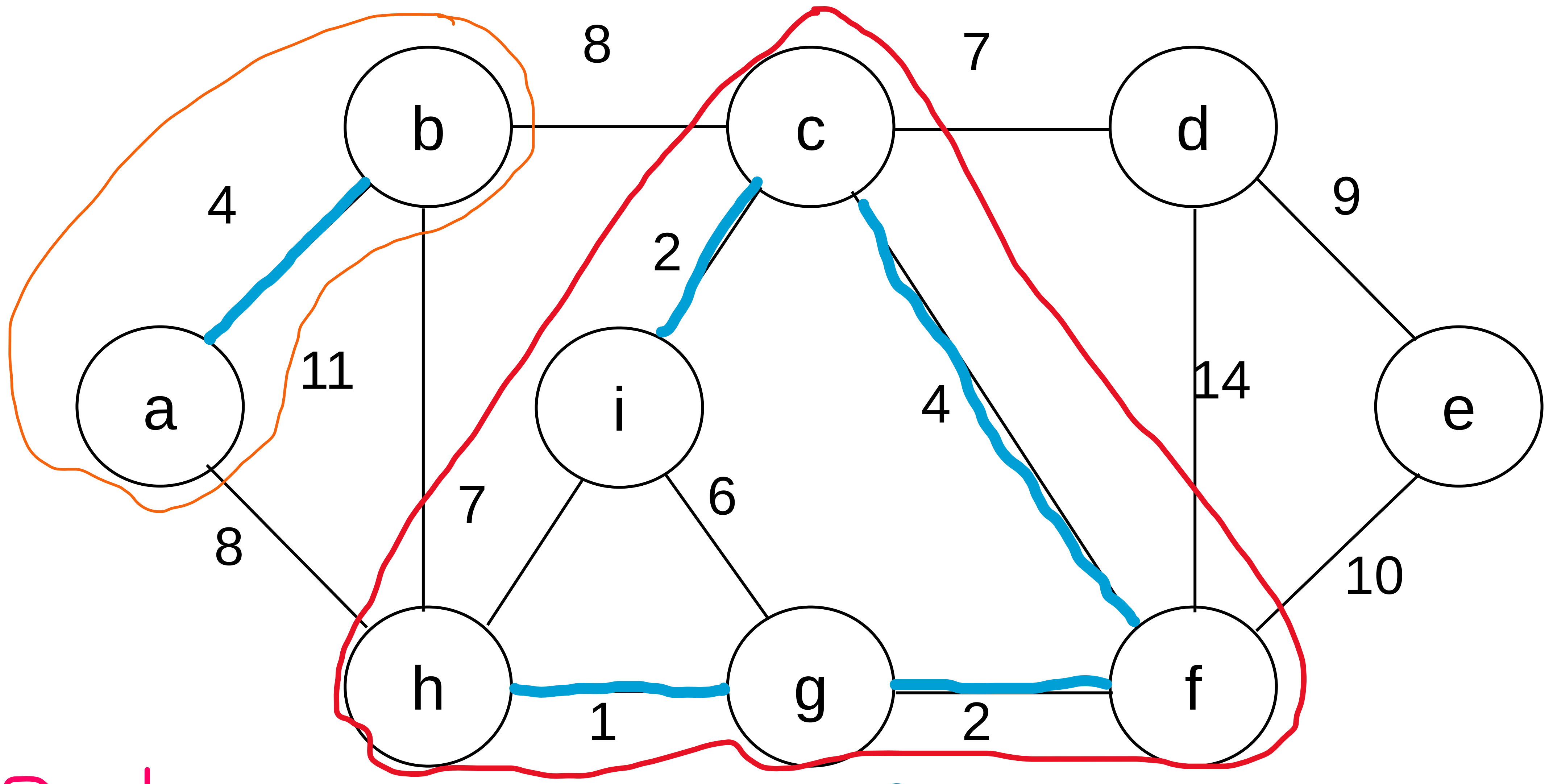
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



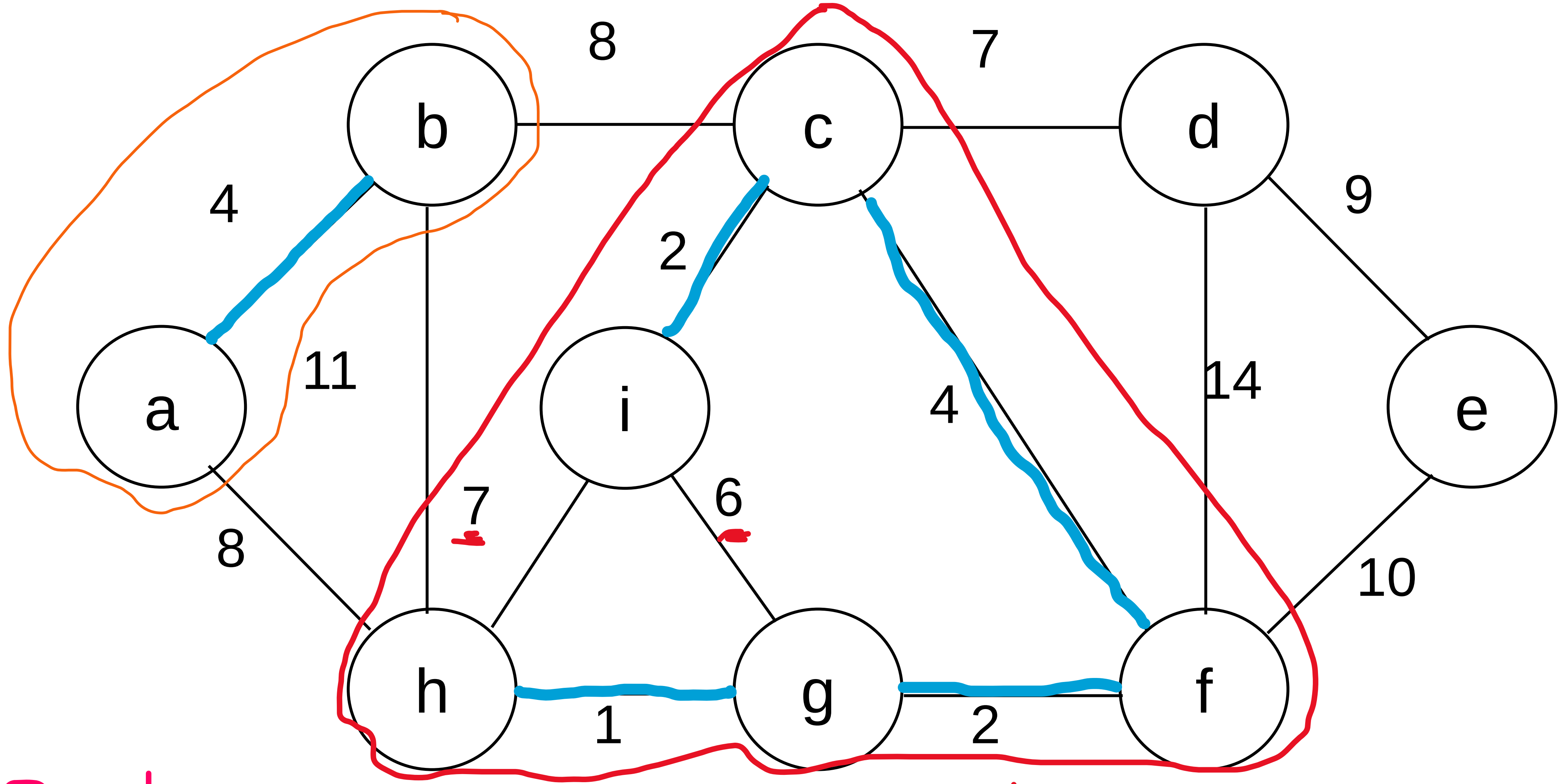
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



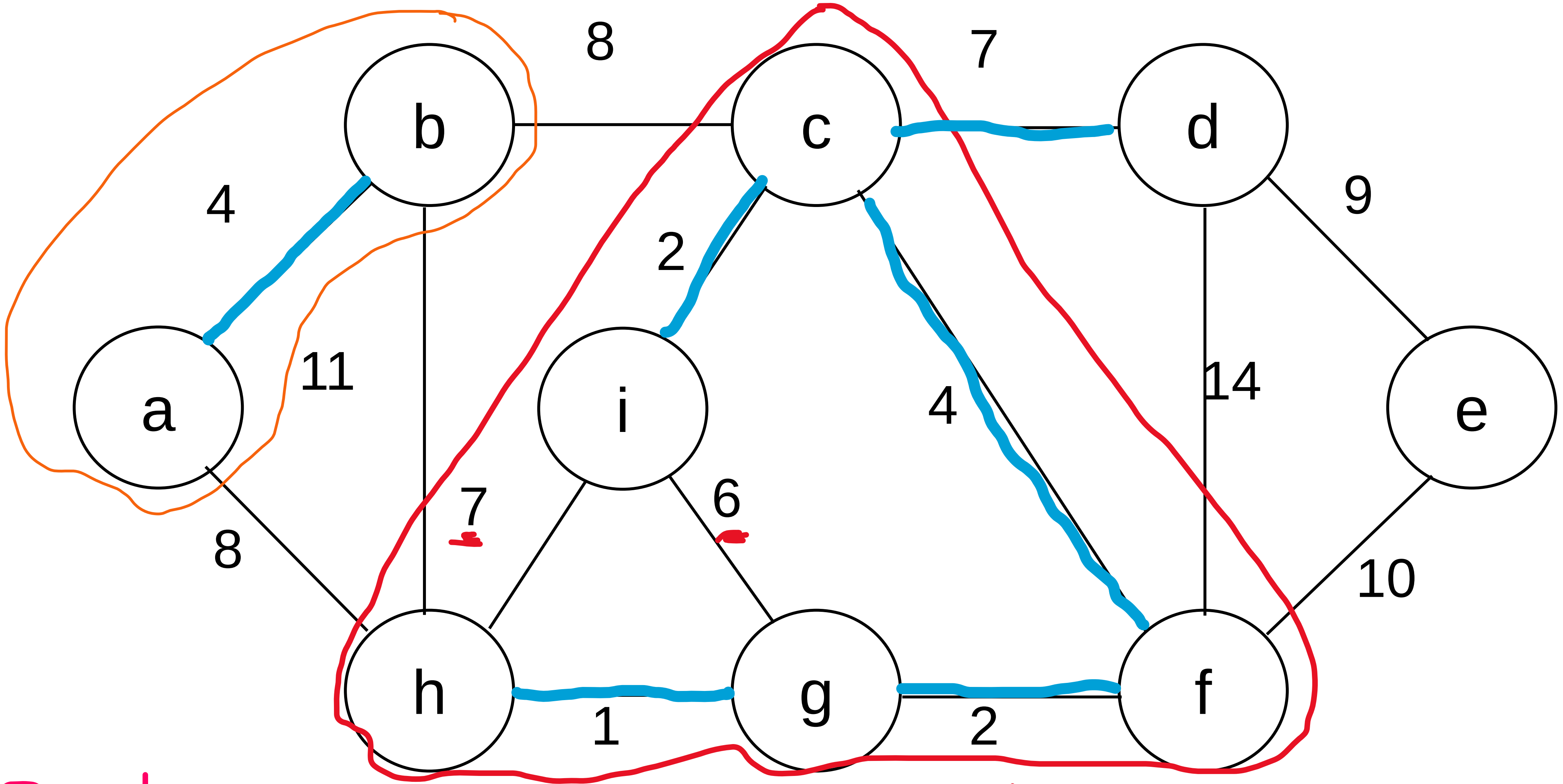
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



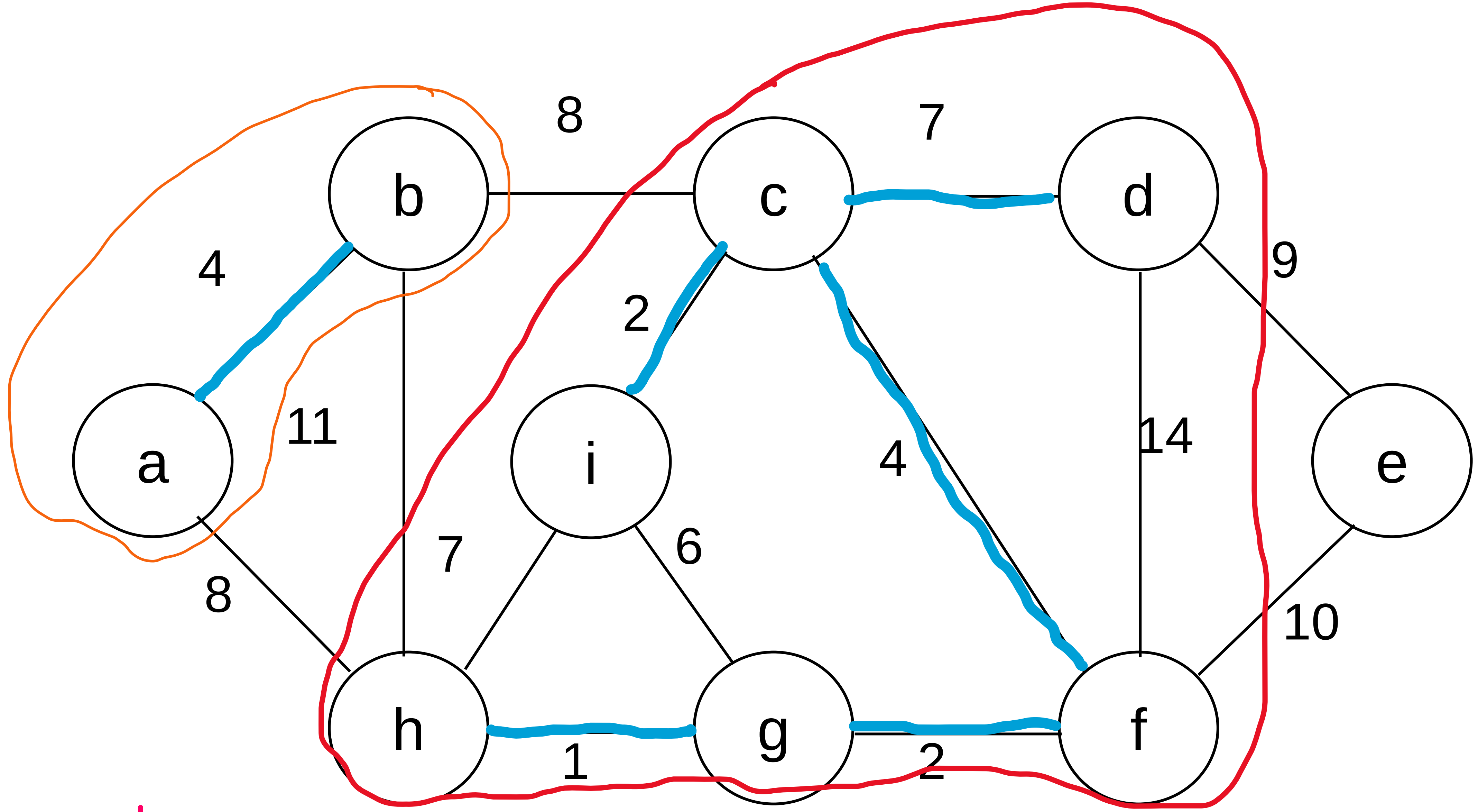
Sortirane težine: 1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 9, 10, 11, 14



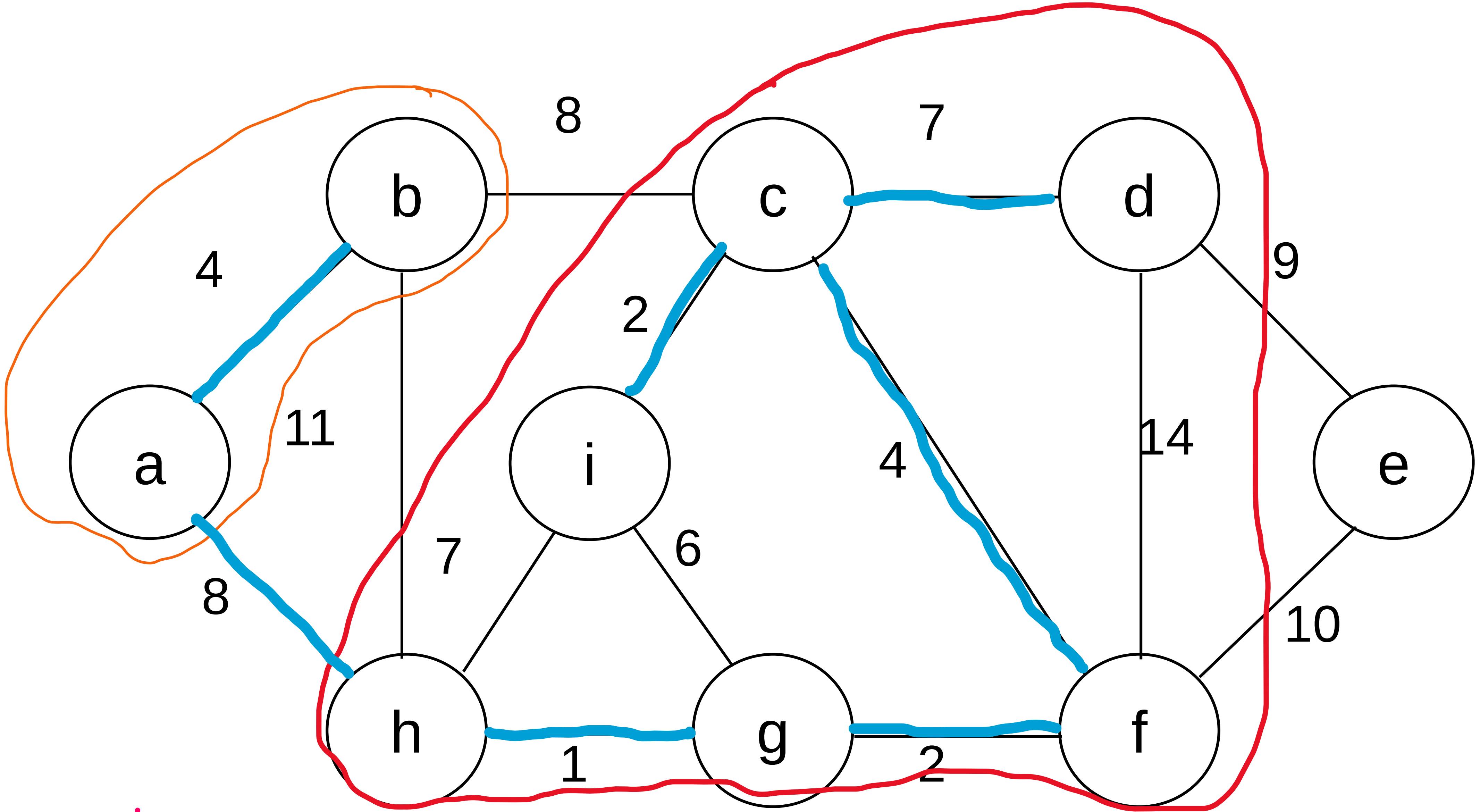
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, 8, 9, 10, 11, 14



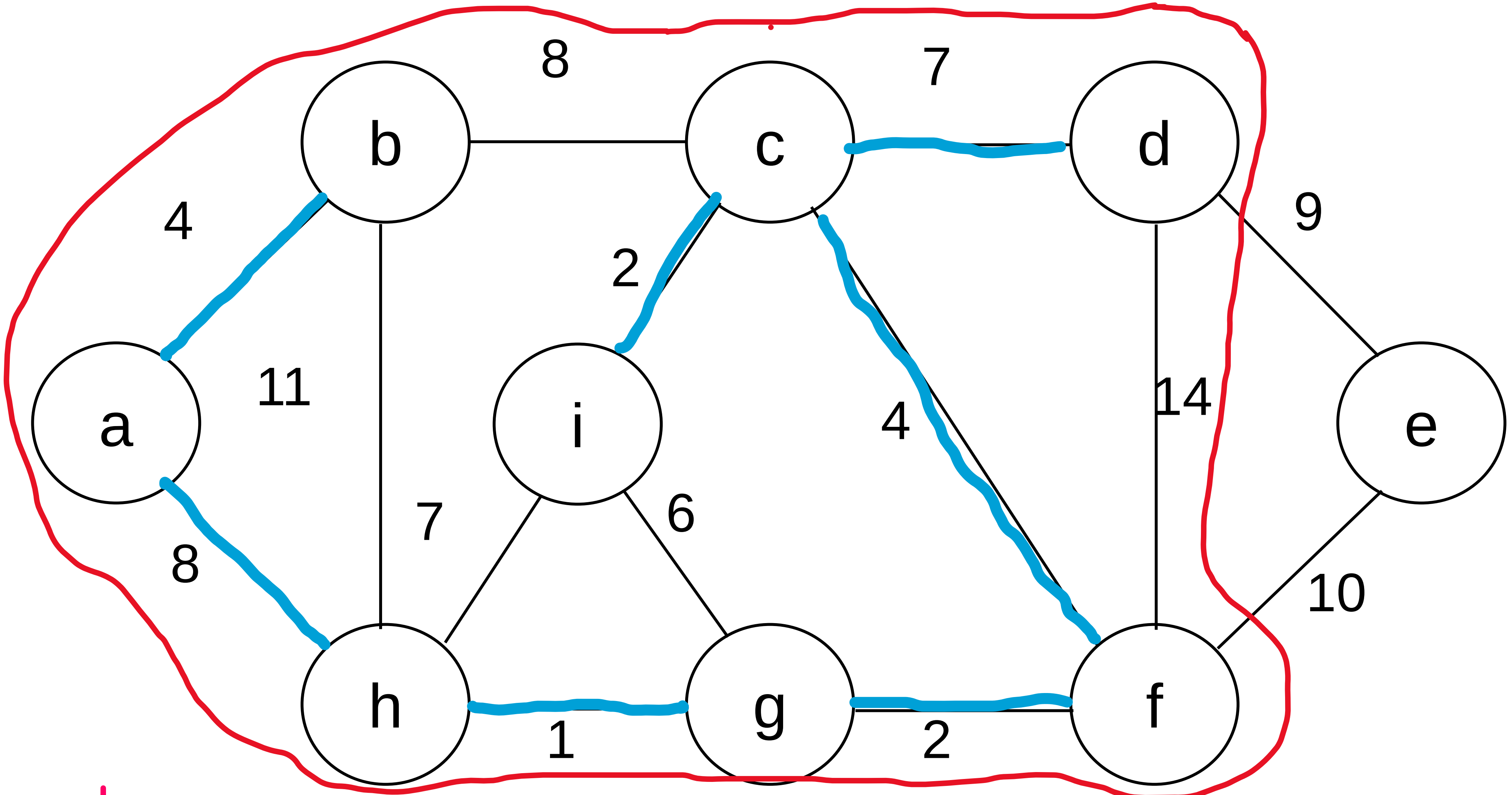
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, 8, 9, 10, 11, 14



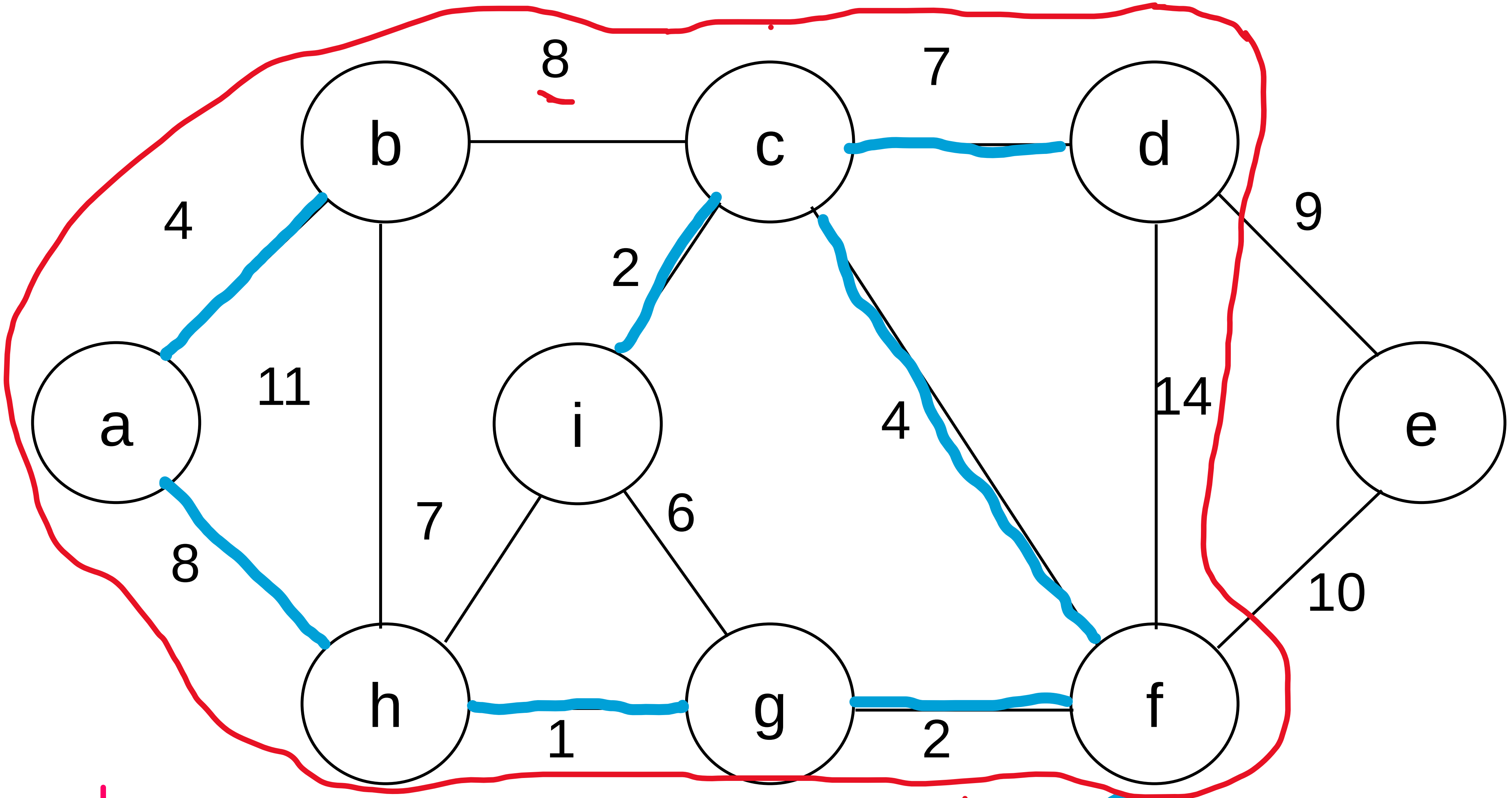
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, 8, 9, 10, 11, 14



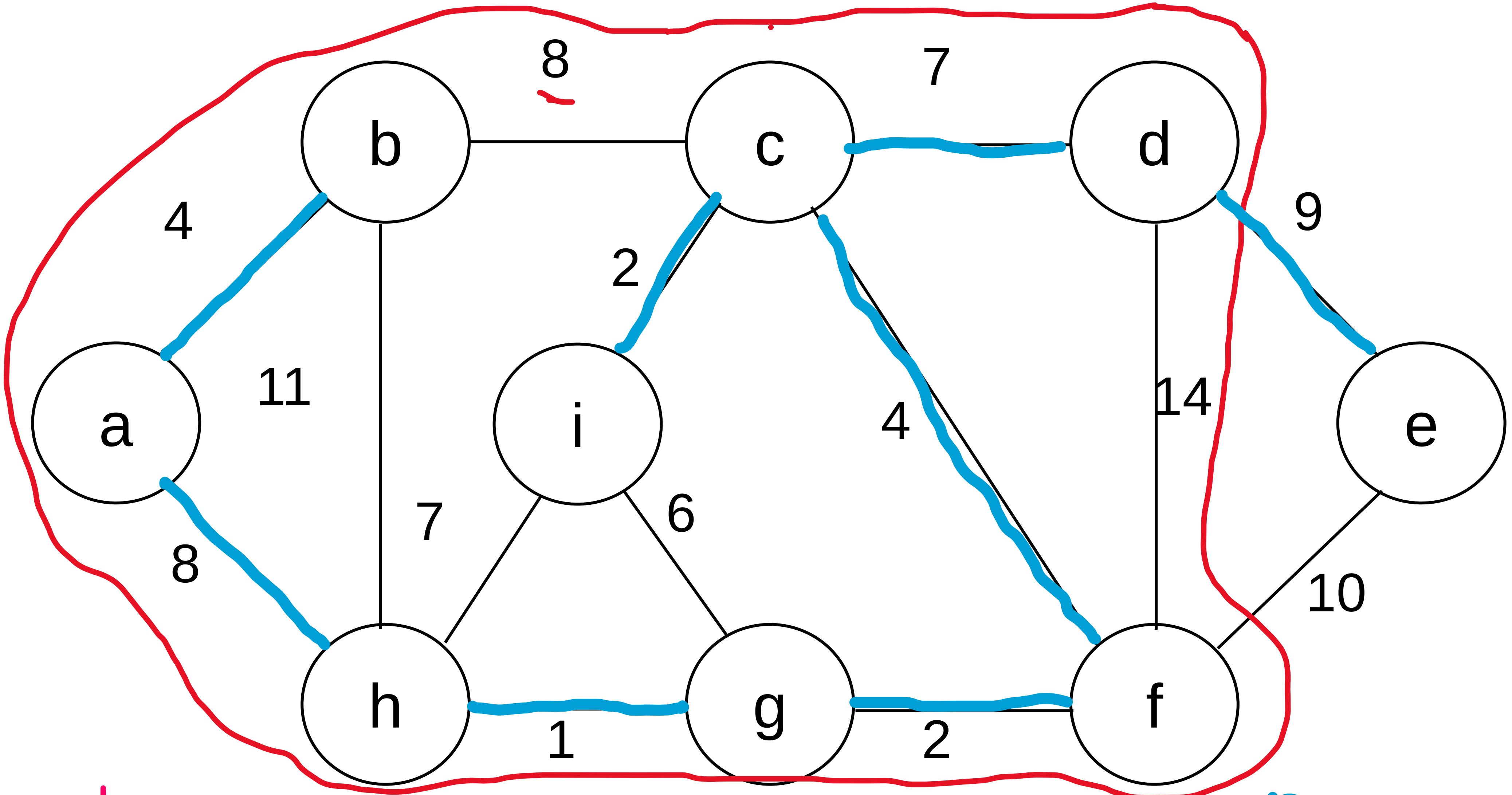
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, 8, 9, 10, 11, 14



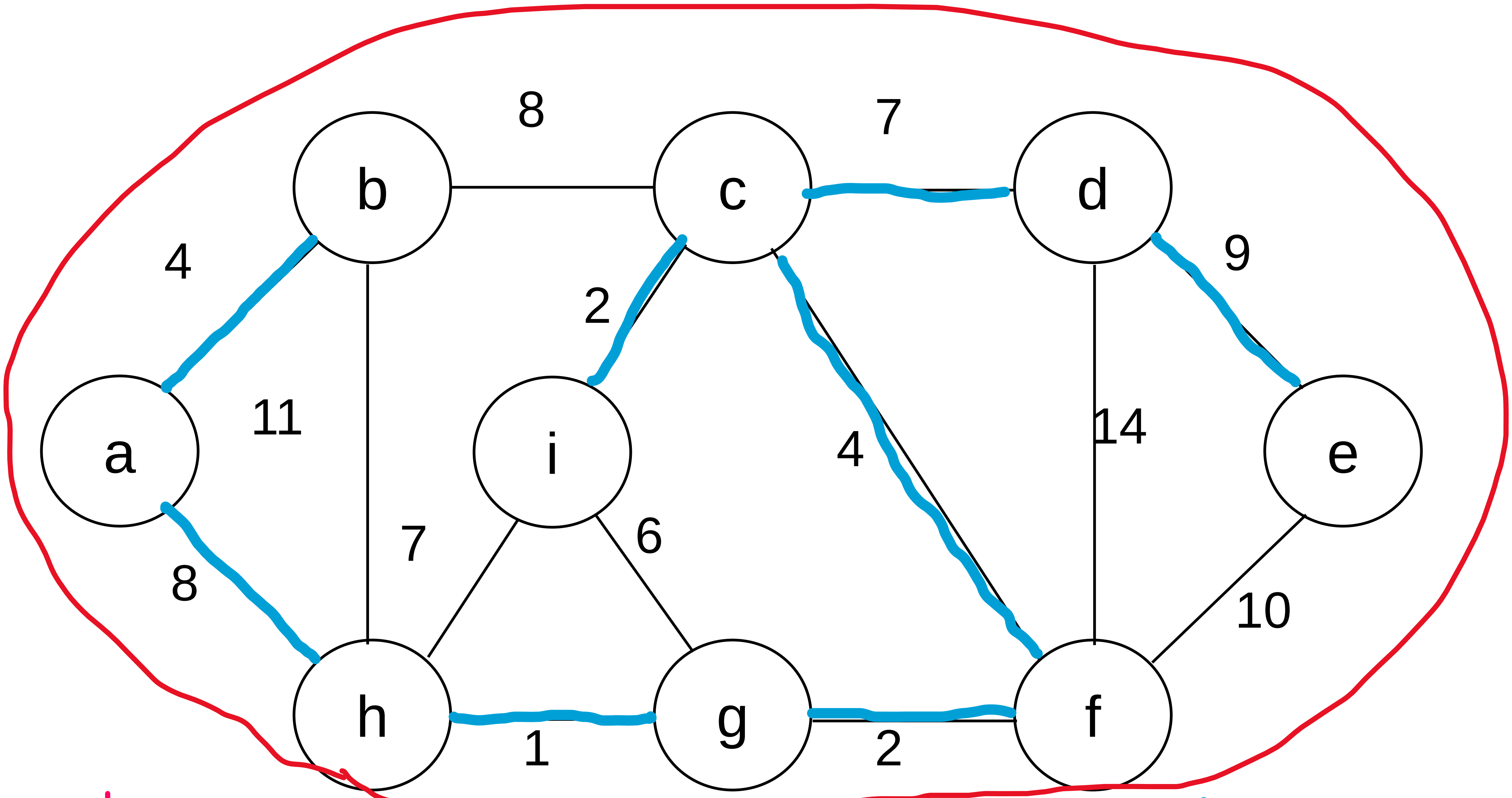
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, 8, 9, 10, 11, 14



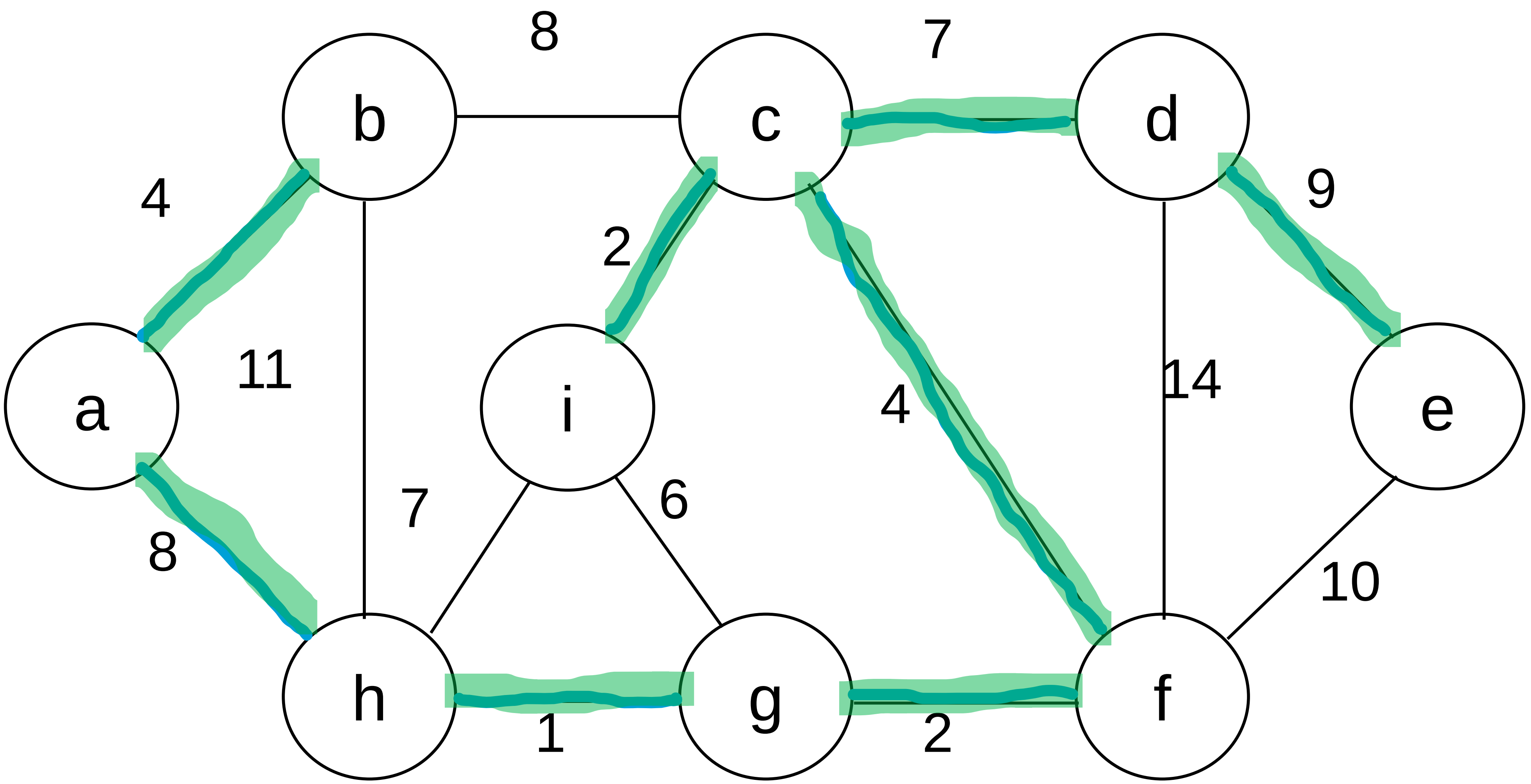
Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, ~~8~~, 9, 10, 11, 14

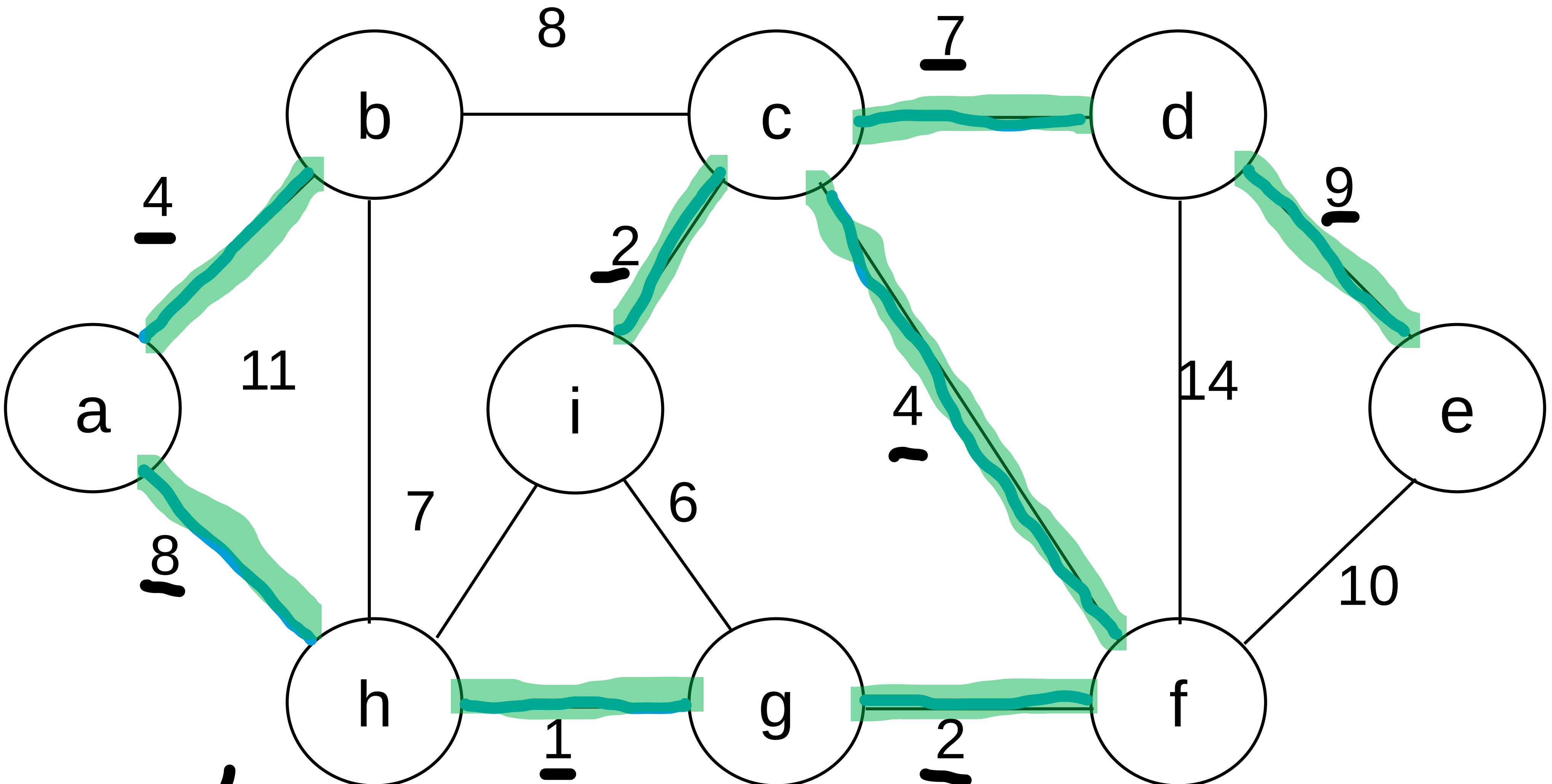


Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, ~~8~~, 9, 10, 11, 14

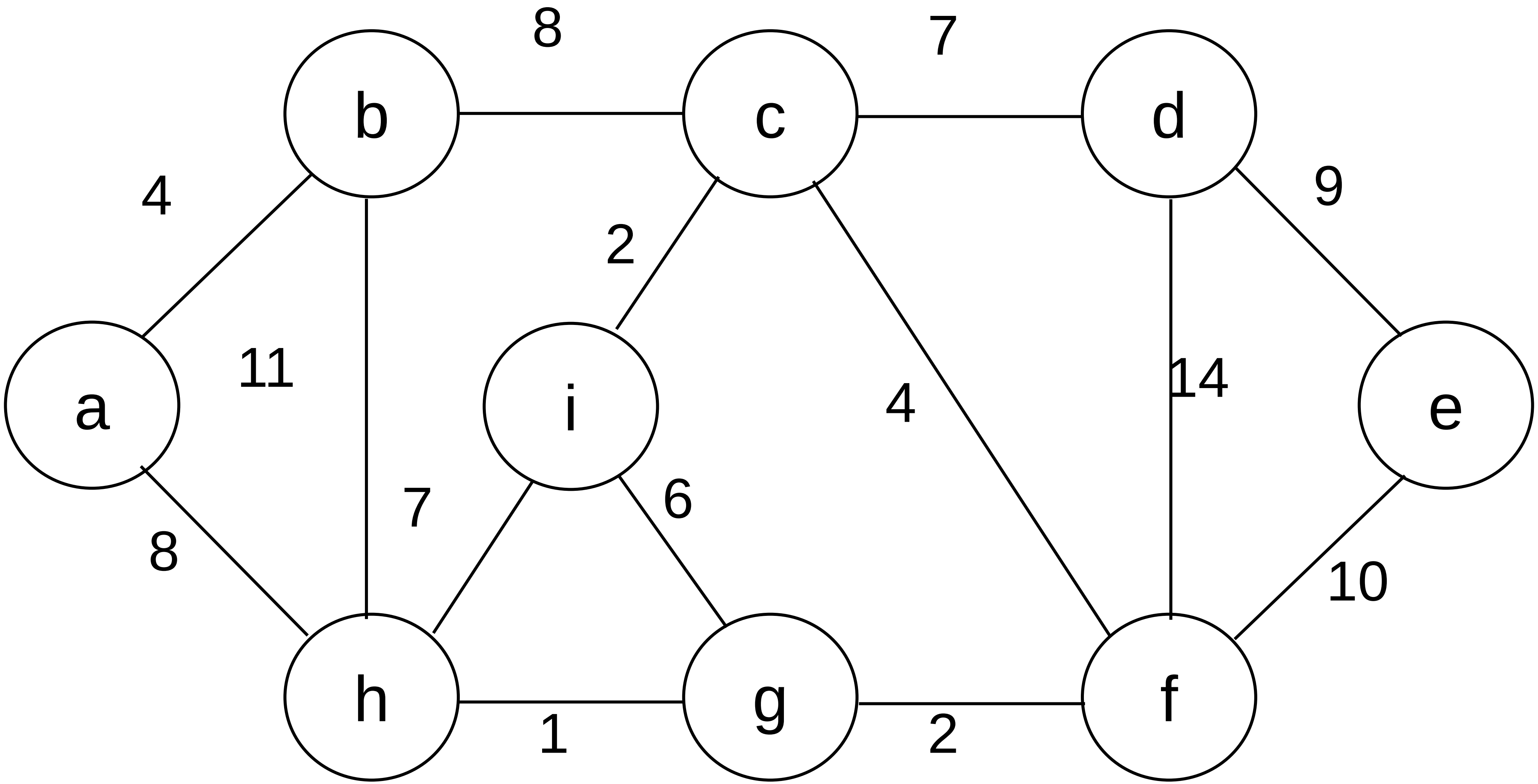


Sortirane težine: 1, 2, 2, 4, 4, ~~6~~, ~~7~~, 7, 8, ~~8~~, 9, 10, 11, 14


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

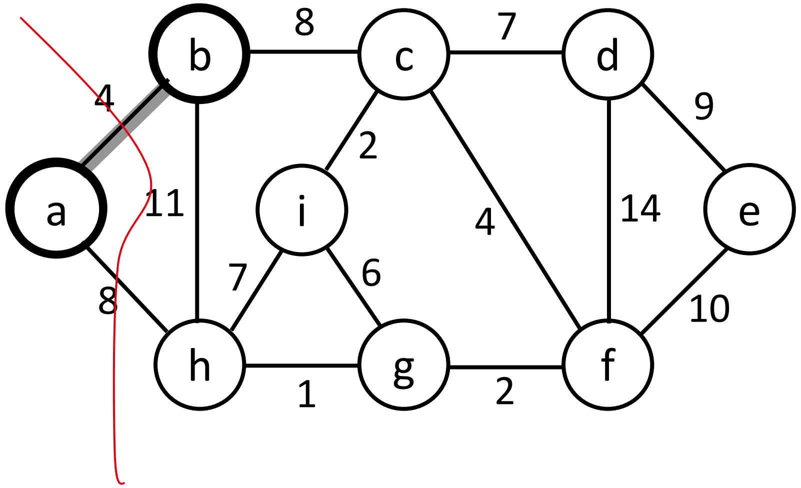


$$\omega(T)=1+2+2+4+4+7+8+9=37$$

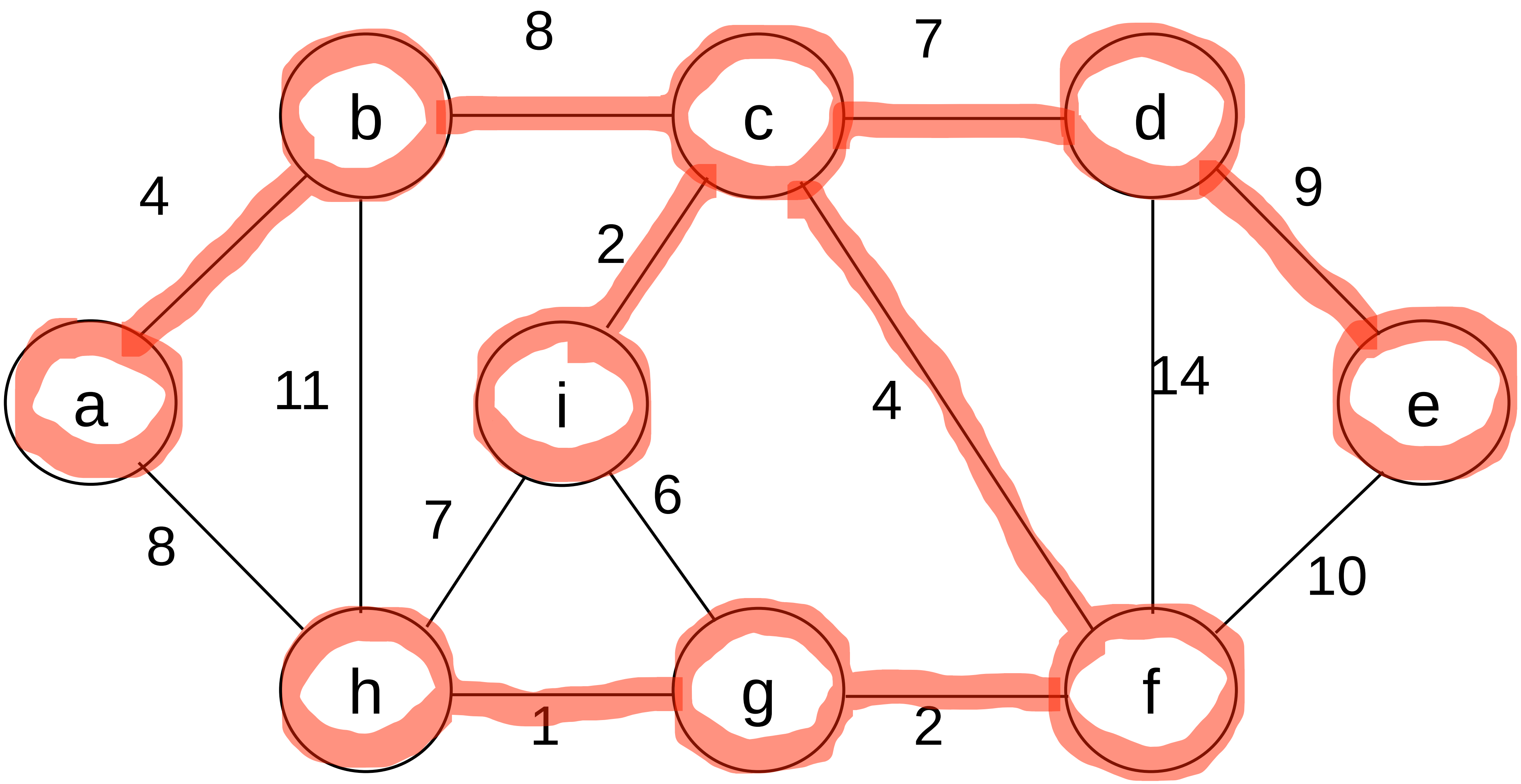

a
b
c
d
e
f
g
h
i
4
8
11
8
7
2
7
6
1
2
4
14
9
10

Primov algoritam

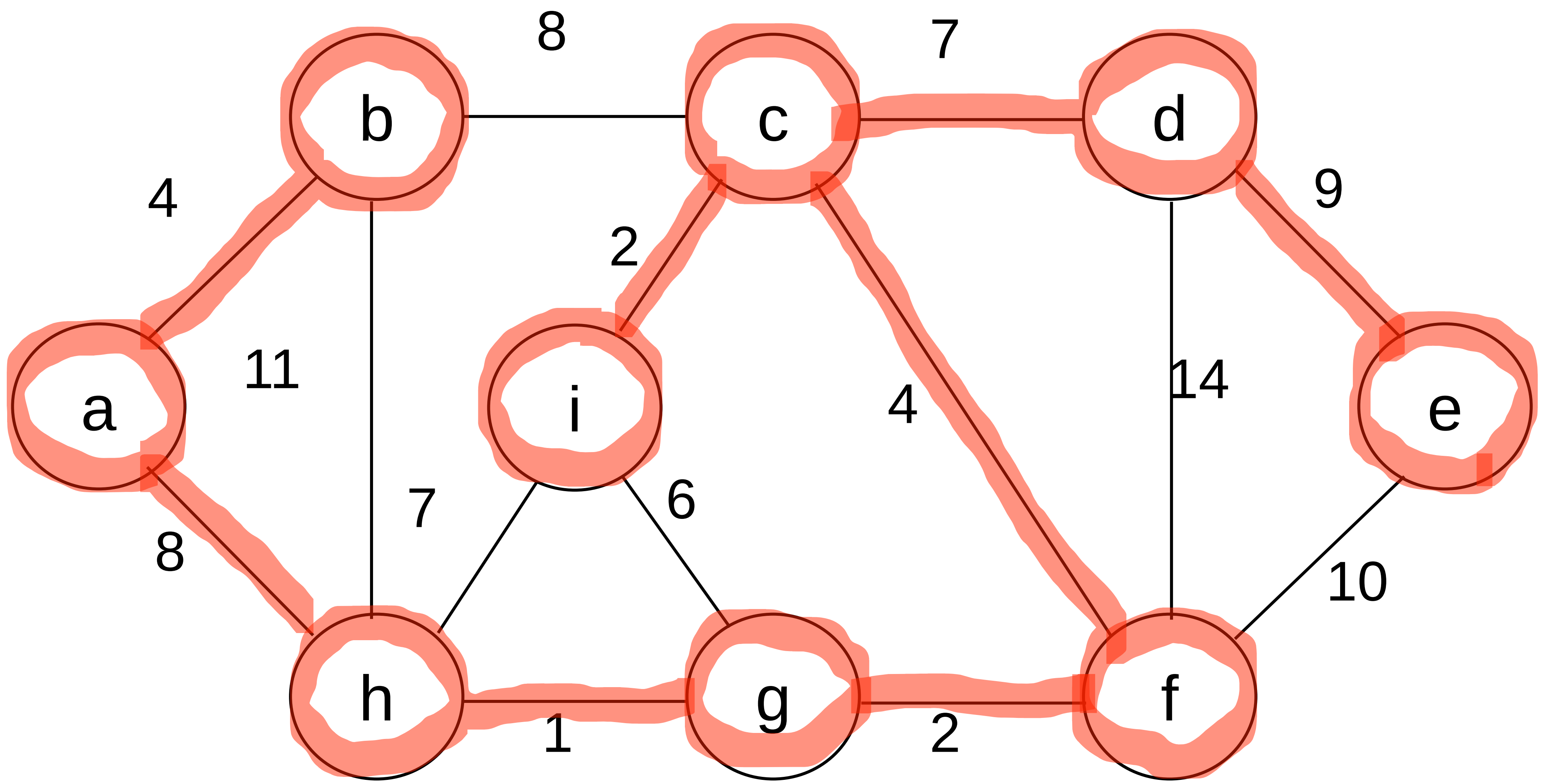
- Grane u skupu A uvek formiraju stablo
- Kreće se od izabranog korena: $V_A = \{a\}$
- U svakom koraku:
  - Pronaći sve lake prelazne grane $(V_A, V \setminus V_A)$
  - Dodati ove grane u A
  - Ponavljati dok stablo ne obuhvati sve čvorove



```
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)
```

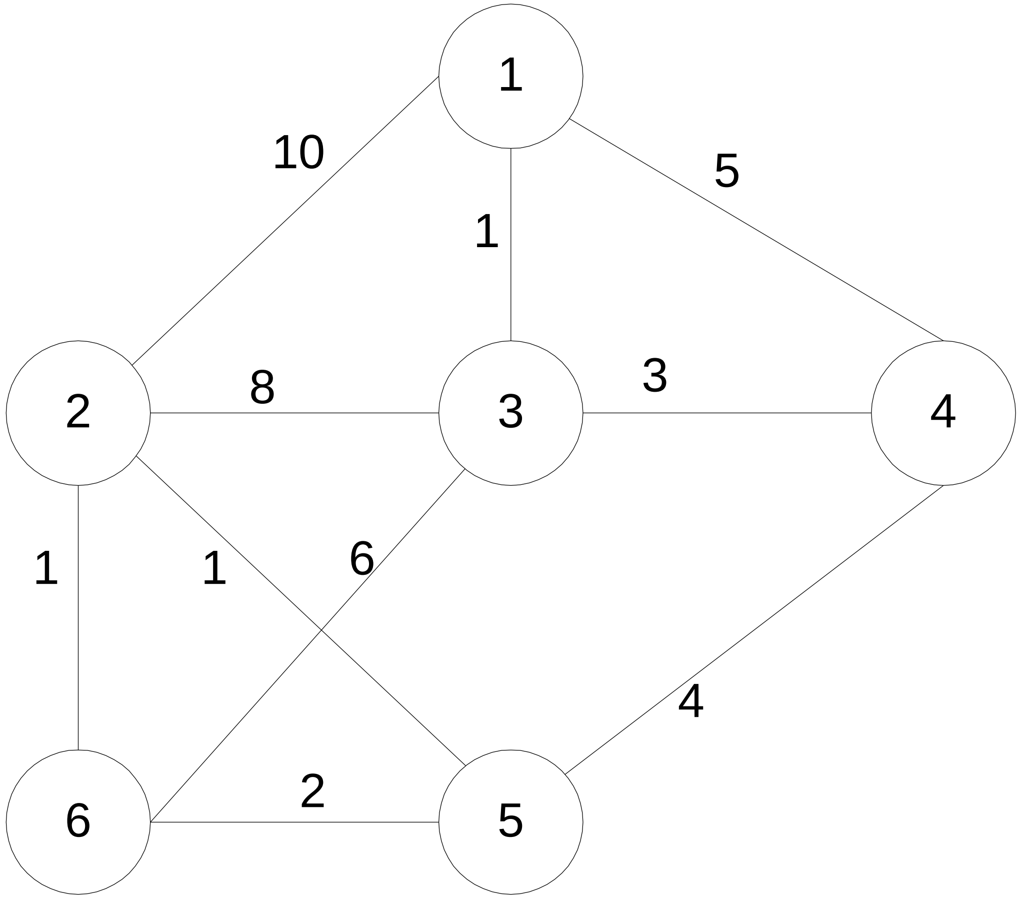


$W(T) = 4 + 8 + 2 + 7 + 9 + 4 + 2 + 1 = 35$

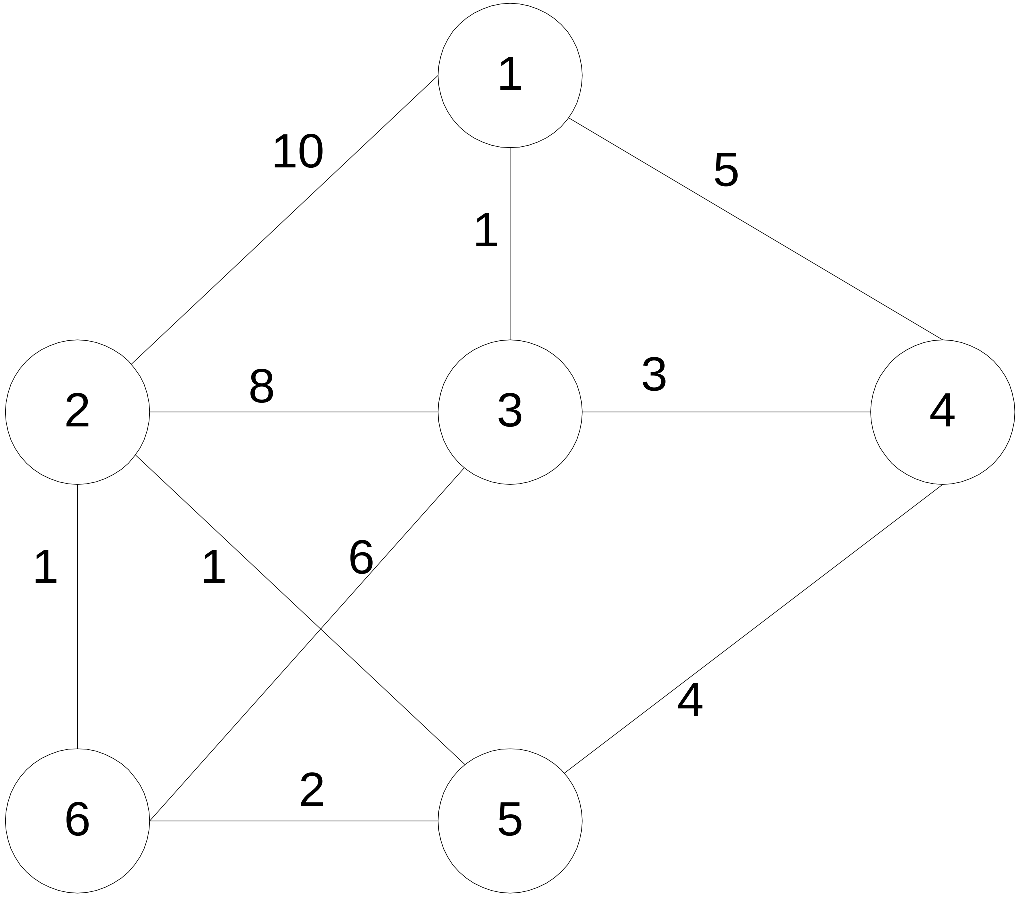


a
b
c
d
e
f
g
h
i
4
8
7
9
11
2
4
14
7
6
10
8
1
2

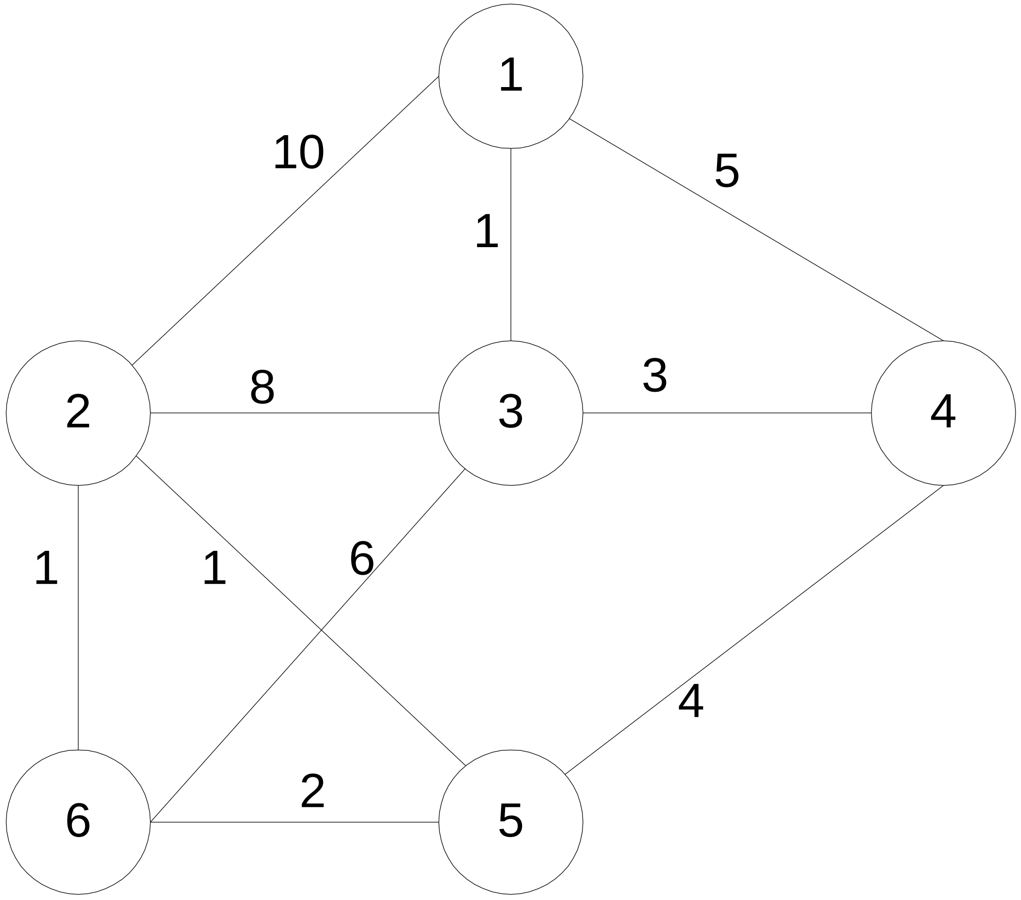
Primer



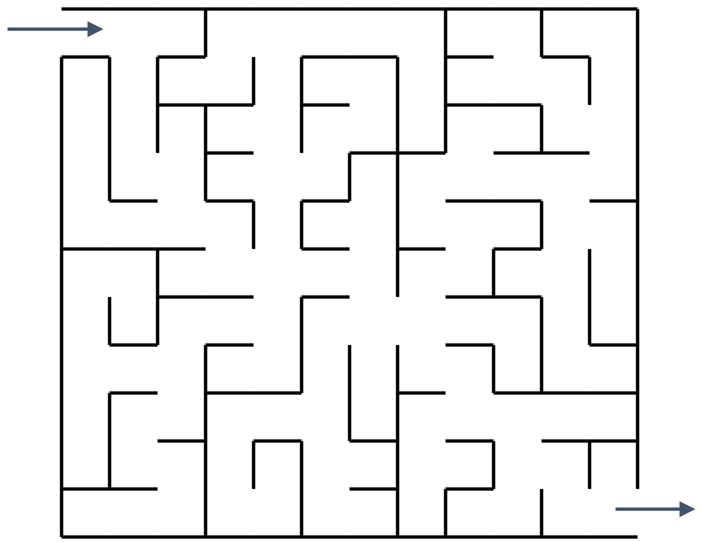
Primer 1



Primer 1

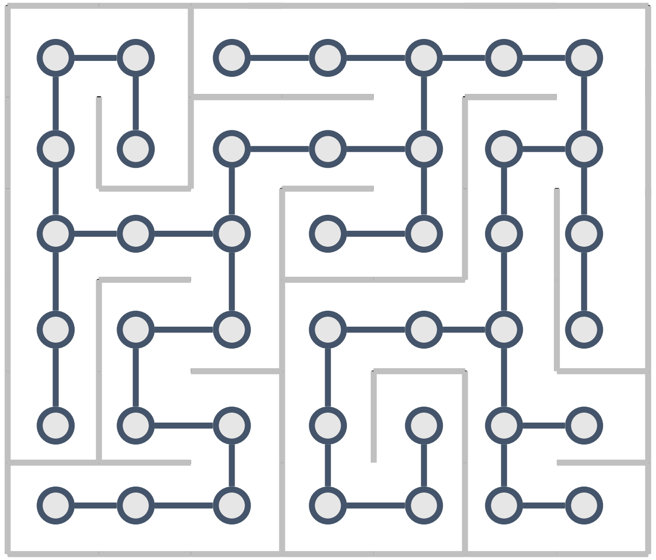


Lavirint



- Za lavirint obično važi,
  - Do svake pozicije se može doći iz startne pozicije
    - U suprotnom „nedostupni deo“ izbacujemo iz razmatranja – smanjujemo problem
  - Postoji samo jedna putanja od starta do cilja
- Ako su ćelije *čvorovi*, a prolaz između ćelija *grane,*
- predstavlja se razapinjućim stablom
- Pošto postoji tačno jedan put između bilo kojeg para ćelija, bilo koje ćelije mogu biti „start“ i „cilj“

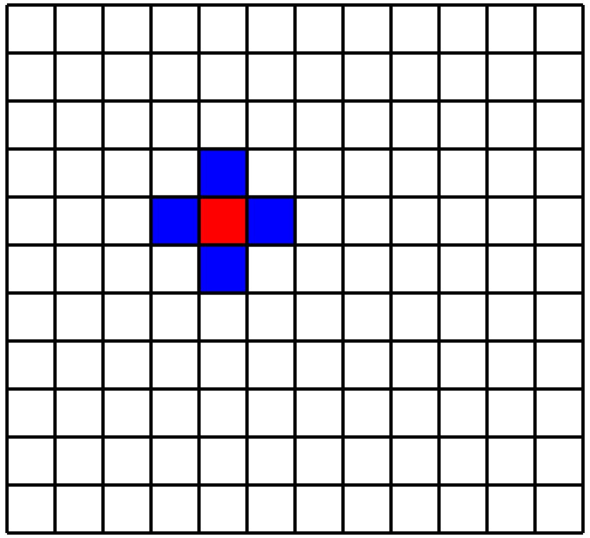
Lavirint kao stablo



- Većina lavirinata se može predstaviti razapinjućim stablom
- Za čvorovi grafa se biraju pozicije lavirinta
- Postoji tačno jedna putanja između dva čvora

Izgradnja lavirinta I

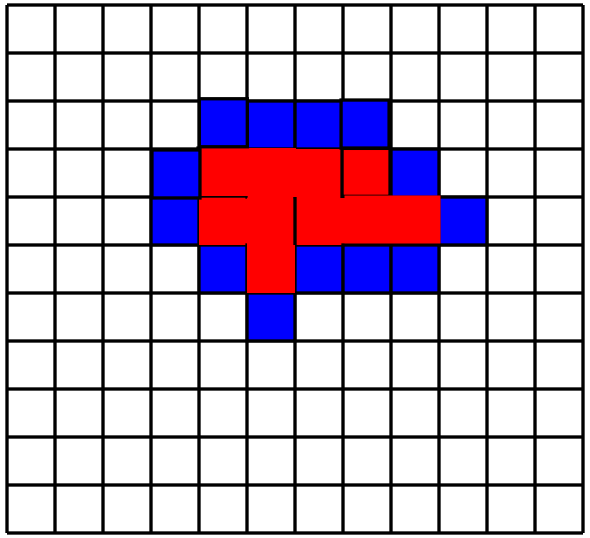
- Ovaj algoritam zahteva dva skupa ćelija
  - IN - skup ćelija koje su u razapinjućem stablu
  - FRONTIER - skup ćelija koje su susedi stabla (a nisu u stablu)
- Inicijalno su sve ćelije zidovi



- Izabere se bilo koja ćelija i stavlja u IN
- Sve susedne ćelije koje nisu u IN, staviti u FRONTIER

Izgradnja lavirinta II

- Ponavljati korake:
  - Prebaciti bilo koju ćeliju C iz FRONTIER i ubaciti u IN
  - Izbrisati zid između C i neke susedne ćelije iz IN
  - Dodati u FRONTIER sve susedne ćelije od C koje nisu ni u IN ni u FRONTIER



- Nastaviti dok se ne isprazni FRONTIER

- Kada je Lavirint kompletan (ili u bilo kom momentu) izabrati početnu i ciljnu ćeliju