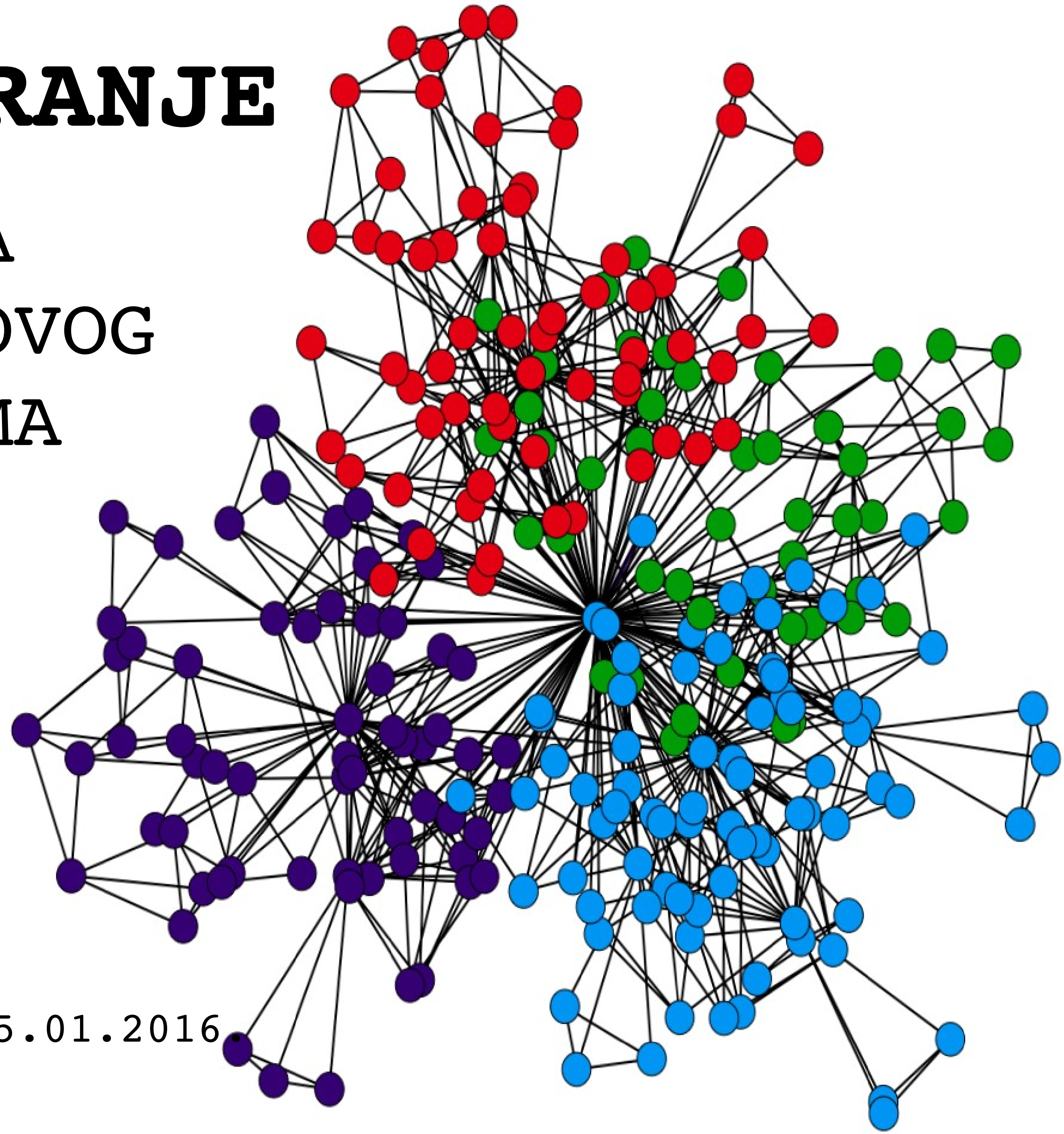
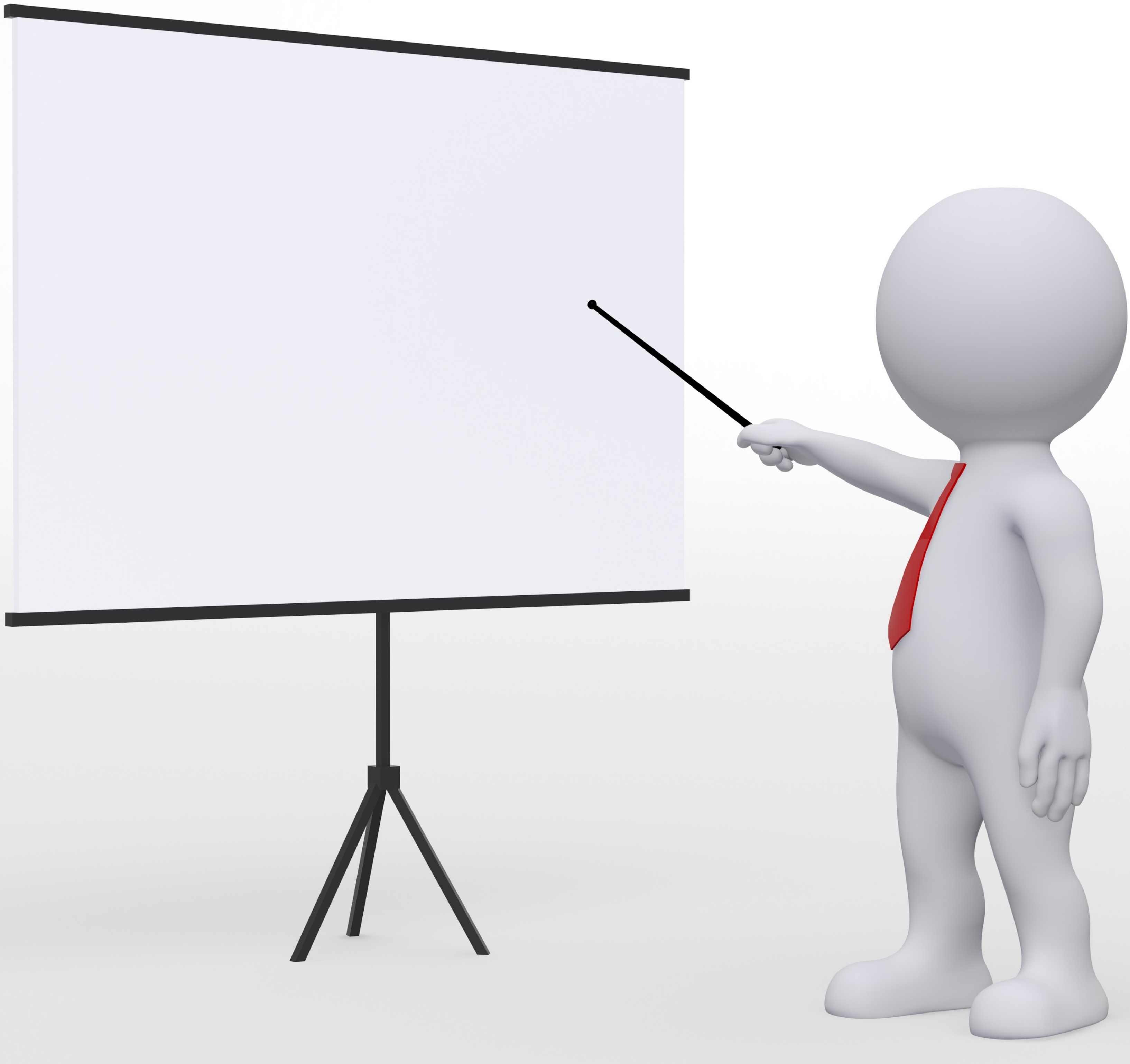
KLASTERIRANJE

PRIMJENA KRUSKALOVOG ALGORITMA

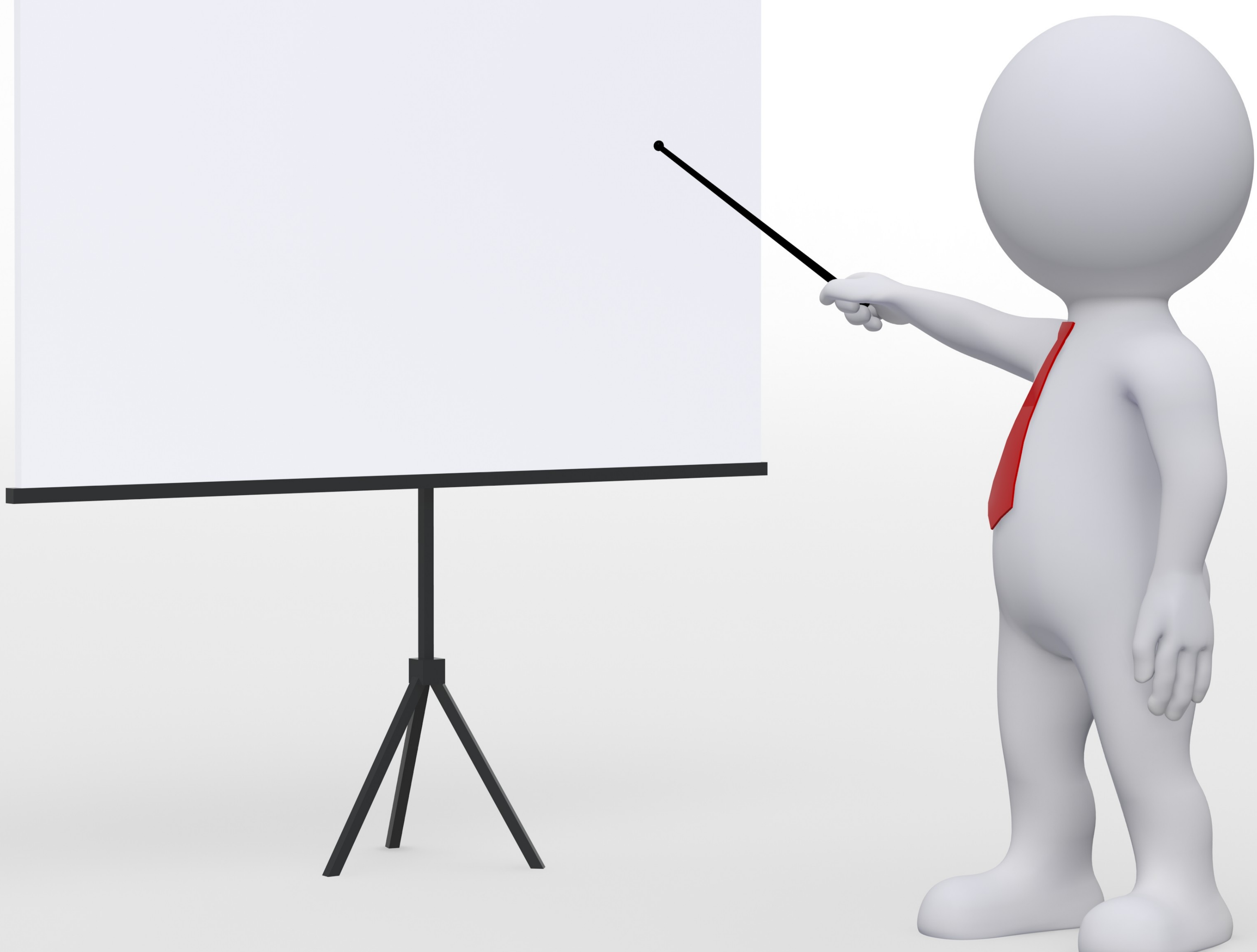


Ana Paliska

PMF – MO, Zagreb 15.01.2016.



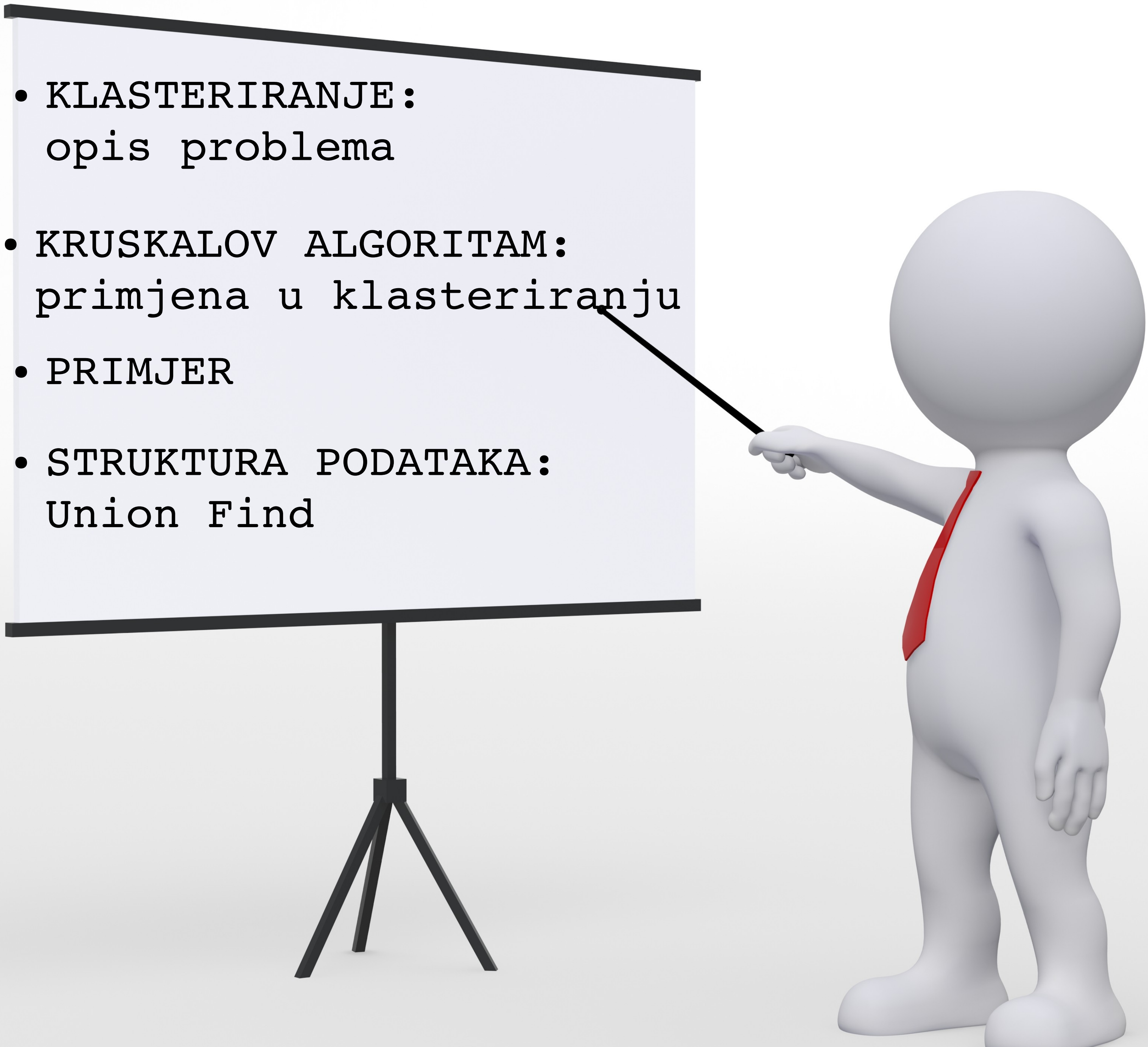
- KLASTERIRANJE: opis problema



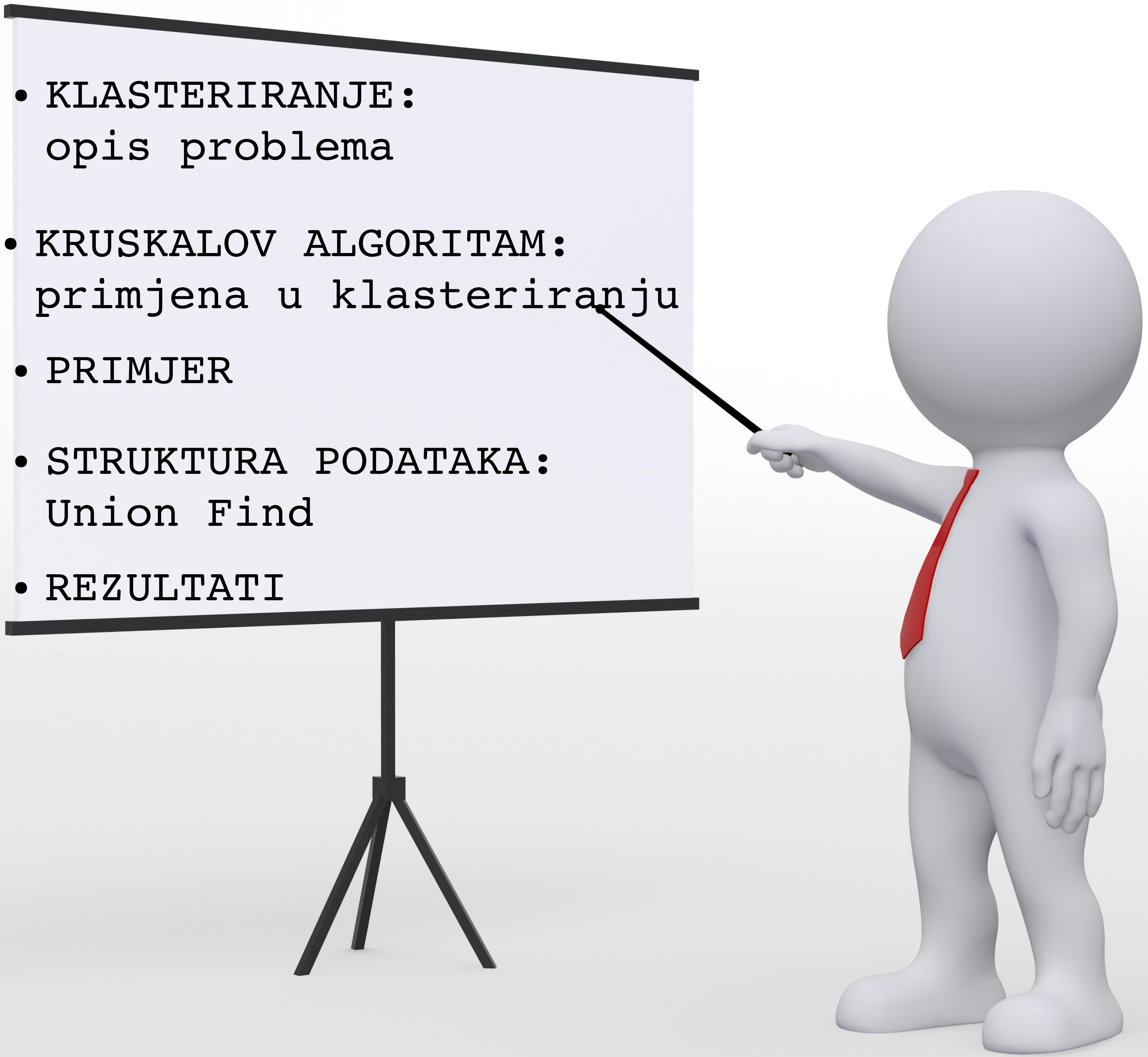
- KLASTERIRANJE:
  opis problema

- KRUSKALOV ALGORITAM:
  primjena u klasteriranju


• KLASTERIRANJE:
opis problema
• KRUSKALOV ALGORITAM:
primjena u klasteriranju
• PRIMJER



• KLASTERIRANJE:
opis problema
• KRUSKALOV ALGORITAM:
primjena u klasteriranju
• PRIMJER
• STRUKTURA PODATAKA:
Union Find


• KLASTERIRANJE:
opis problema
• KRUSKALOV ALGORITAM:
primjena u klasteriranju
• PRIMJER
• STRUKTURA PODATAKA:
Union Find
• REZULTATI


• KLASTERIRANJE: opis problema
• KRUSKALOV ALGORITAM: primjena u klasteriranju
• PRIMJER
• STRUKTURA PODATAKA: Union Find
• REZULTATI

KLASTERIRANJE: opis problema

KLASTERIRANJE: opis problema

- problem **grupiranja** objekata u podgrupe (klastere)
- slični objekti u istim klasterima
- *data mining, statistička analiza podataka, strojno učenje, prepoznavanje uzoraka...*

KLASTERIRANJE: opis problema

- problem **grupiranja** objekata u podgrupe (klastere)
- slični objekti u istim klasterima
- *data mining, statistička analiza podataka, strojno učenje, prepoznavanje uzoraka...*

- **mjera sličnosti** → udaljenost
- **reprezentacija grafom**: objekti su vrhovi povezani bridovima, a težina brida je udaljenost

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

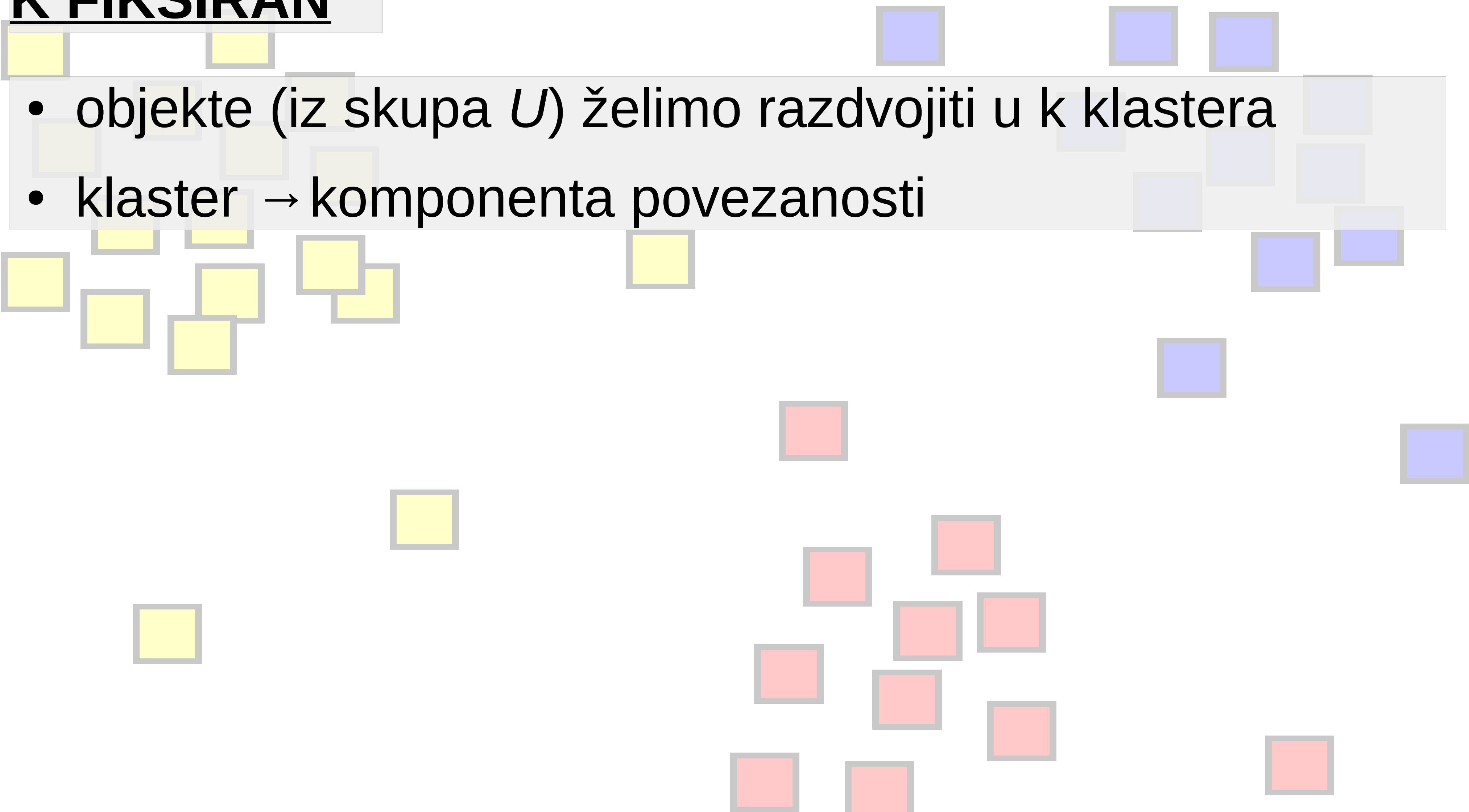
K FIKSIRAN

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

<u>K FIKSIRAN</u>

- objekte (iz skupa *U*) želimo razdvojiti u k klastera
- klaster → komponenta povezanosti

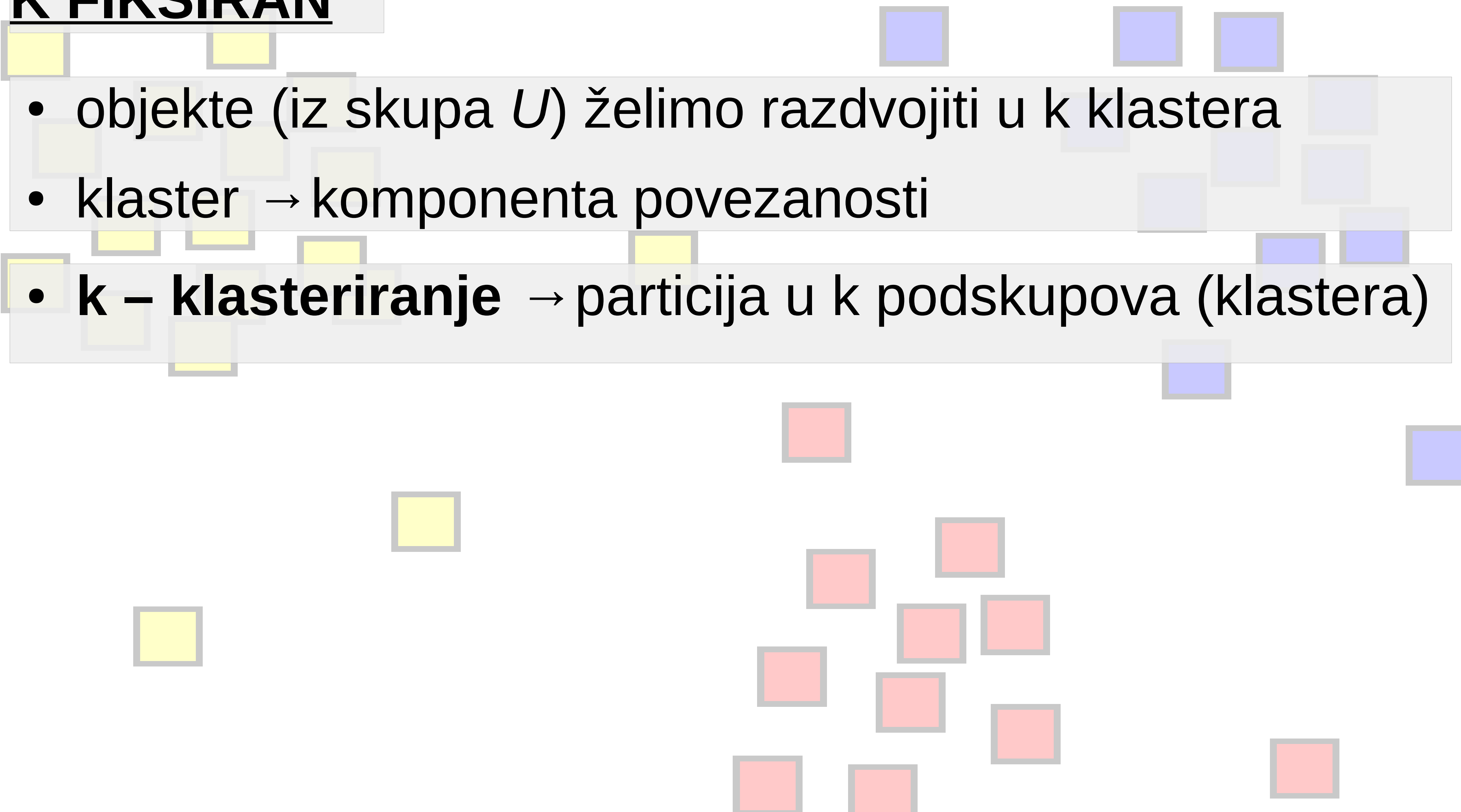


KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

K FIKSIRAN

- objekte (iz skupa *U*) želimo razdvojiti u k klastera
- klaster → komponenta povezanosti
- **k – klasteriranje** → particija u k podskupova (klastera)



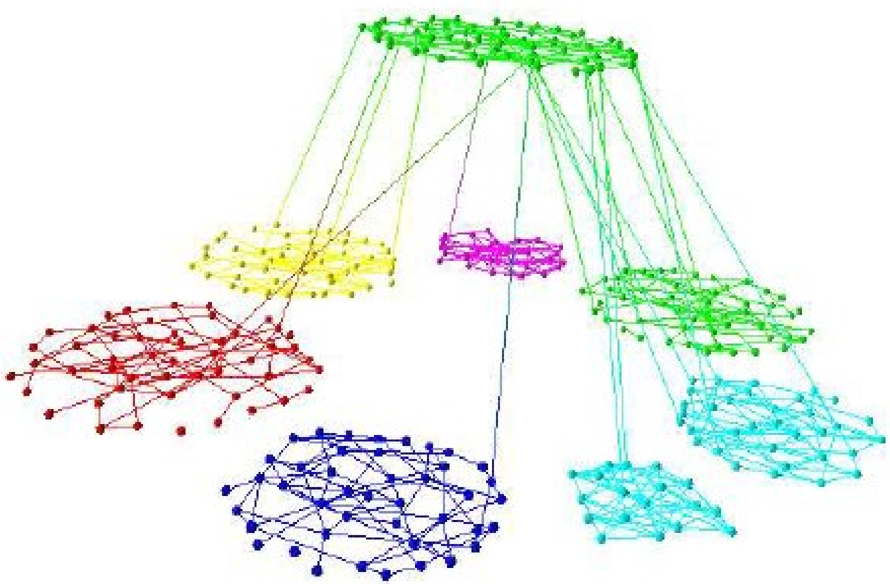
KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

<u>K FIKSIRAN</u>

- objekte (iz skupa *U*) želimo razdvojiti u k klastera
- klaster →komponenta povezanosti

- **k – klasteriranje** →particija u k podskupova (klastera)



- **razmak k – klastera** →<u>minimalna</u> udaljenost između bilo kojeg para vrhova koji leže u različitim klasterima
  - <u>minimalna</u> udaljenost između klastera

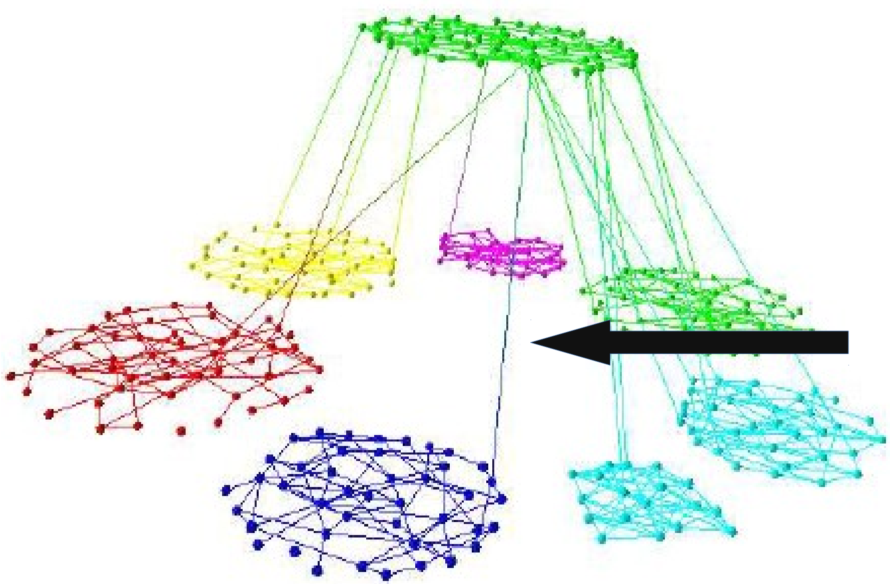
KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

<u>K FIKSIRAN</u>

- objekte (iz skupa *U*) želimo razdvojiti u k klastera
- klaster → komponenta povezanosti

- **k – klasteriranje** → particija u k podskupova (klastera)



- **razmak k – klastera** → <u>minimalna</u> udaljenost između bilo kojeg para vrhova koji leže u različitim klasterima
  - <u>minimalna</u> udaljenost između klastera

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

- **PROBLEM SE SVODI NA**
  - pronalazak k – klastera s **maksimalnim razmakom**

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

- **PROBLEM SE SVODI NA**
  - pronalazak k – klastera s **maksimalnim razmakom**
- **problem:**
  - eksponencijalno mnogo k - klastera

KLASTERIRANJE: opis problema

Klasteriranje s maksimalnim razmakom

- **PROBLEM SE SVODI NA**
  - pronalazak k – klastera s **maksimalnim razmakom**
- **problem:**
  - eksponencijalno mnogo k - klastera
- **pomoć:**
  - Kruskalov algoritam

KRUSKALOV ALGORITAM

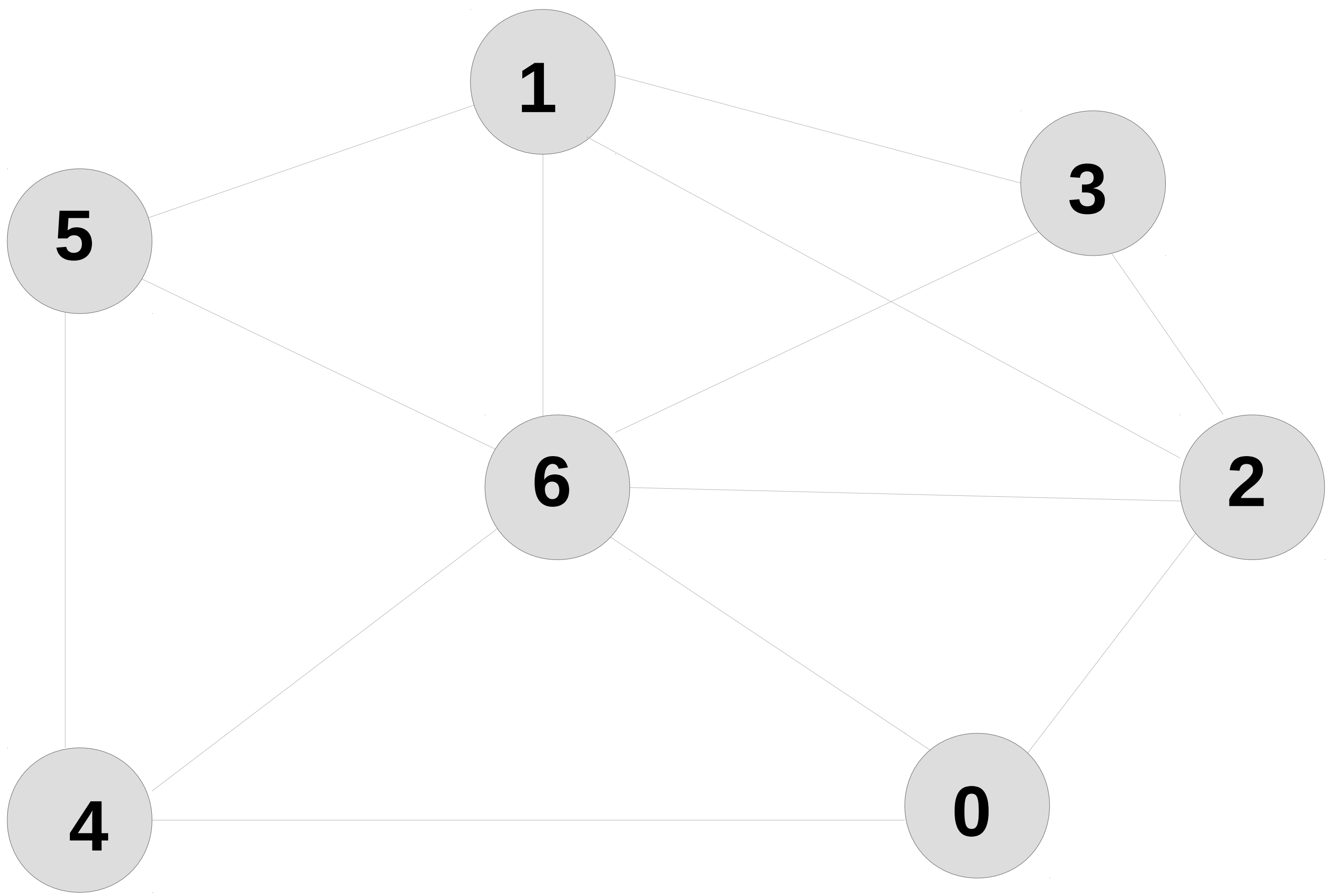
Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

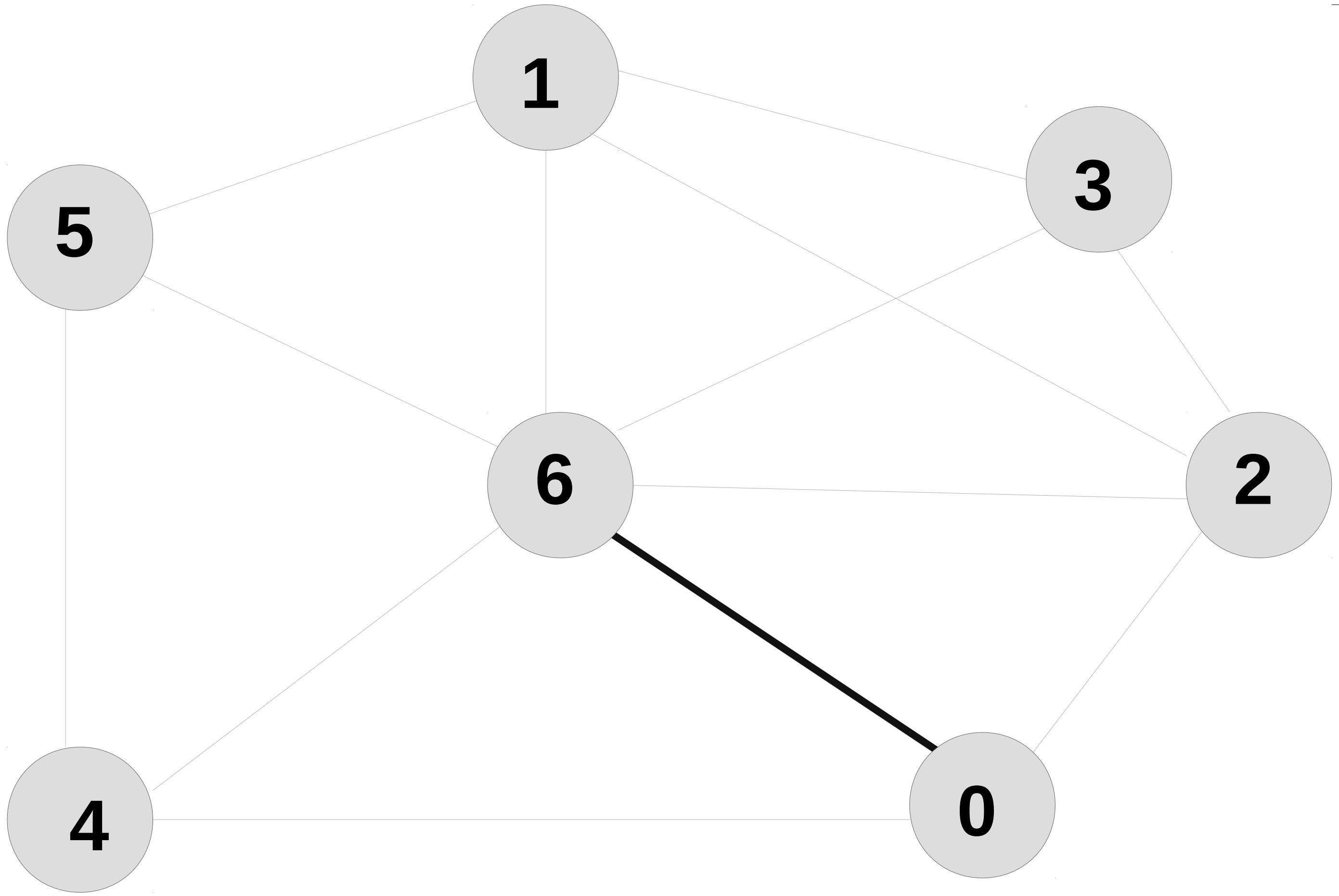


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

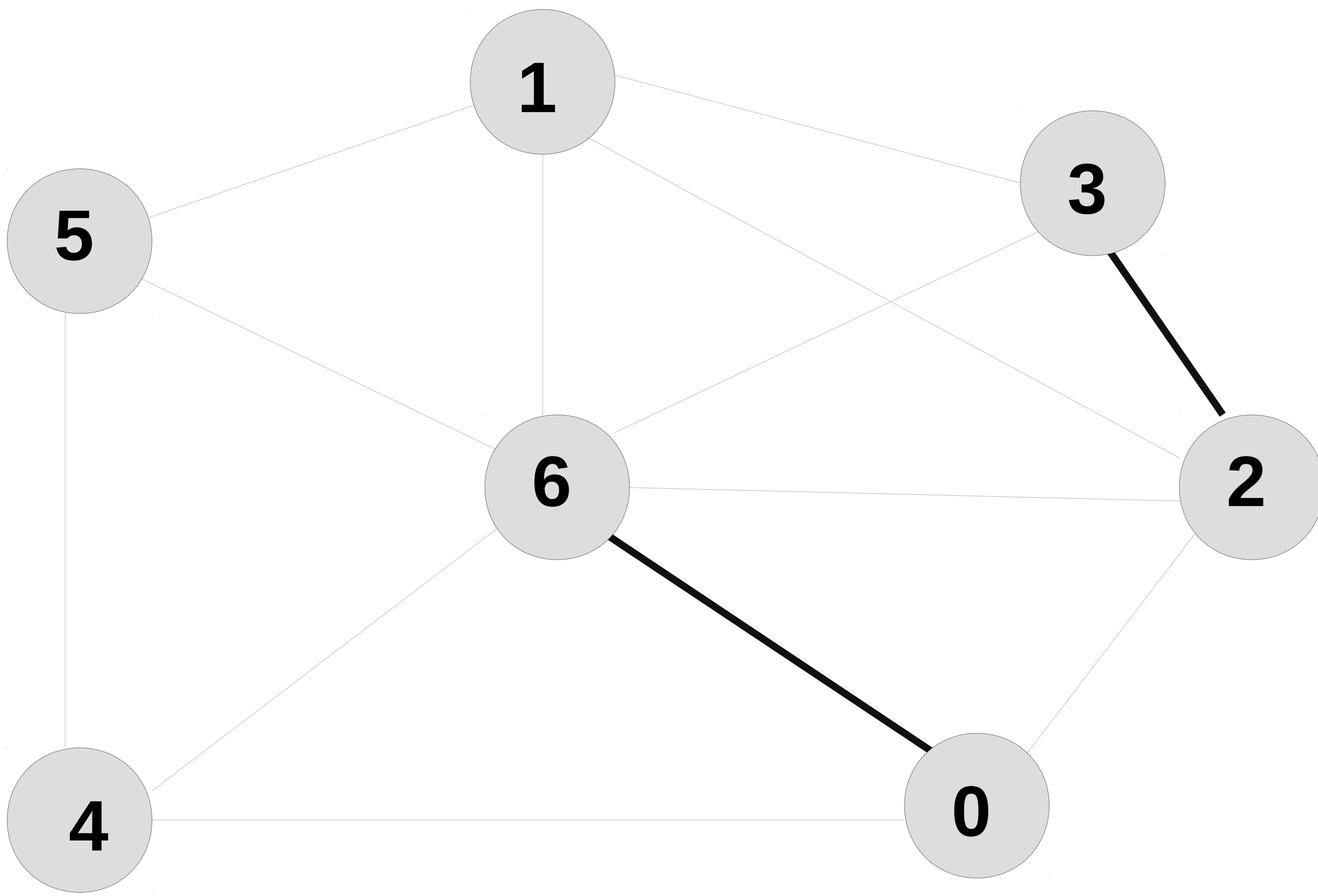


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

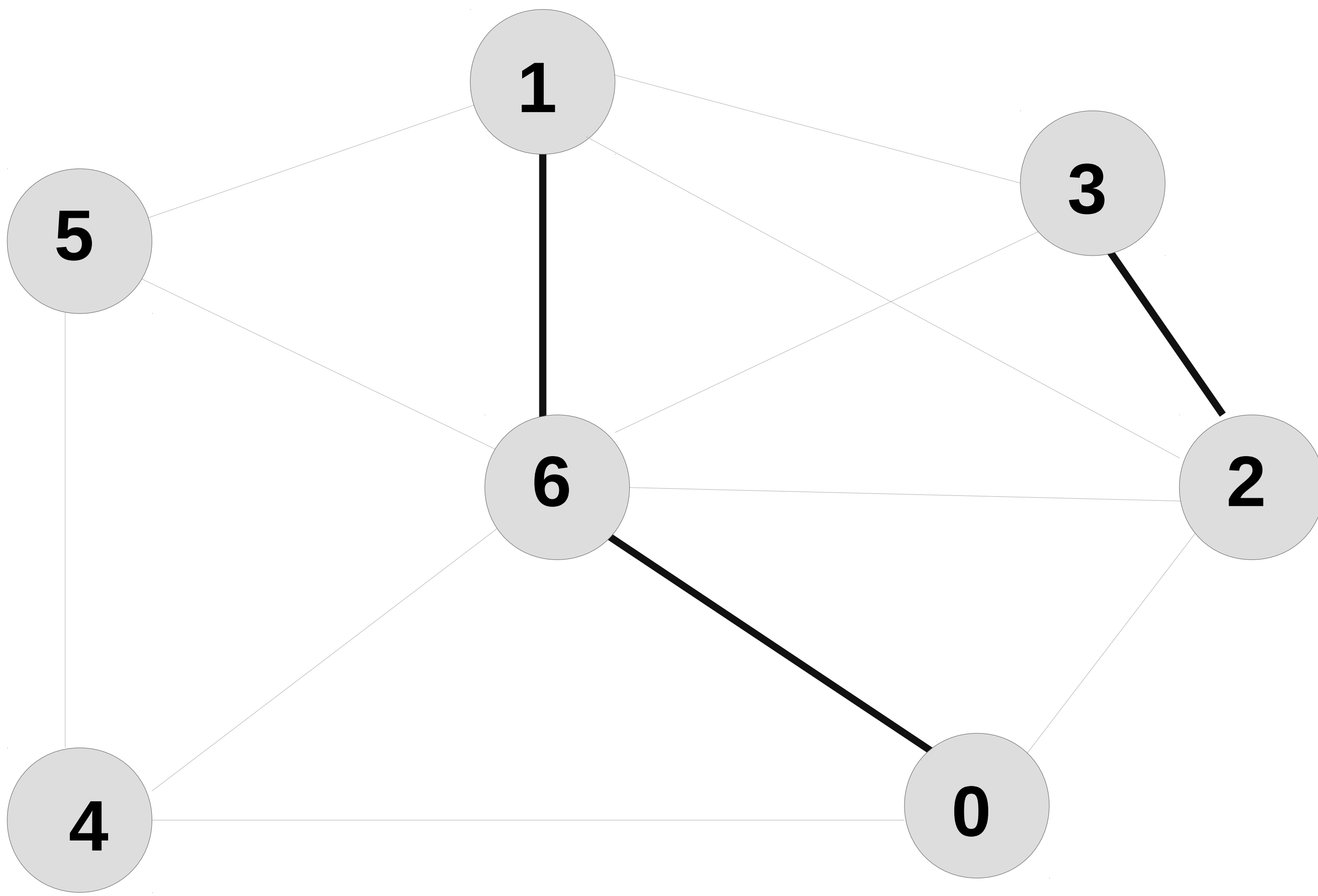


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

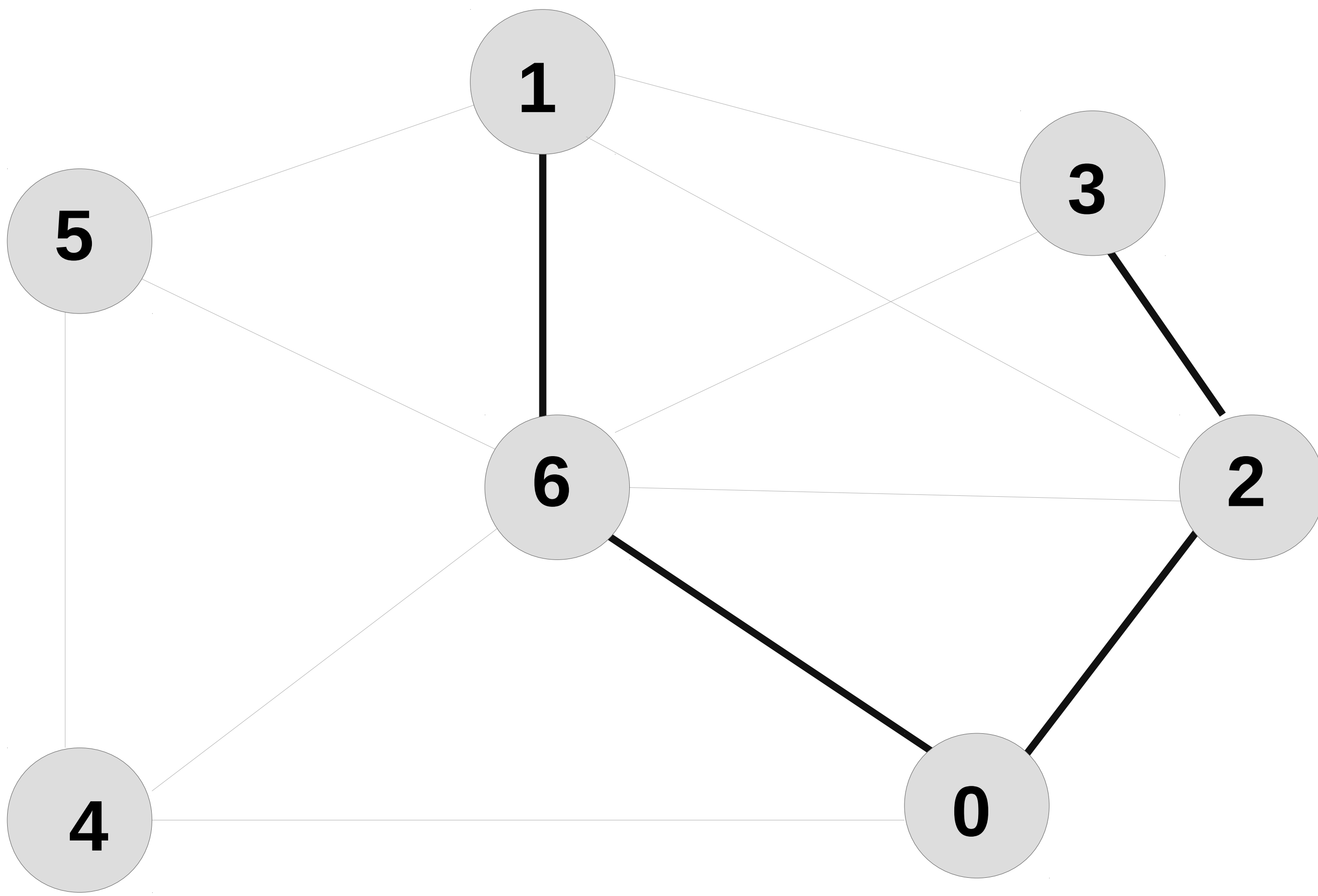


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

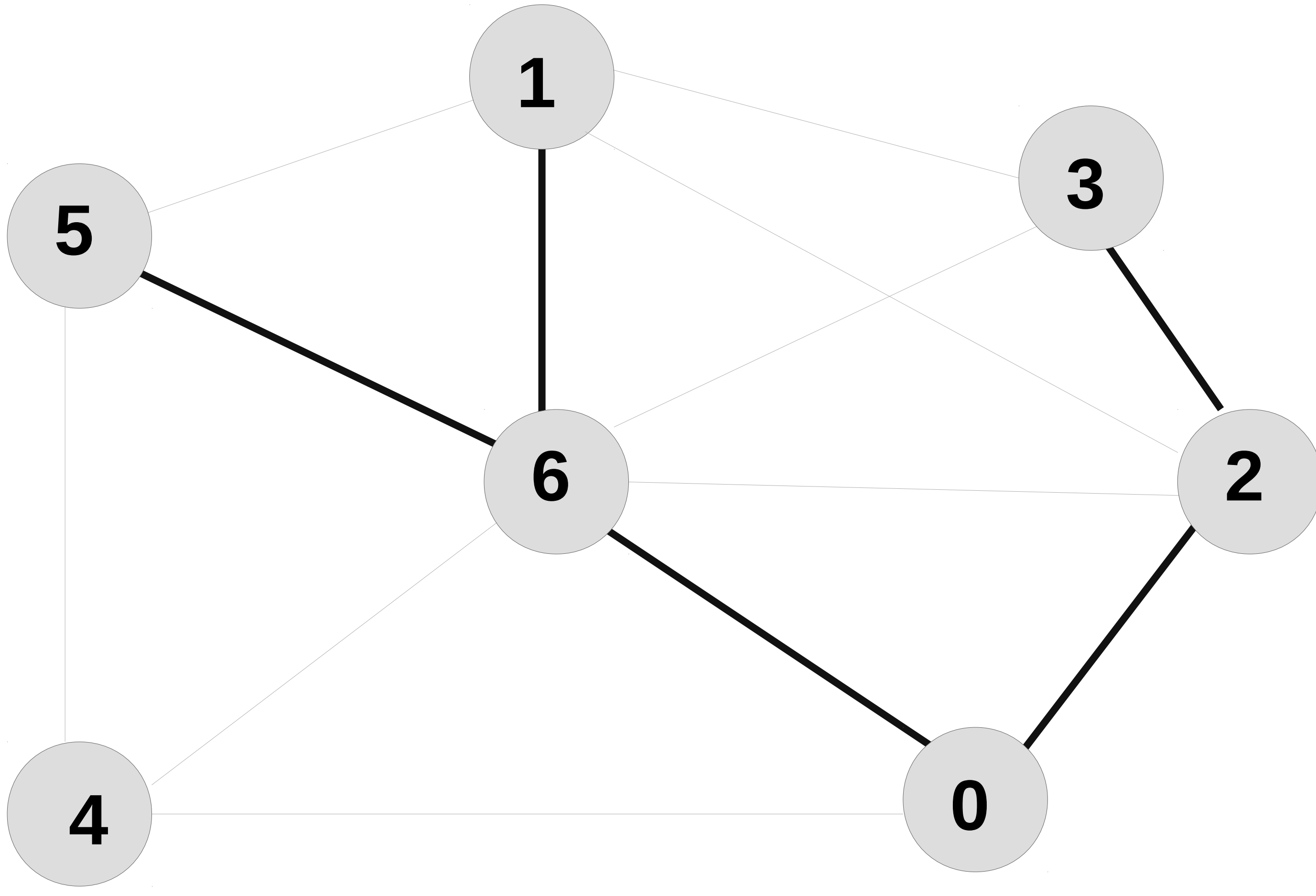


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

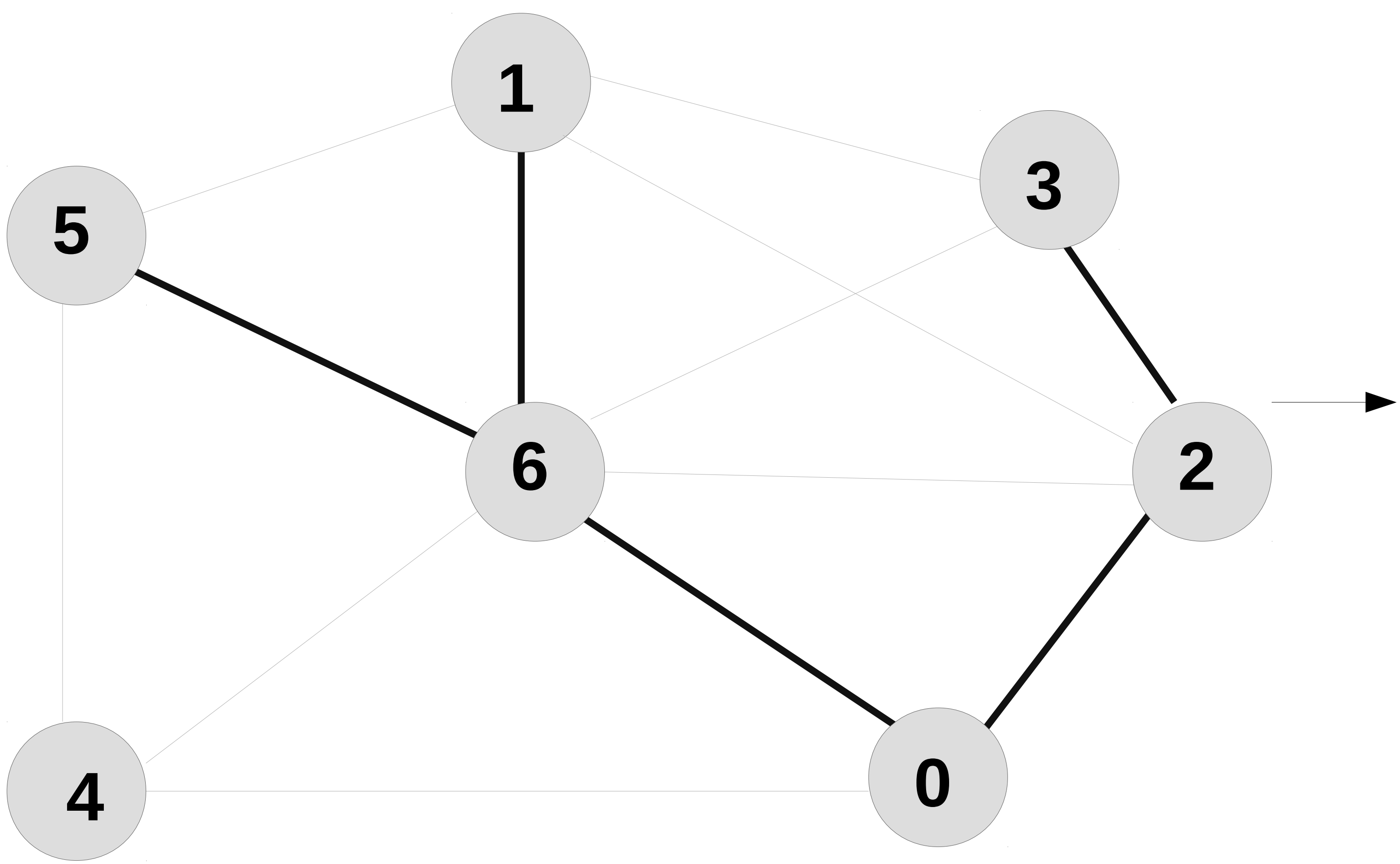


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

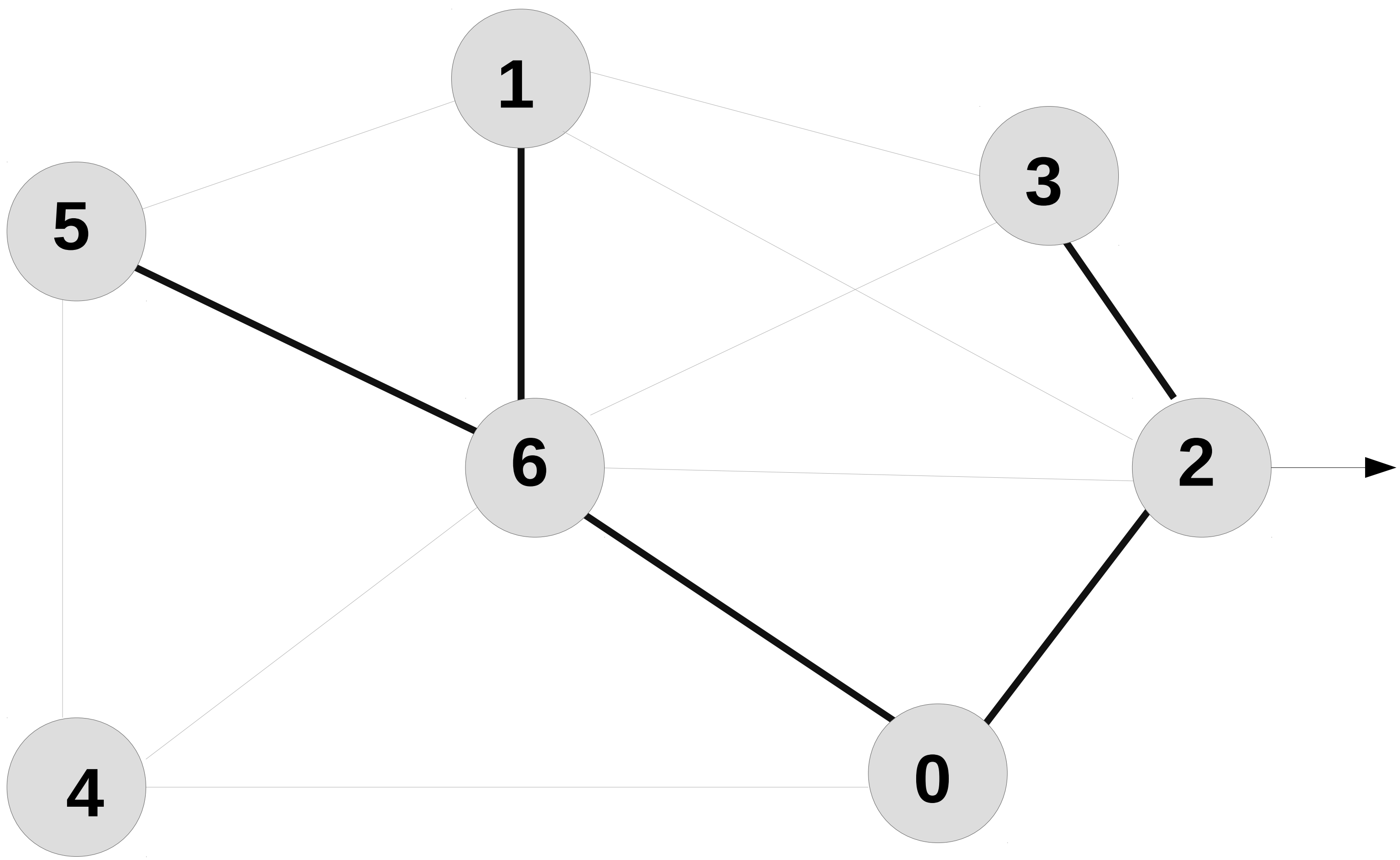


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

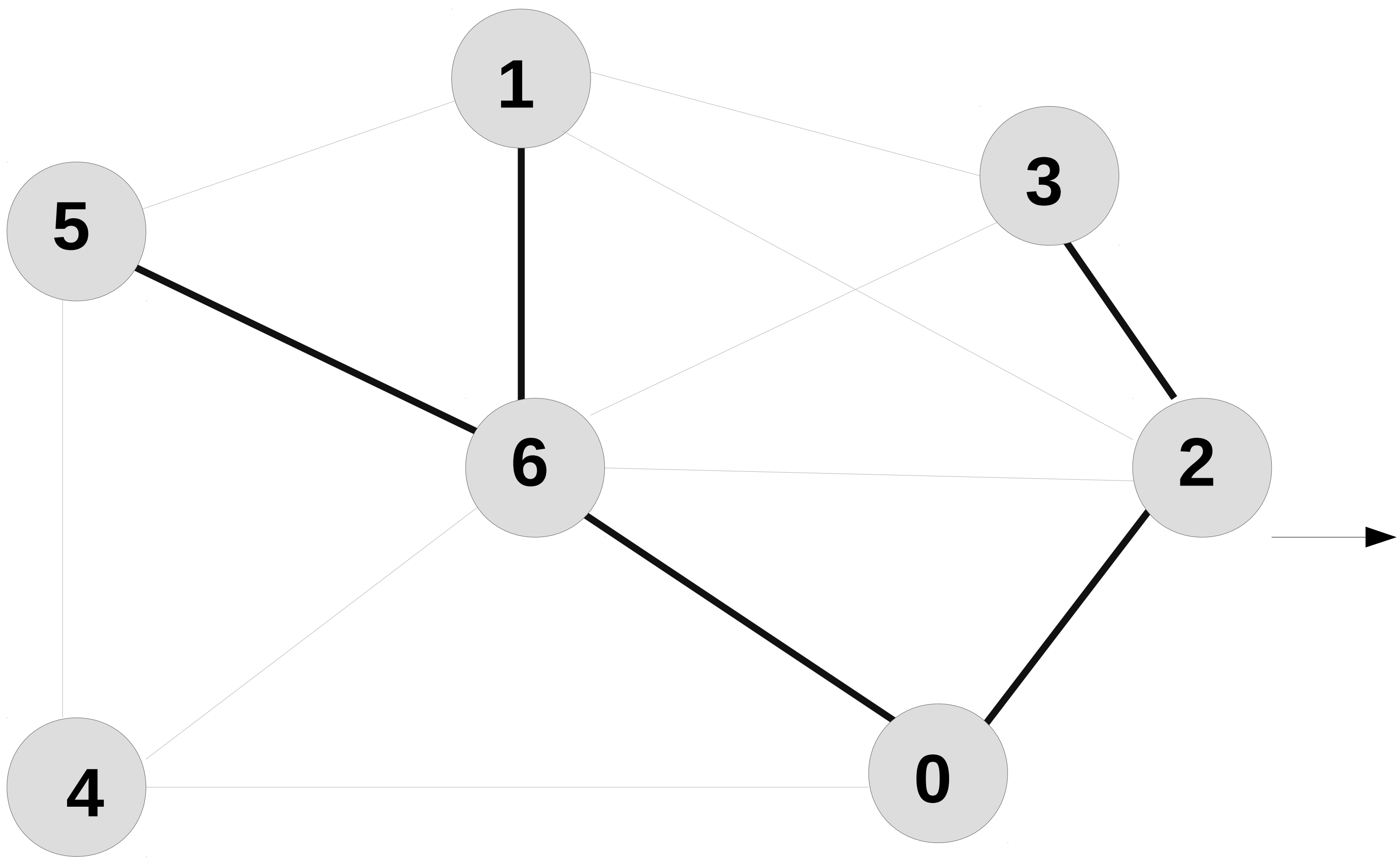


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

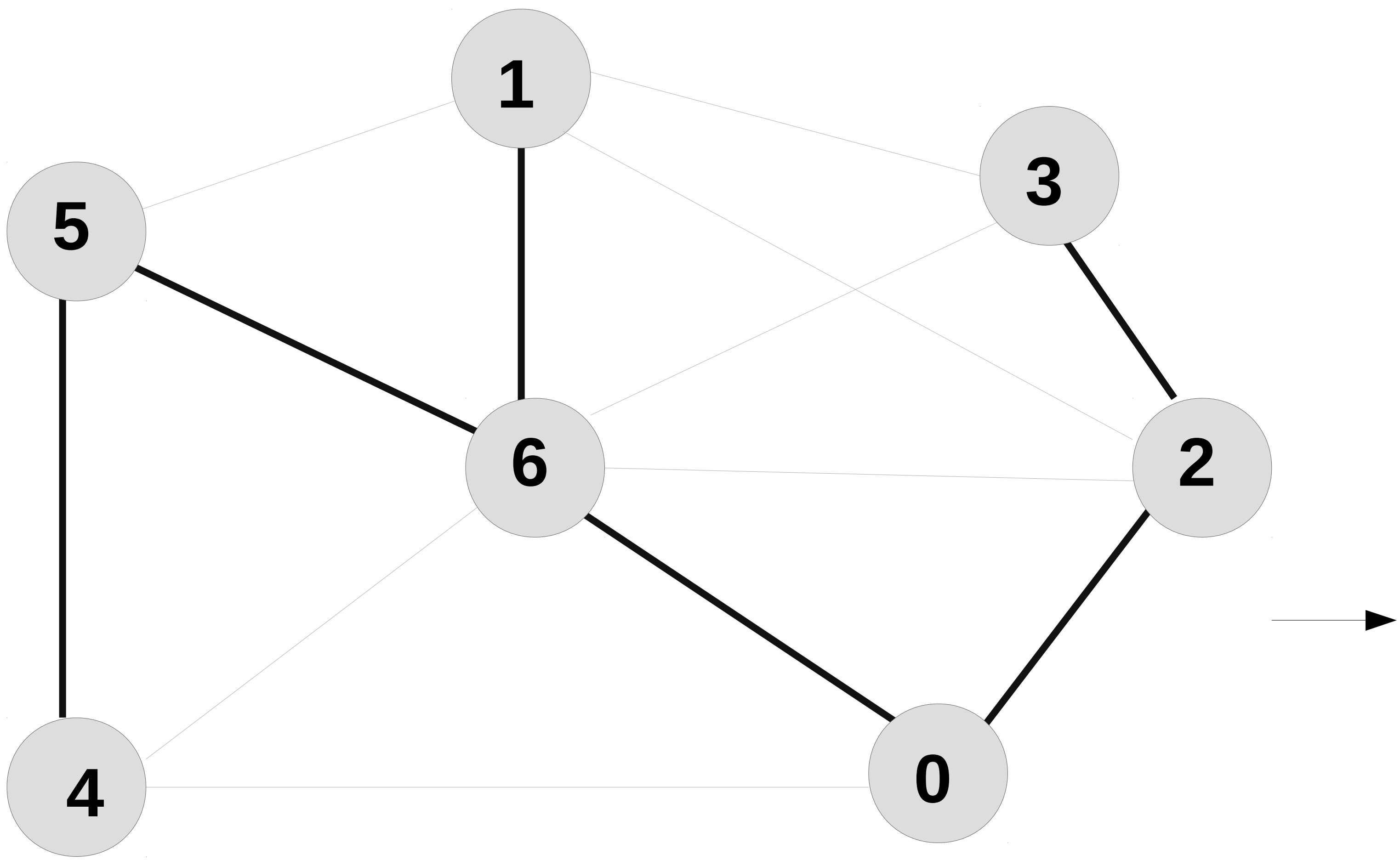


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

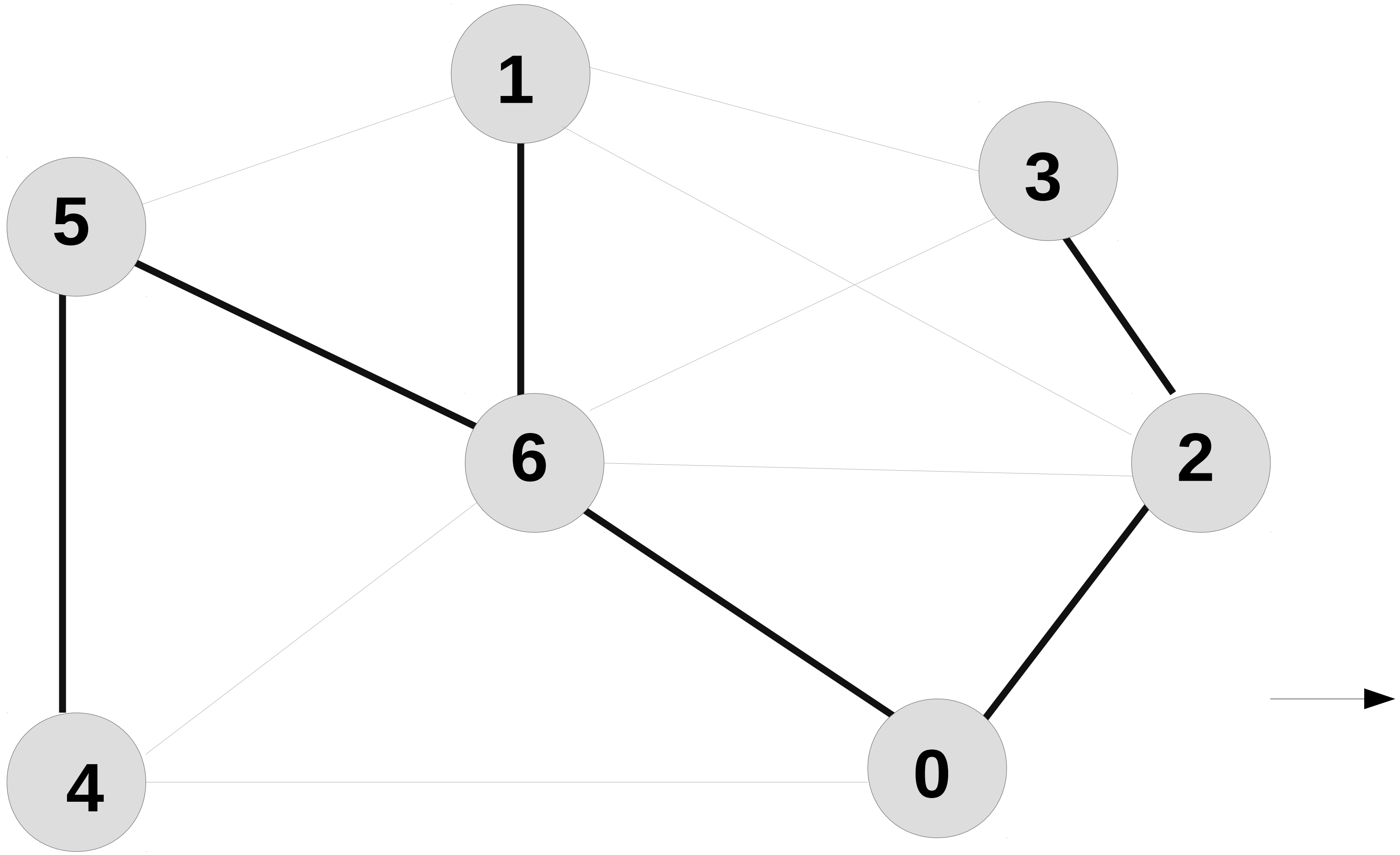


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

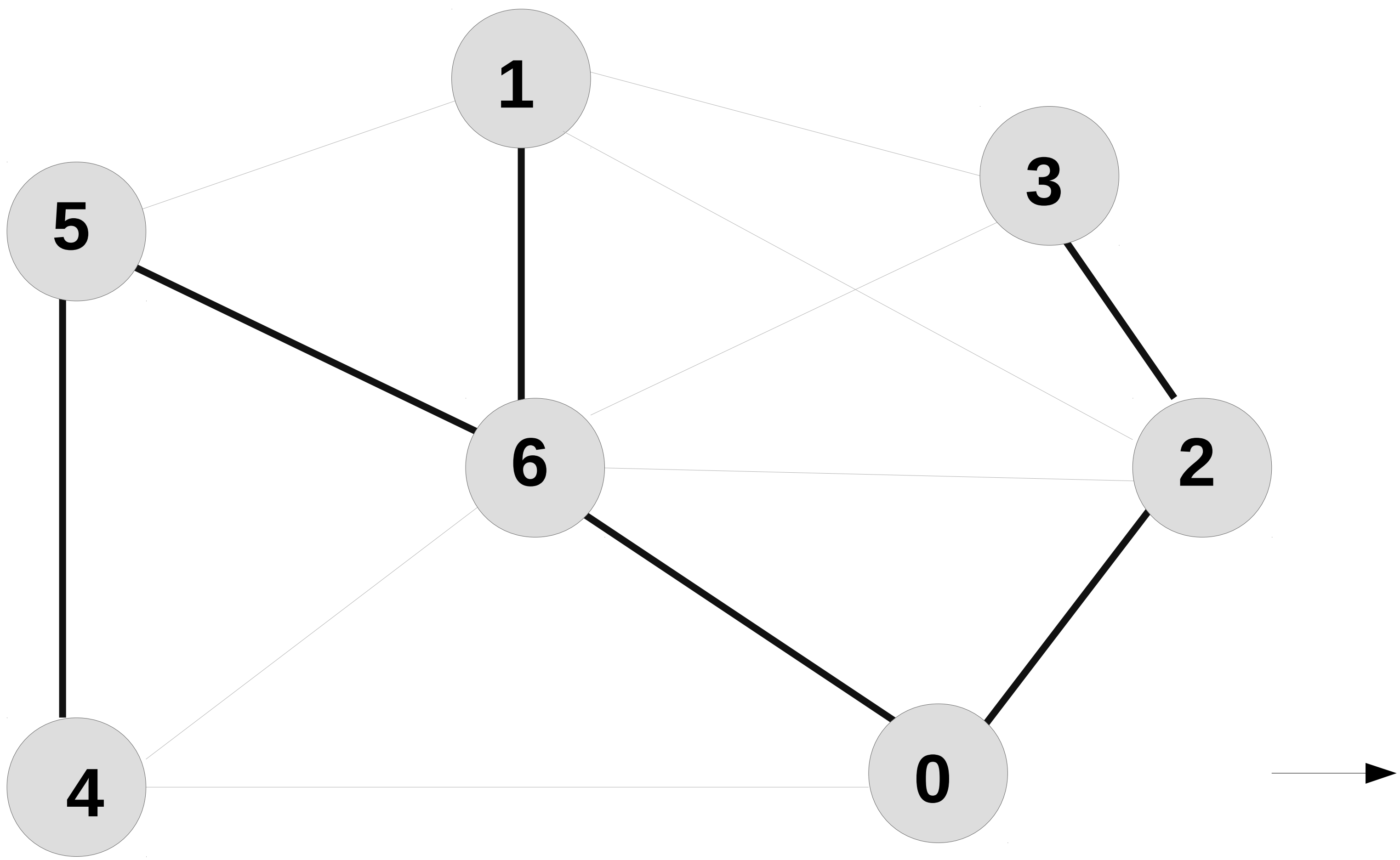


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla

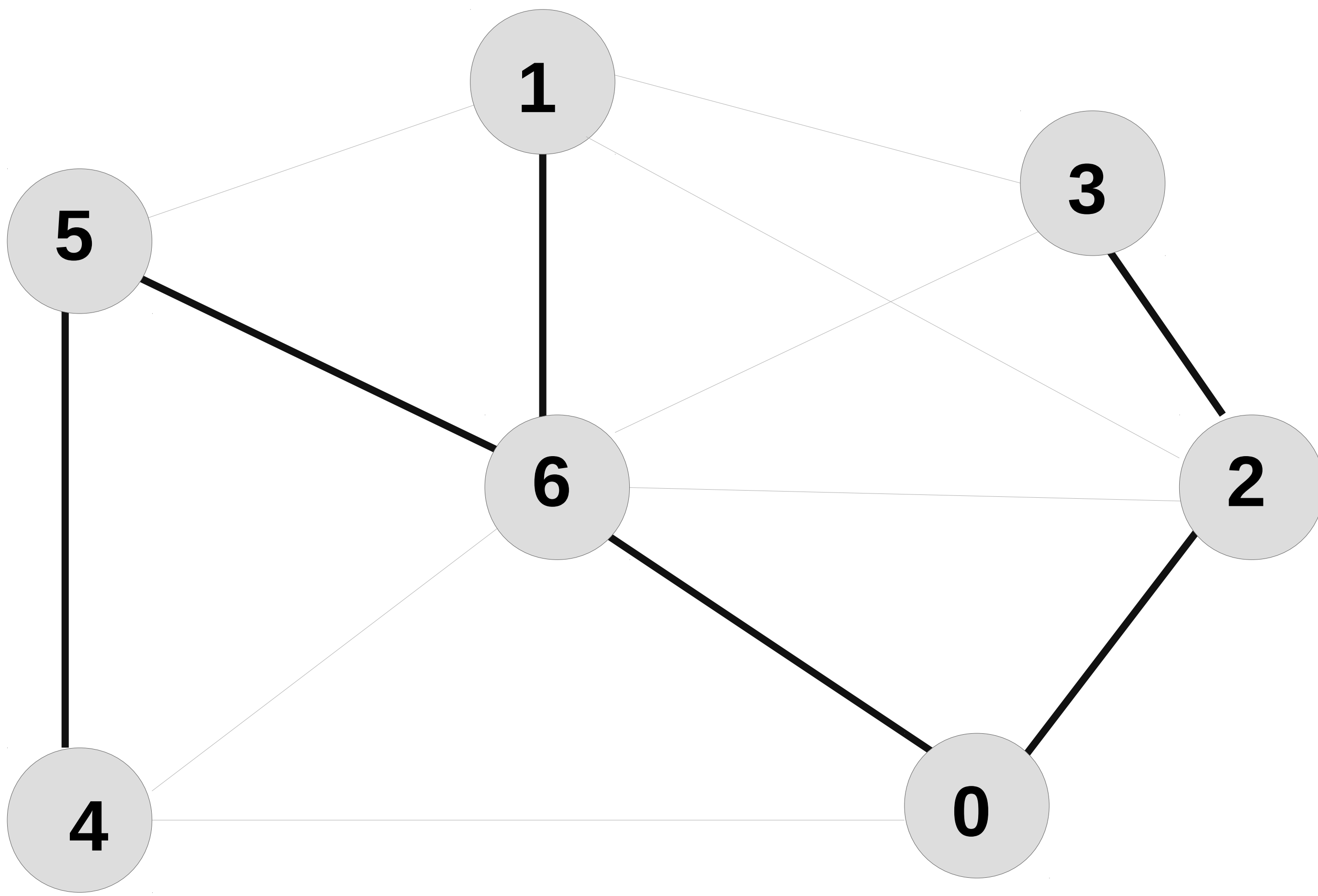


0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

Minimalno razapinjuće stablo

- pohlepni algoritam za traženje minimalnog razapinjućeg stabla



0 – 6	0.16
2 – 3	0.17
1 – 6	0.19
0 – 2	0.26
5 – 6	0.28
1 – 3	0.29
1 – 5	0.32
2 – 6	0.34
4 – 5	0.35
1 – 2	0.36
4 – 6	0.37
0 – 4	0.38

KRUSKALOV ALGORITAM

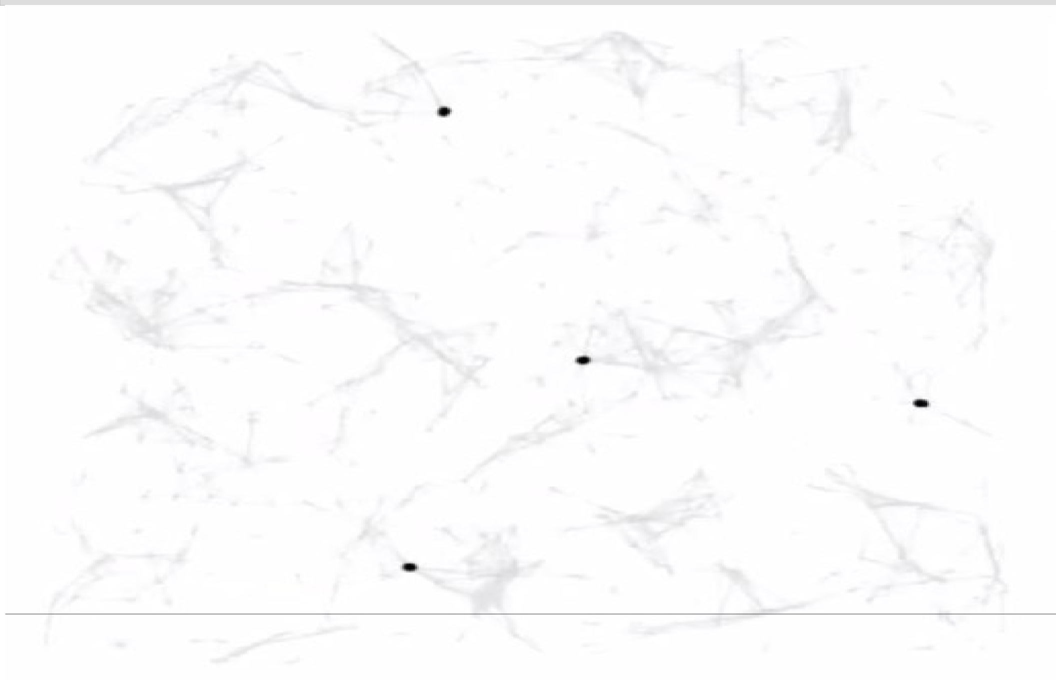
Primjena u klasteriranju

KRUSKALOV ALGORITAM

Primjena u klasteriranju

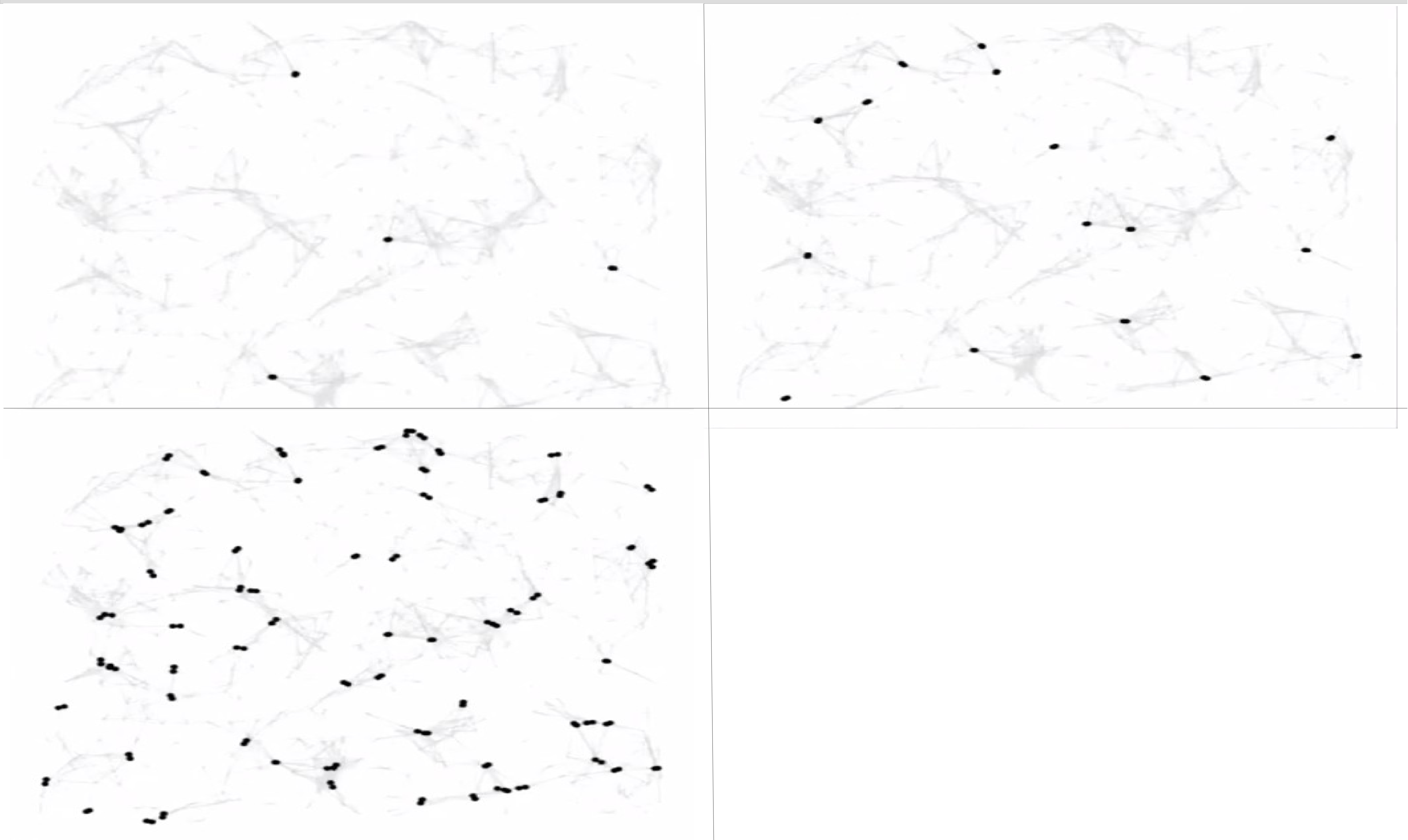
KRUSKALOV ALGORITAM

Primjena u klasteriranju



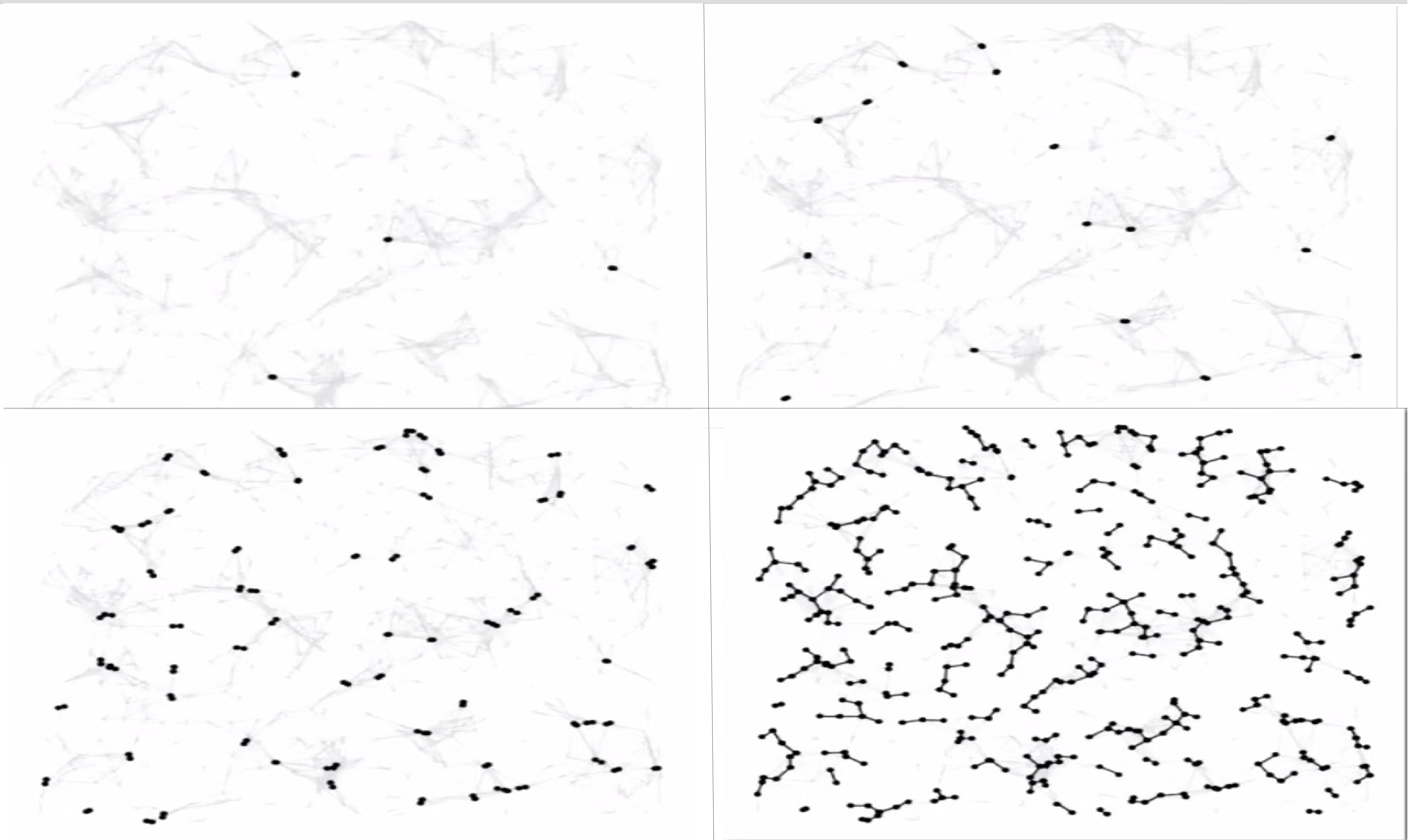
KRUSKALOV ALGORITAM

Primjena u klasteriranju



KRUSKALOV ALGORITAM

Primjena u klasteriranju



KRUSKALOV ALGORITAM

Primjena u klasteriranju

KRUSKALOV ALGORITAM

Primjena u klasteriranju

- problem k – klasteriranja svodi se na traženje minimalnog razapinjućeg stabla (MST) pomoću Kruskalovog algoritma

KRUSKALOV ALGORITAM

Primjena u klasteriranju

- problem k – klasteriranja svodi se na traženje minimalnog razapinjućeg stabla (MST) pomoću Kruskalovog algoritma
- algoritam staje kada nastane k komponenti povezanosti $C_1,\ldots,C_k$

KRUSKALOV ALGORITAM

Primjena u klasteriranju

- problem k – klasteriranja svodi se na traženje minimalnog razapinjućeg stabla (MST) pomoću Kruskalovog algoritma
- algoritam staje kada nastane k komponenti povezanosti $C_1,\ldots,C_k$
- ekvivalentno izbacivanju k – 1 najtežih bridova iz MST – a dobivenog Kruskalovim algoritmom

KRUSKALOV ALGORITAM

Primjena u klasteriranju

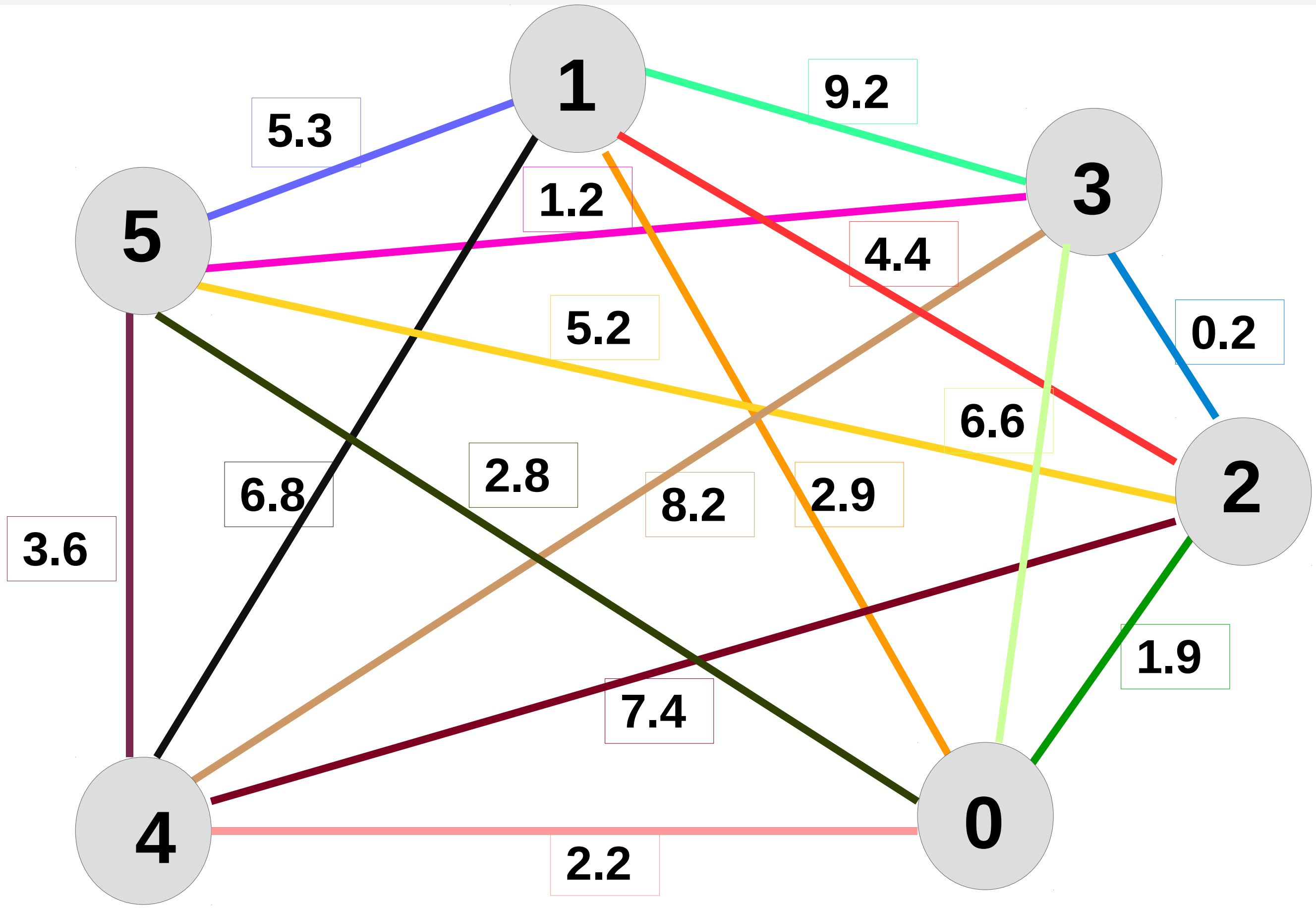
- problem k – klasteriranja svodi se na traženje minimalnog razapinjućeg stabla (MST) pomoću Kruskalovog algoritma
- algoritam staje kada nastane k komponenti povezanosti $C_1,\ldots,C_k$
- ekvivalentno izbacivanju k – 1 najtežih bridova iz MST – a dobivenog Kruskalovim algoritmom

Teorem

Komponente povezanosti $C_1,\ldots,C_k$ *dobivene micanjem k-1 najtežih bridova iz MST – a Kruskalovog algoritma definira k – kluster s maksimalnim razmakom.*

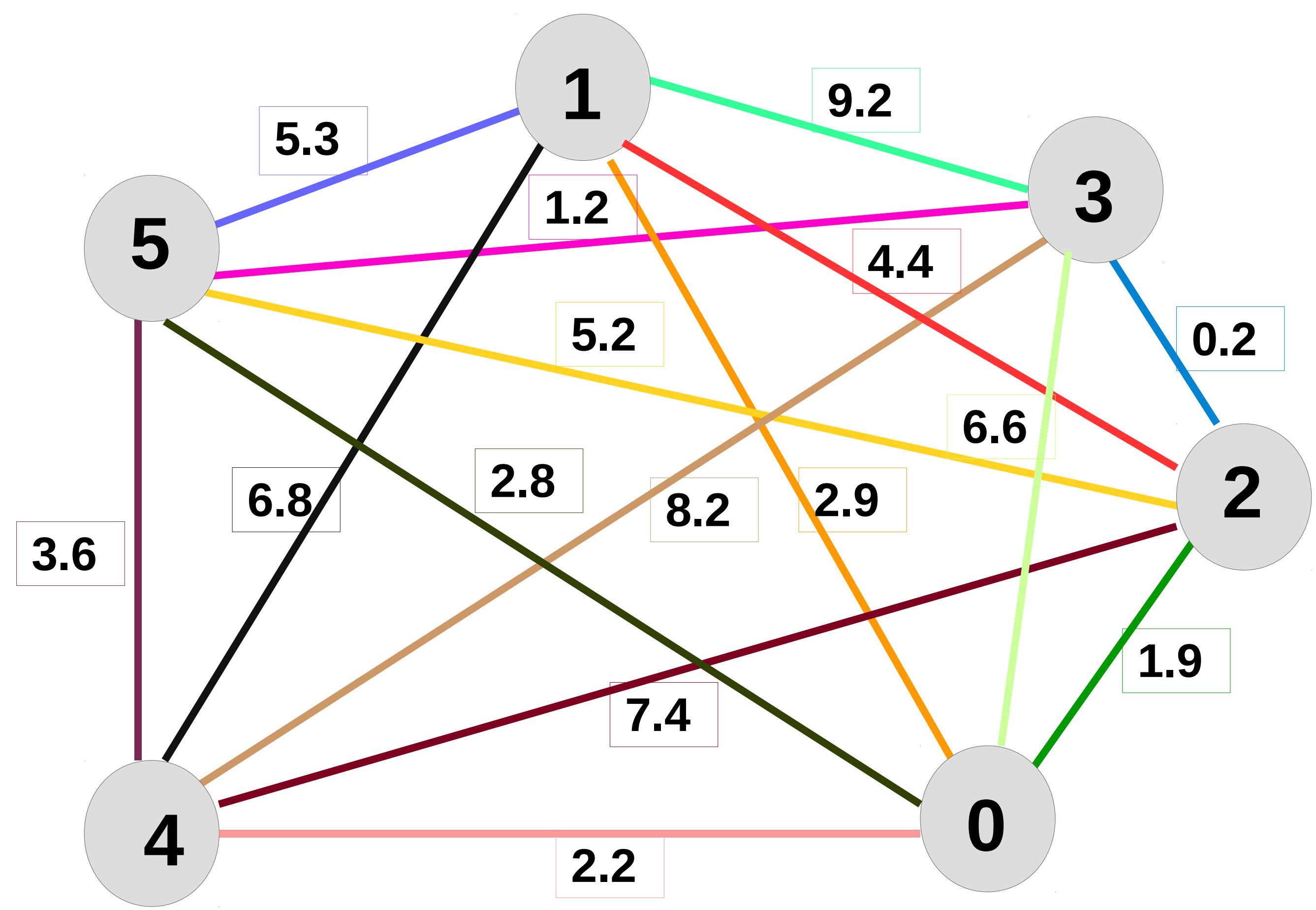
PRIMJER – rad algoritma

- pretpostavimo da je dan skup od 6 objekata koji je prezentiran grafom od **6 vrhova** i **15 bridova** (sve udaljenosti su definirane):



PRIMJER – rad algoritma

1. **sortiramo** bridove po težini

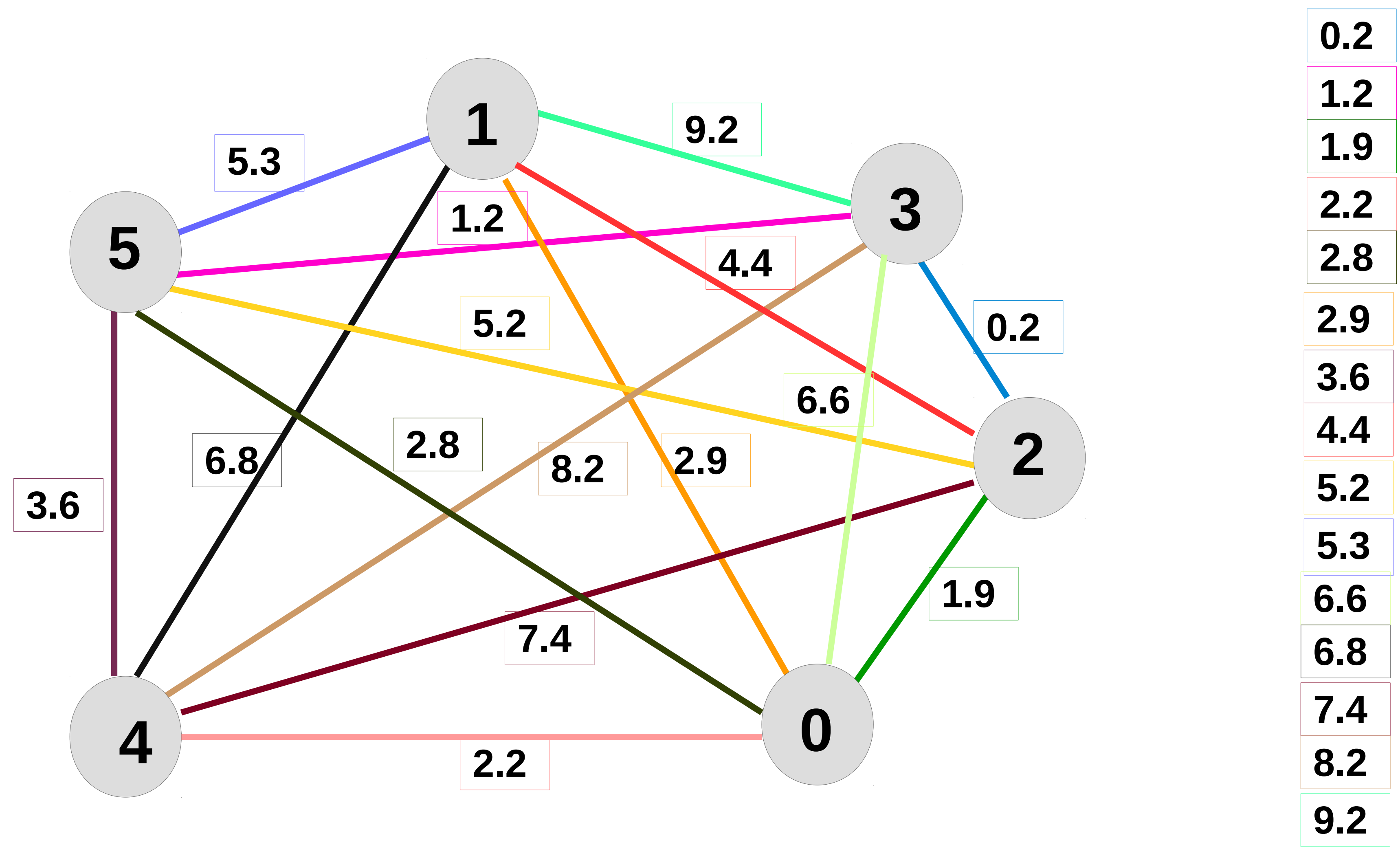


0.2
1.2
1.9
2.2
2.8
2.9
3.6
4.4
5.2
5.3
6.6
6.8
7.4
8.2
9.2

PRIMJER – rad algoritma

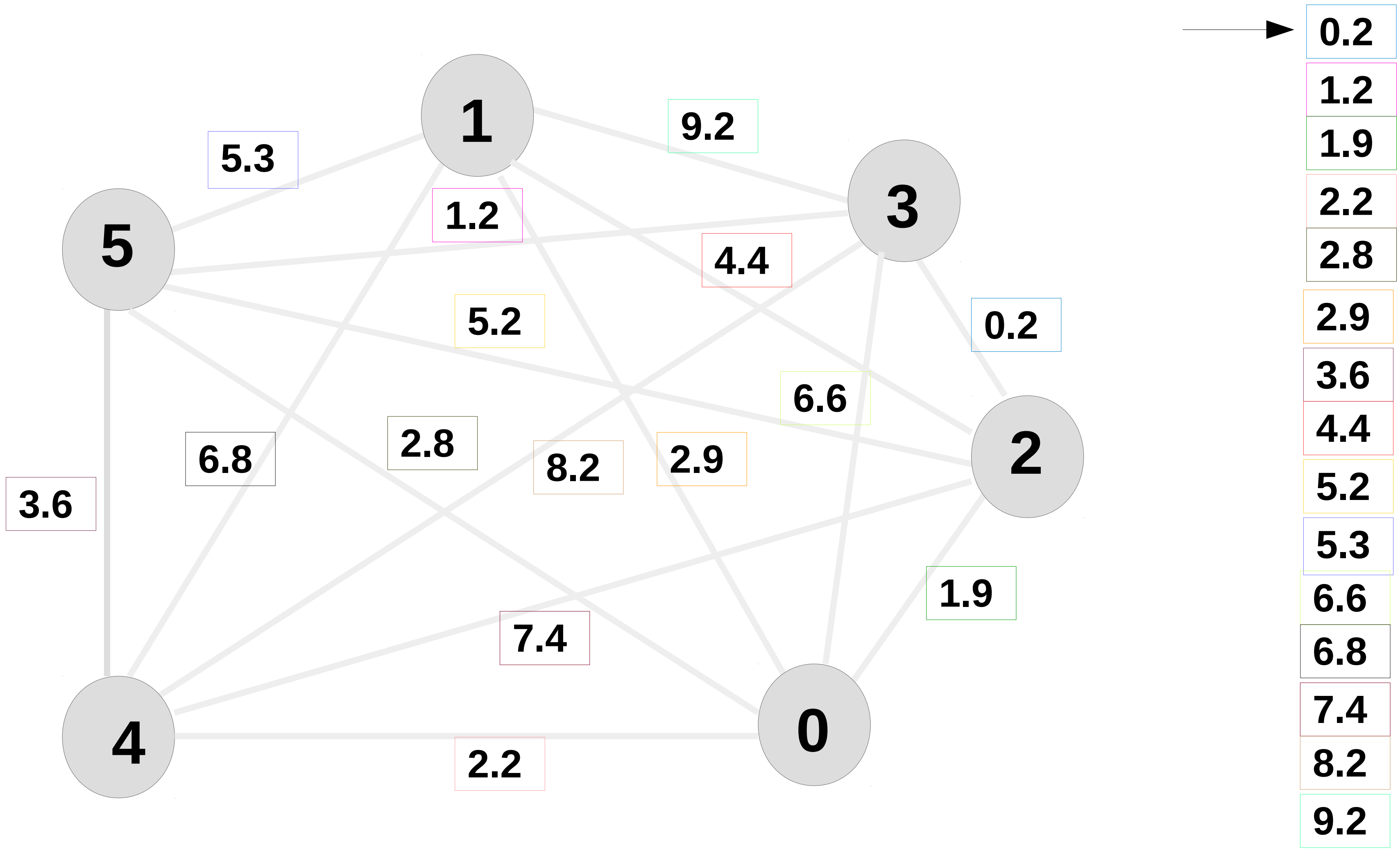
2. *odaberemo* **broj klastera k**

k = 3



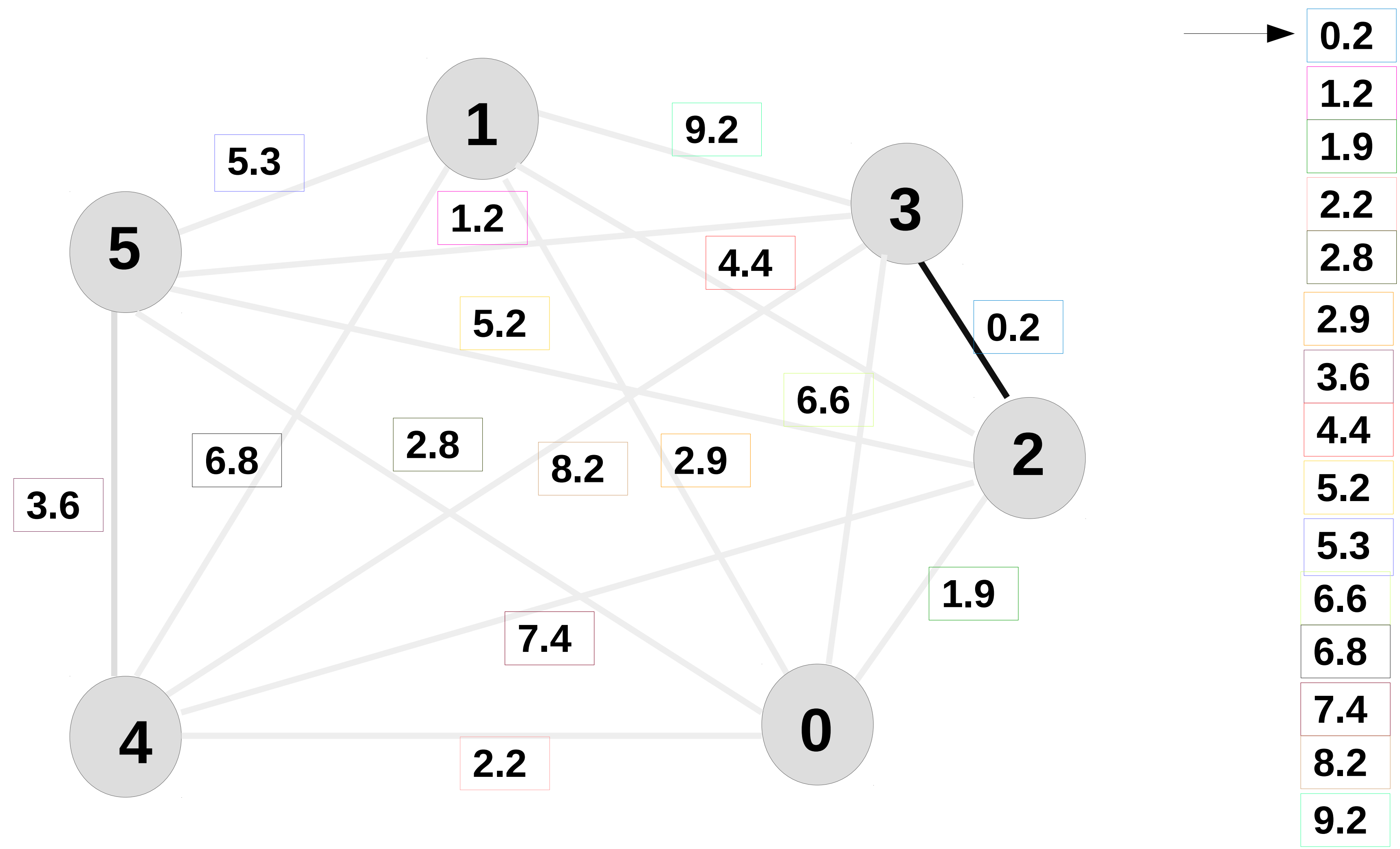
PRIMJER – rad algoritma

3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



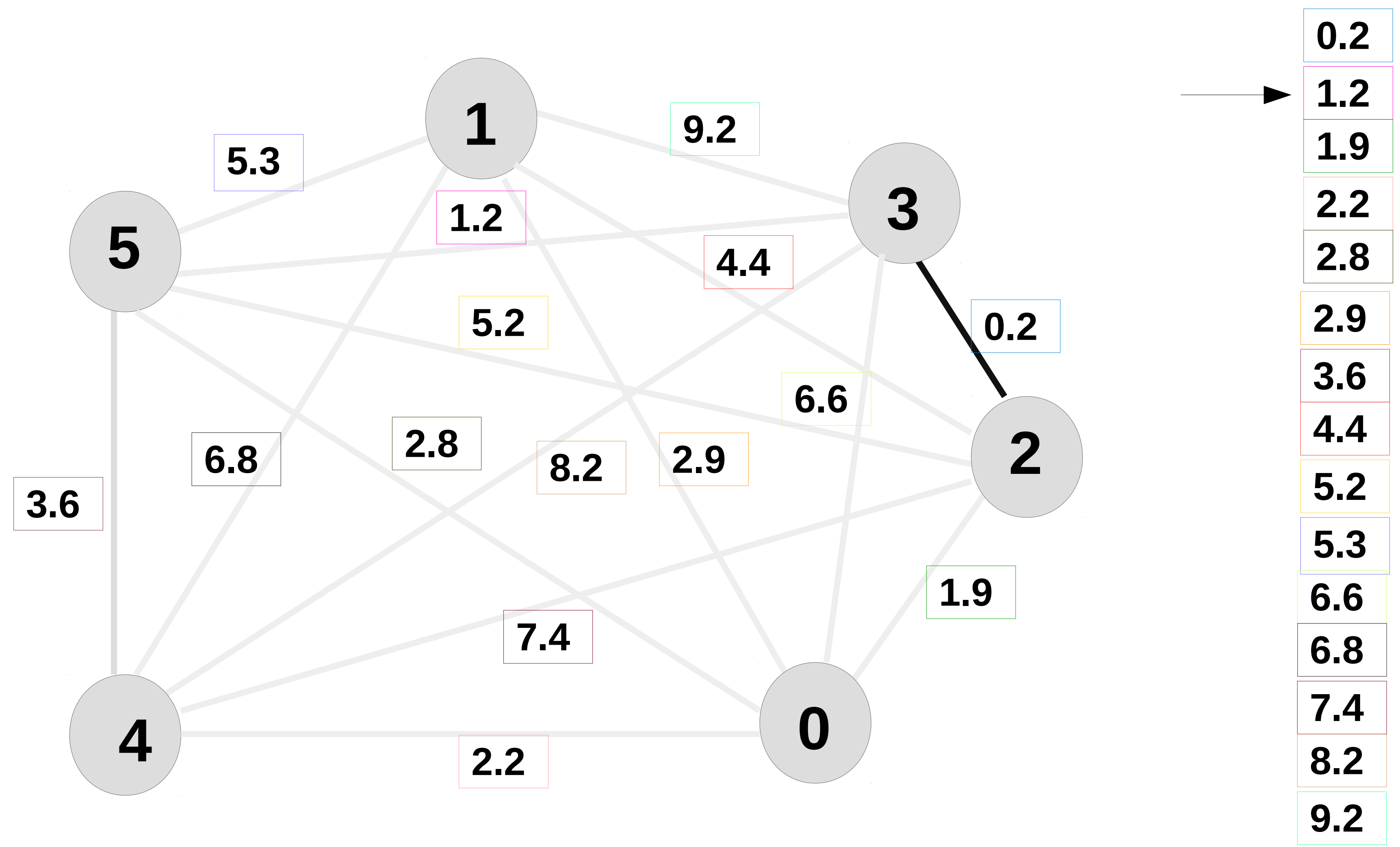
PRIMJER – rad algoritma

3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



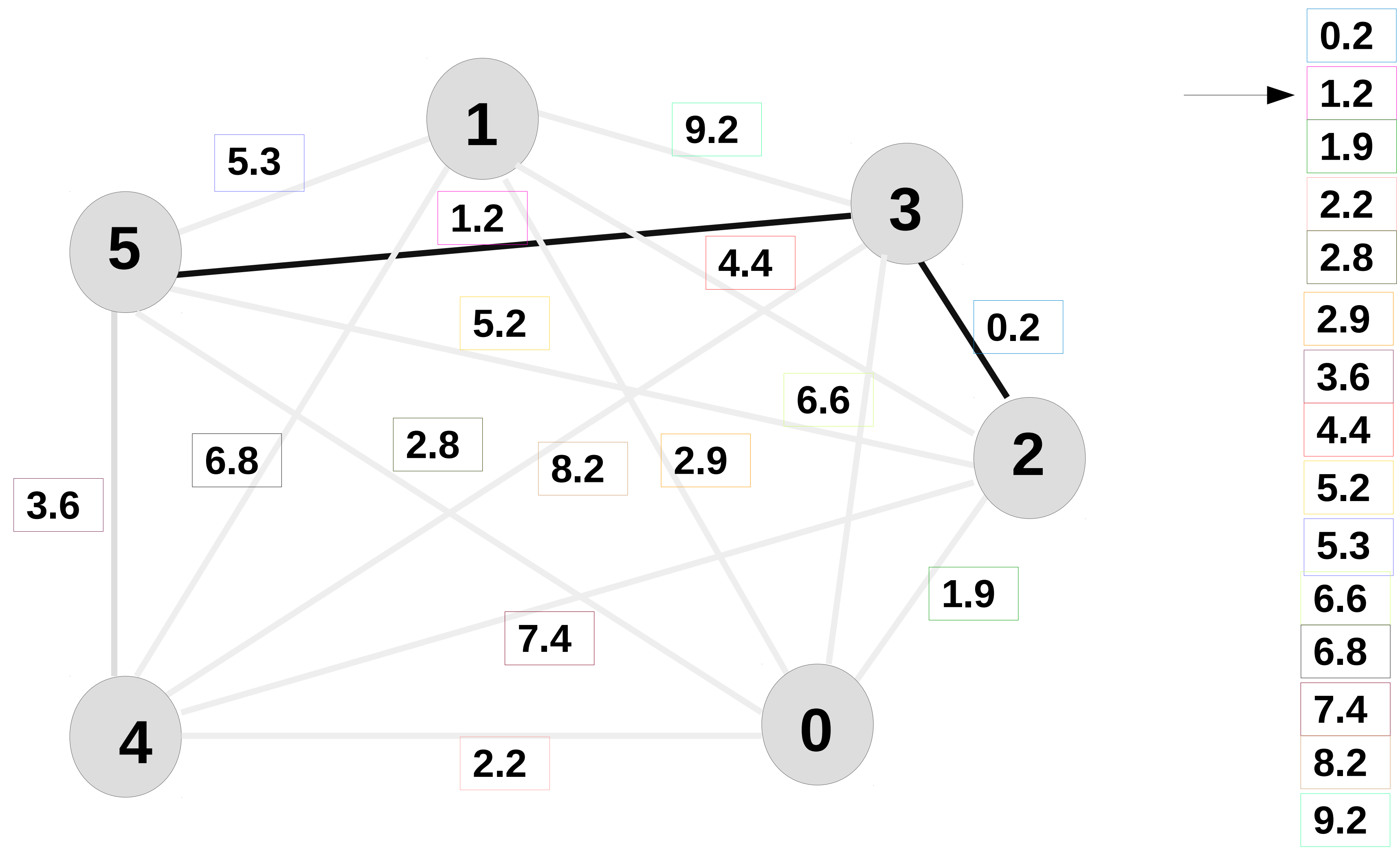
PRIMJER – rad algoritma

3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



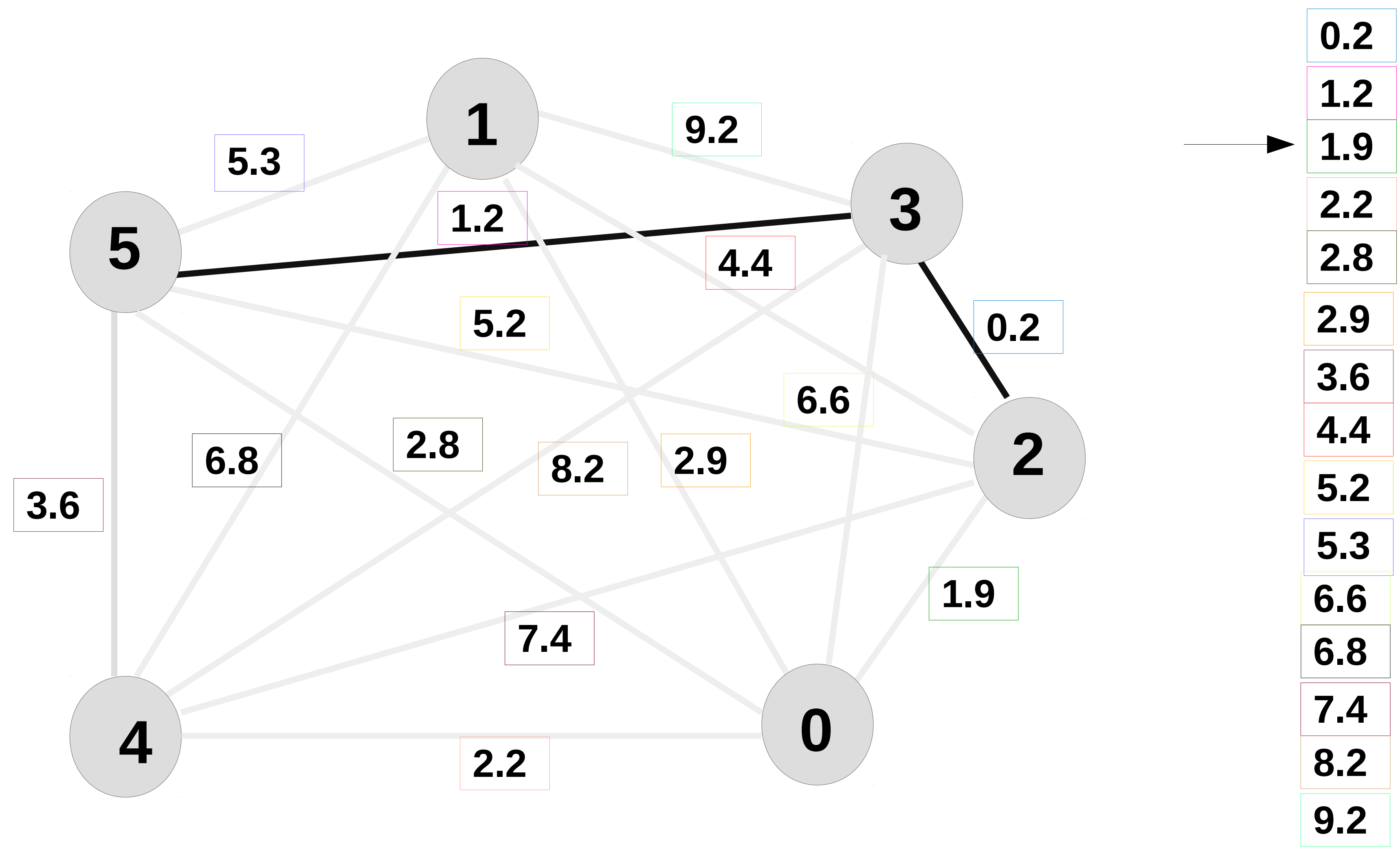
PRIMJER – rad algoritma

3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



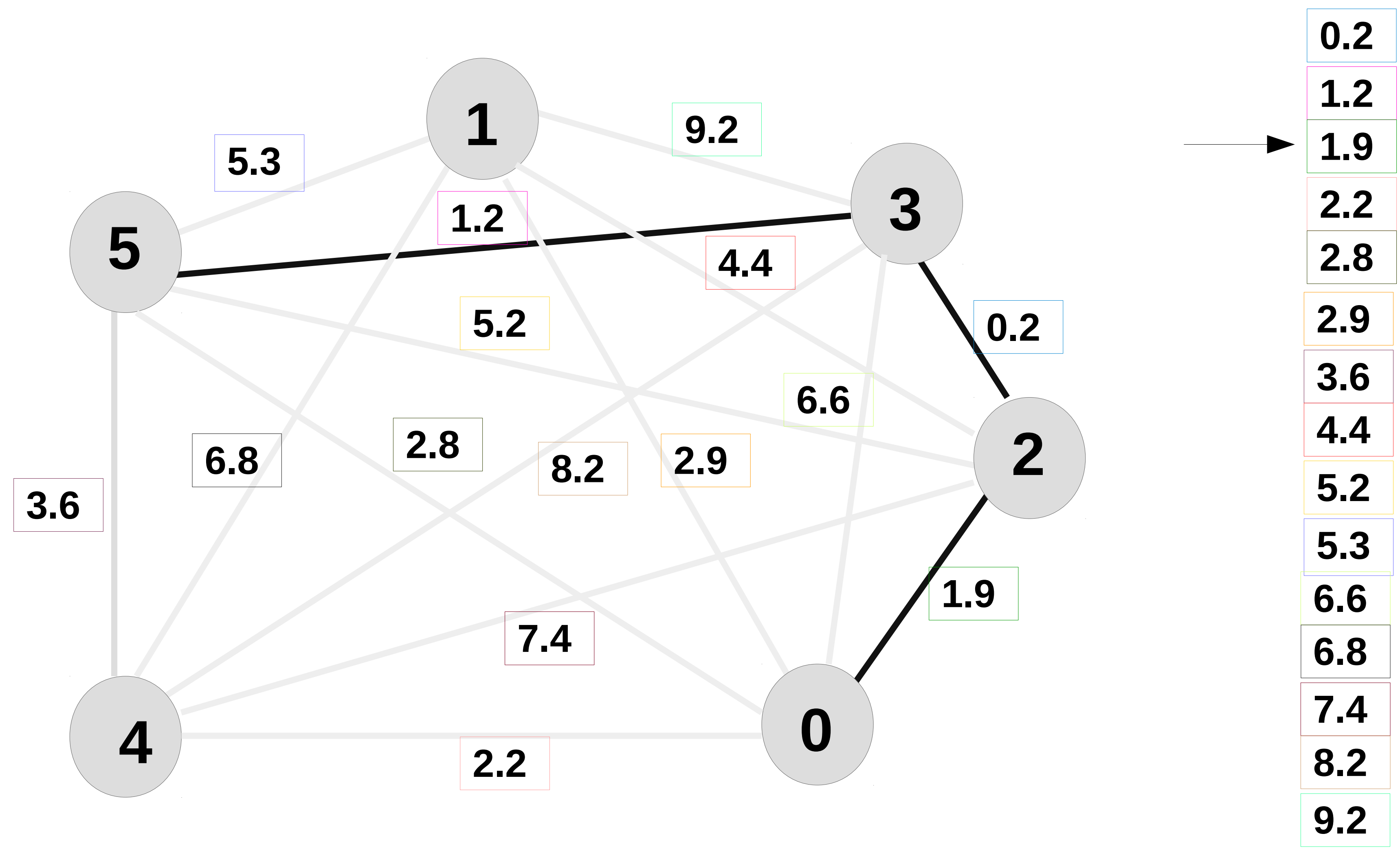
PRIMJER – rad algoritma

3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



PRIMJER – rad algoritma

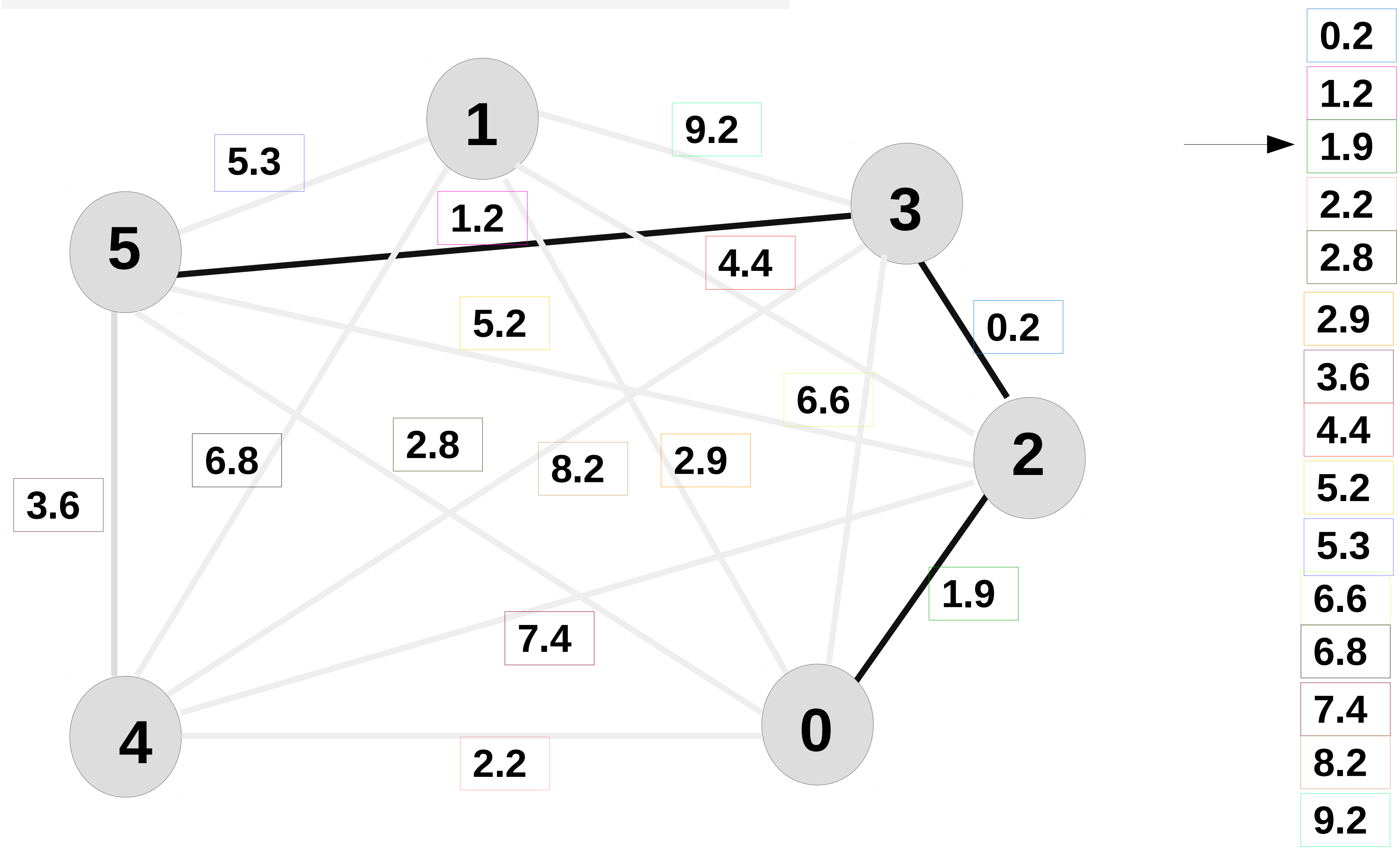
3. Kruskalov algoritam dok ne nastanu 3 komponente povezanosti (3 klastera)



PRIMJER – rad algoritma

4. STOP!

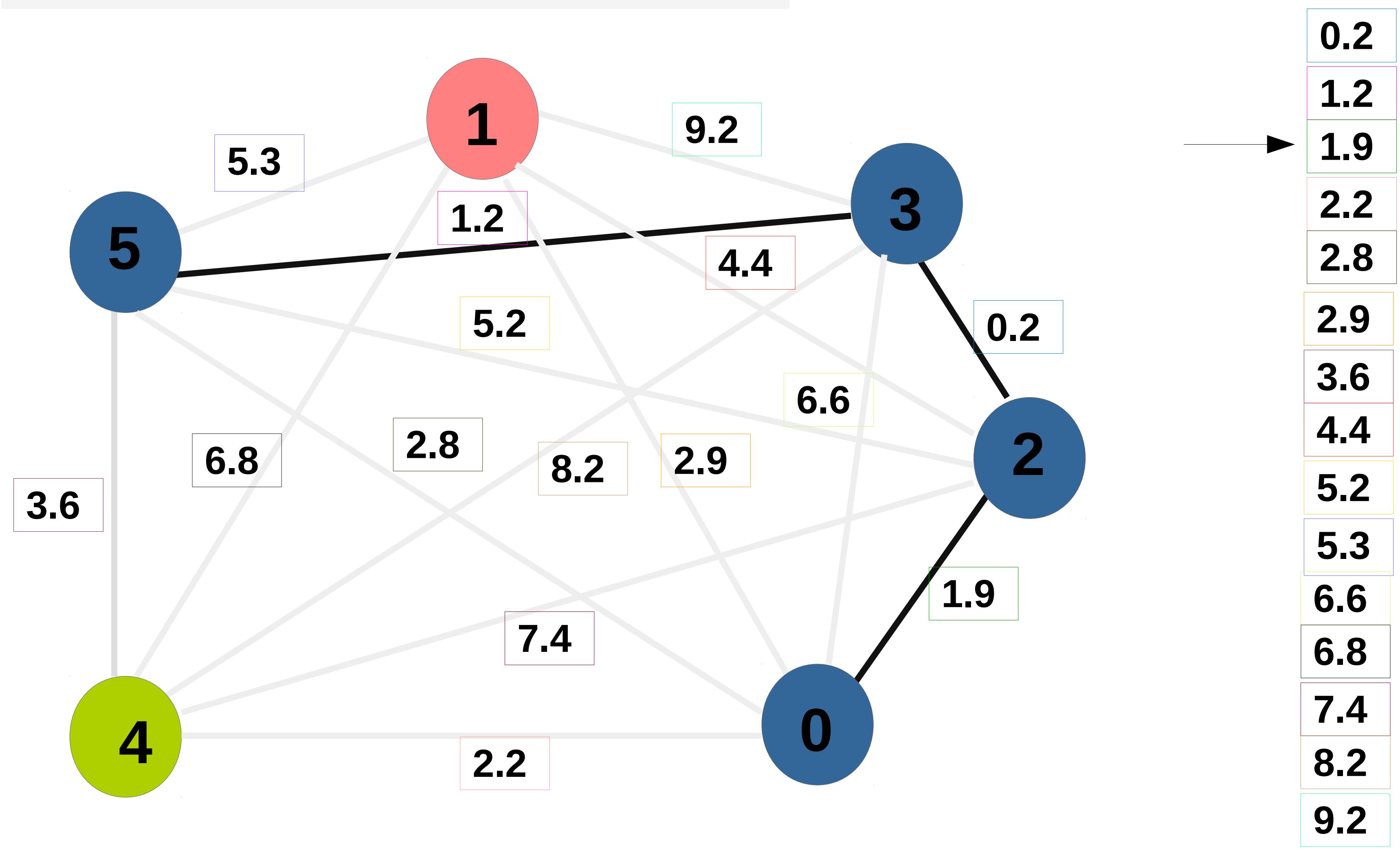
3 komponente povezanosti!



PRIMJER – rad algoritma

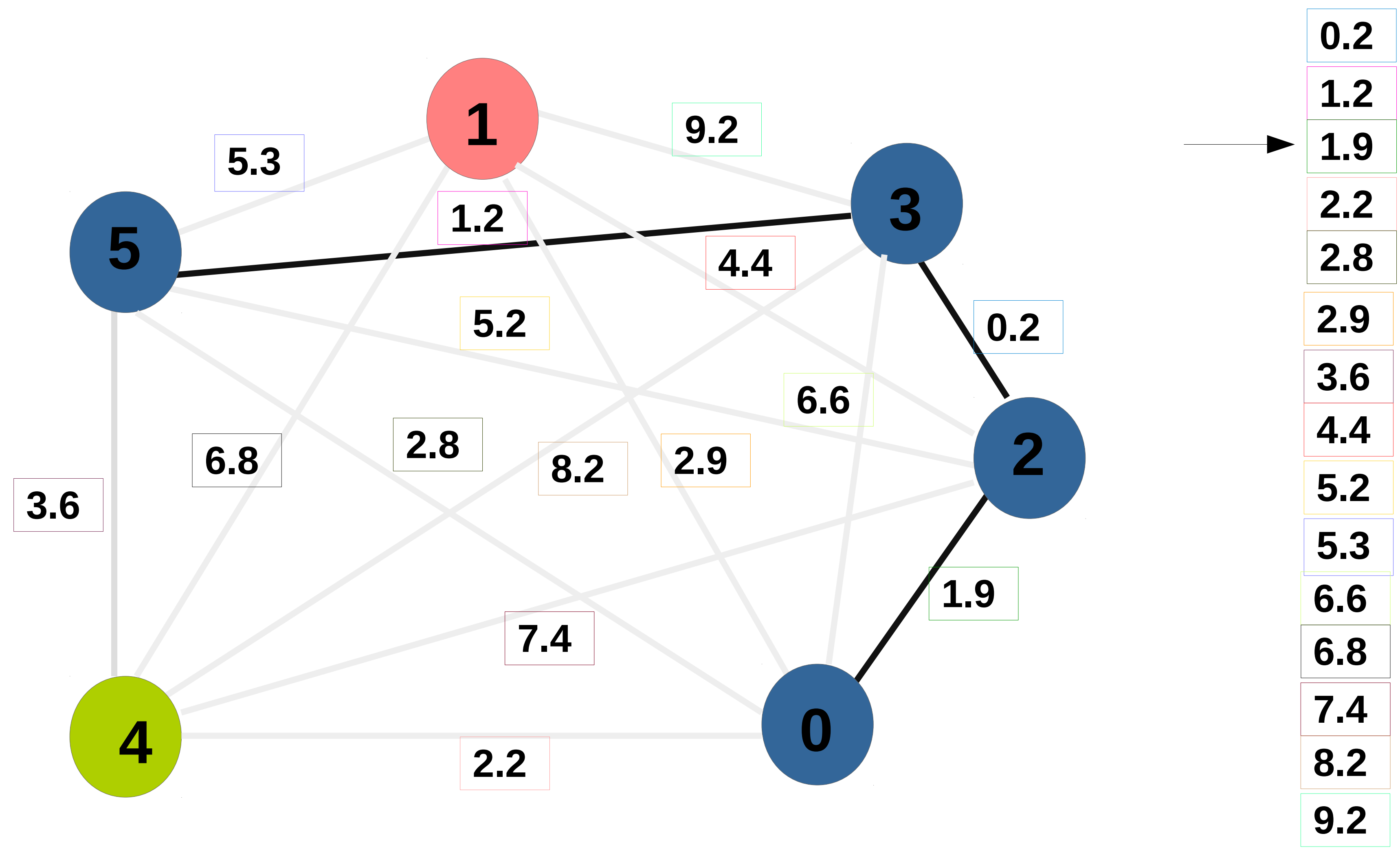
4. STOP!

3 komponente povezanosti!



PRIMJER – rad algoritma

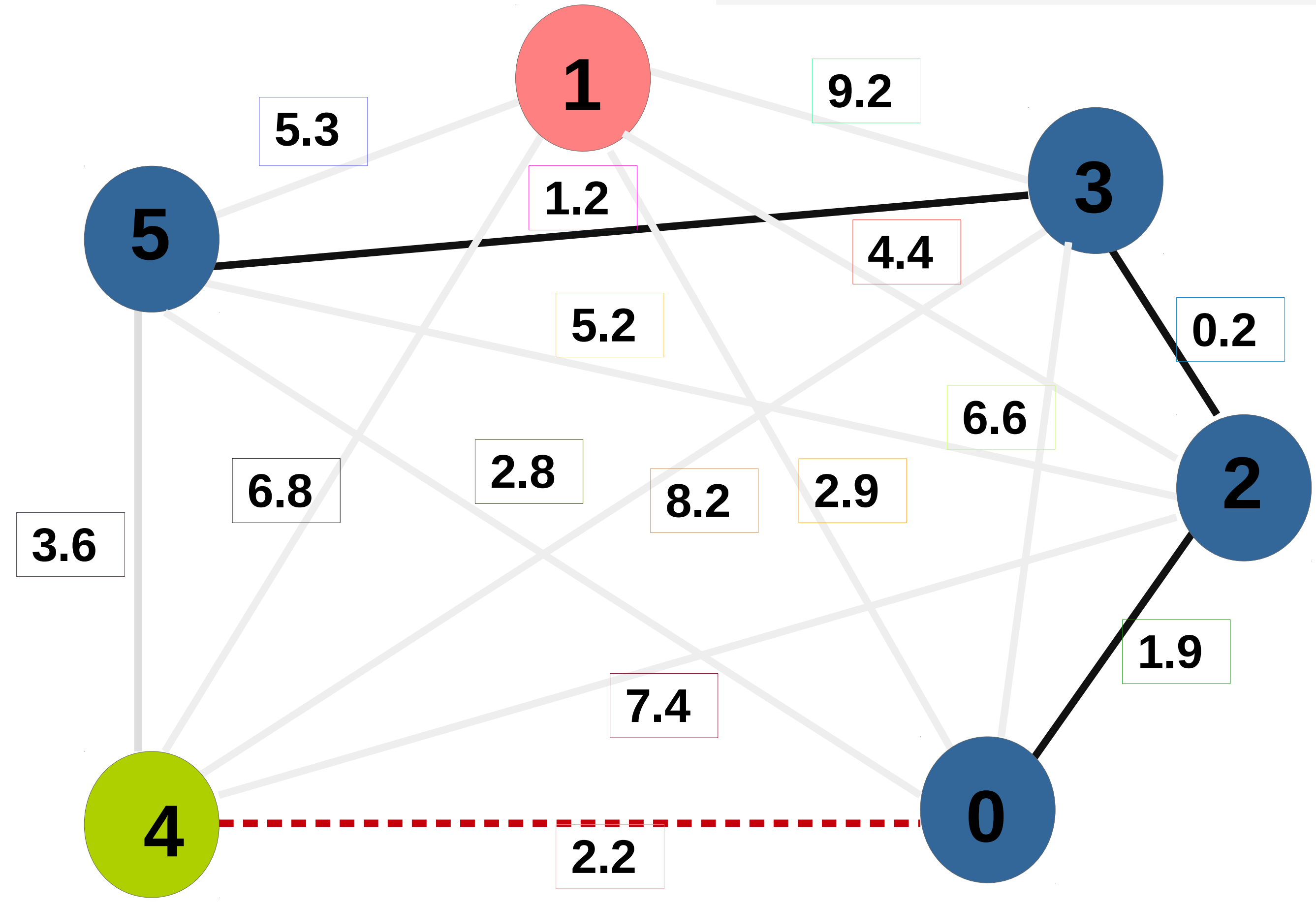
Razmak? (minimalna udaljenost između bilo koja dva vrha u različitim klasterima)

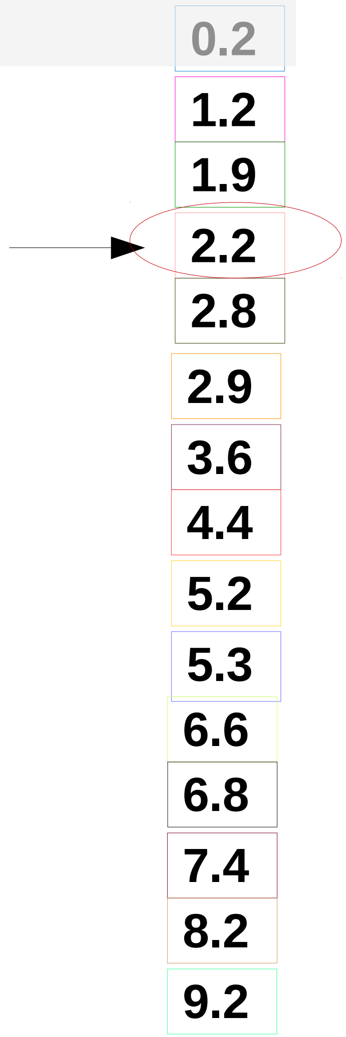


PRIMJER – rad algoritma

Razmak? (minimalna udaljenost između bilo koja dva vrha u različitim klasterima)

Sljedeći brid koji bi Kruskalov algoritam dodao!





KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- neka $\mathcal{C}$ označava k-klasteriranje $C_1,\ldots,C_k$ dobiveno Kruskalovim algoritmom

KRUSKALOV ALGORITAM

Primjena u klasteriranju

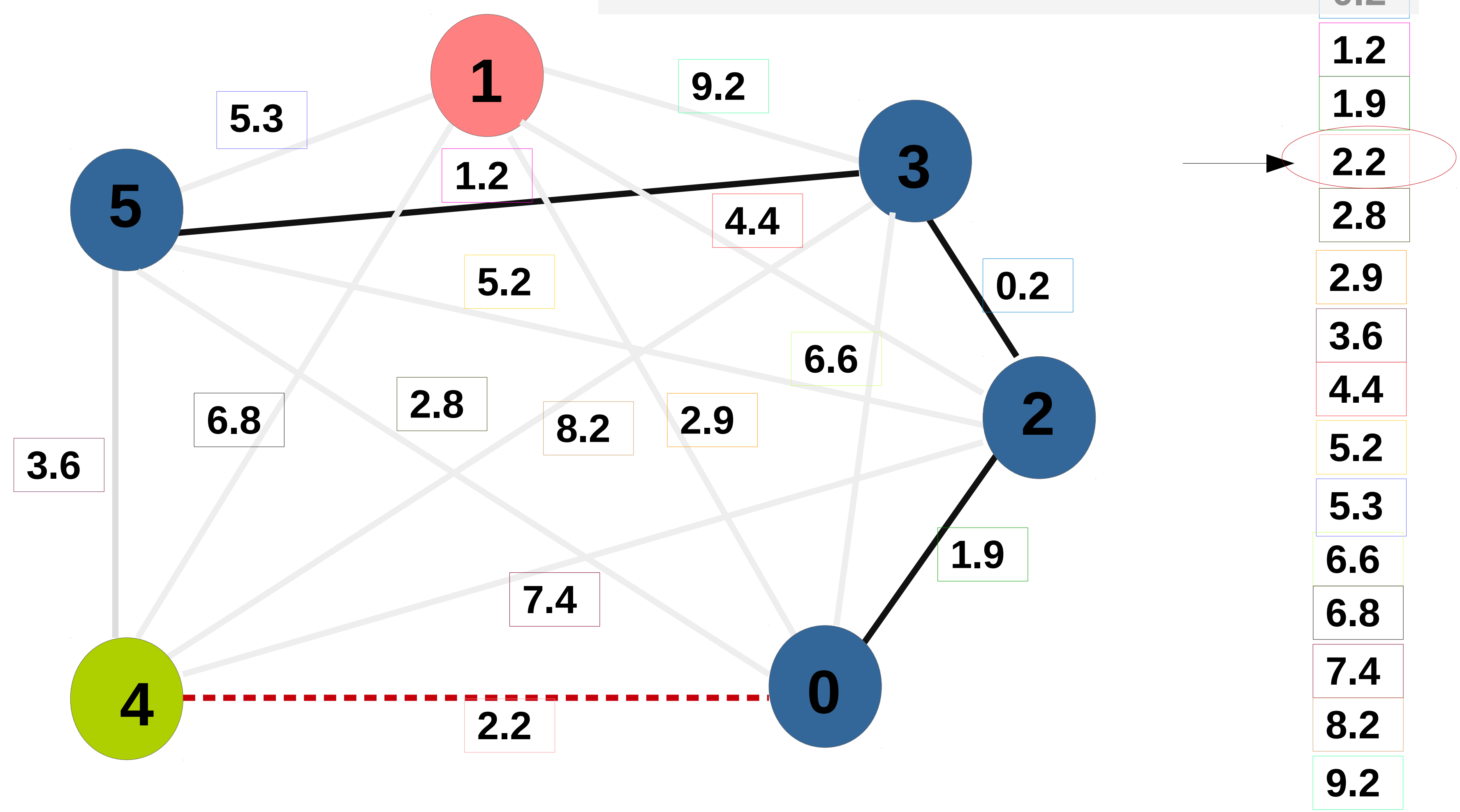
Dokaz

- neka $\mathcal{C}$ označava k-klasteriranje $C_1,\ldots,C_k$ dobiveno Kruskalovim algoritmom
- tada je razmak (*spacing*) u $\mathcal{C}$ upravo duljina (k – 1). najtežeg brida (d*)

PRIMJER – rad algoritma

Razmak? (minimalna udaljenost između bilo koja dva vrha u različitim klasterima)

Sljedeći brid koji bi Kruskalov algoritam dodao!



KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- neka $\mathcal{C}$ označava k-klasteriranje $C_1,\ldots,C_k$ dobiveno Kruskalovim algoritmom
- tada je razmak (*spacing*) u $\mathcal{C}$ upravo duljina (k – 1). najtežeg brida (d*)
- pretpostavimo da je $\mathcal{C}^*$ $(C^*_1,\ldots,C^*_k)$ neko drugo k – klasteriranje različito od $\mathcal{C}$

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

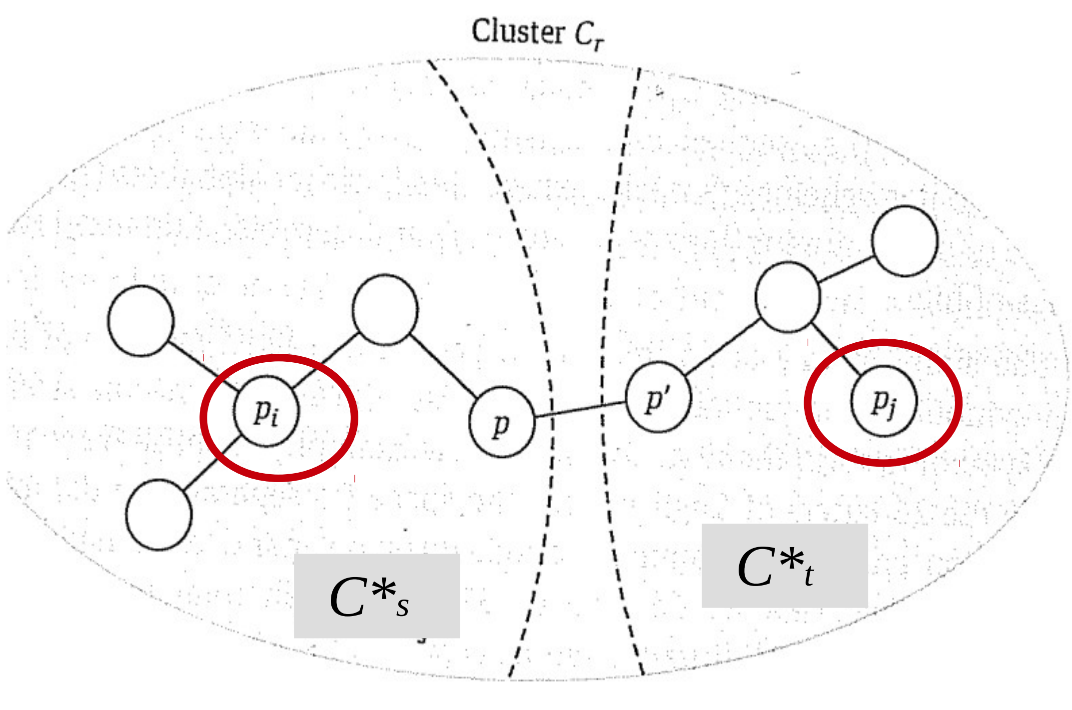
- neka $\mathcal{C}$ označava k-klasteriranje $C_1,\ldots,C_k$ dobiveno Kruskalovim algoritmom
- tada je razmak (*spacing*) u $\mathcal{C}$ upravo duljina (k – 1). najtežeg brida (d*)
- pretpostavimo da je $\mathcal{C}^*$ $(C^*_1,\ldots,C^*_k)$ neko drugo k – klasteriranje različito od $\mathcal{C}$
- tada mora postojati jedan od klastera C_r koji nije podskup niti jednog klastera u $\mathcal{C}^*$ $\rightarrow$ postoje točke p_i *i* p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t

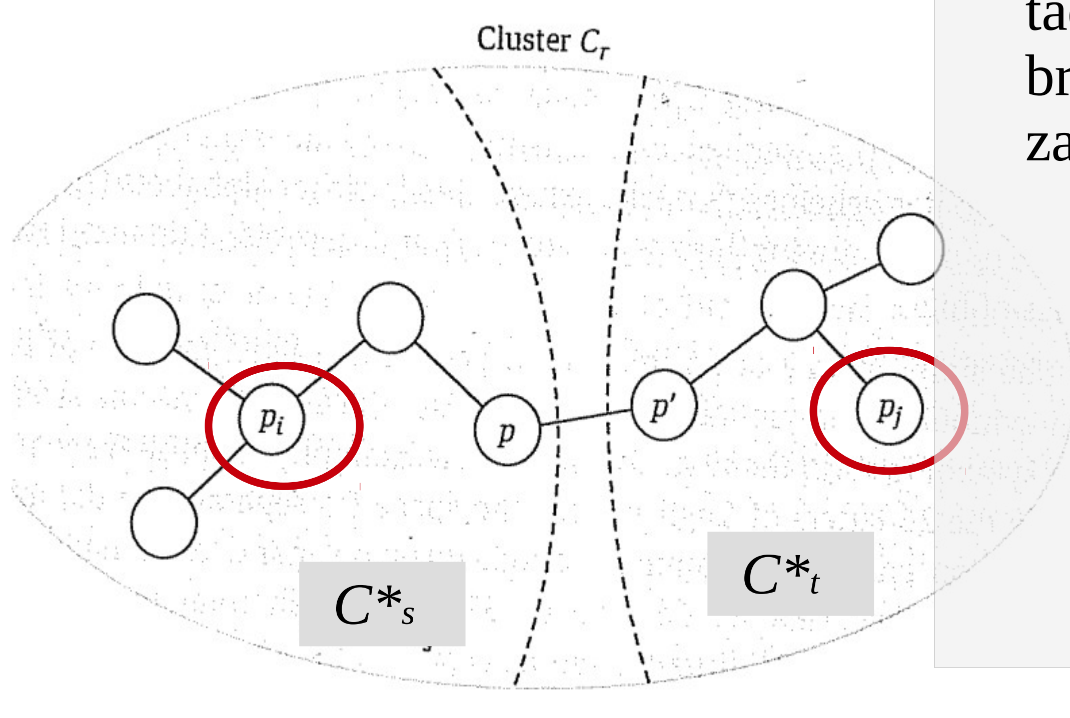


KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u C^* (BSO) u redom C^*_s i C^*_t



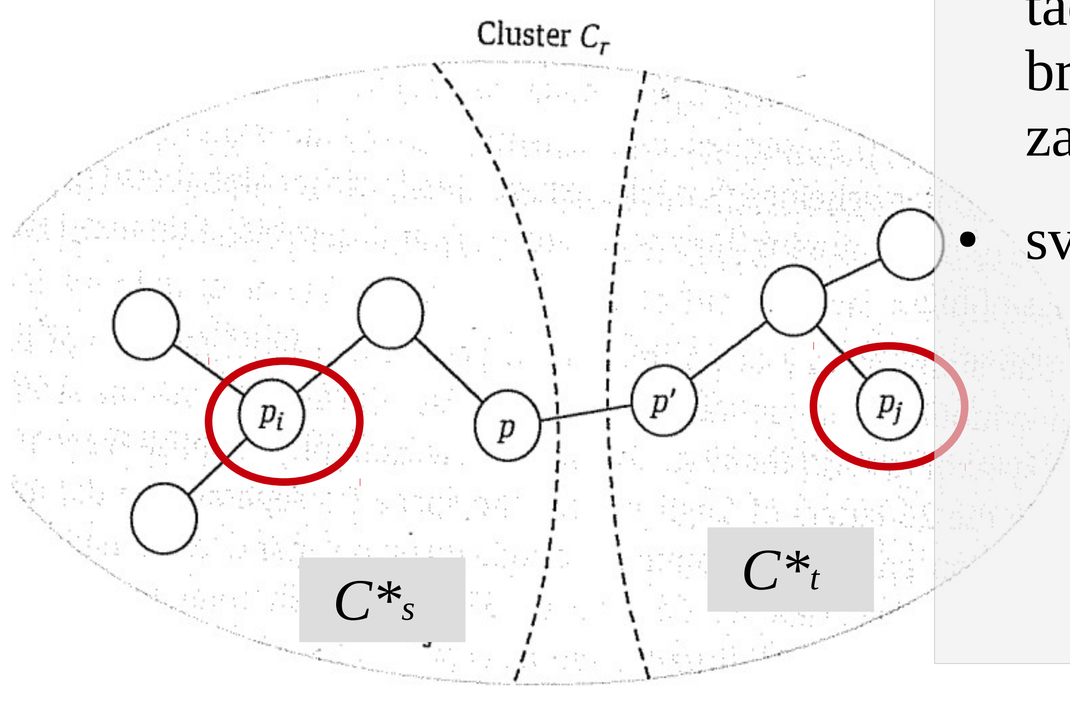
- jer p_i i p_j pripadaju istoj komponenti Cr → tada je Kruskalov algoritam dodao sve bridove na putu P od p_i do p_j prije zaustavljanja

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t



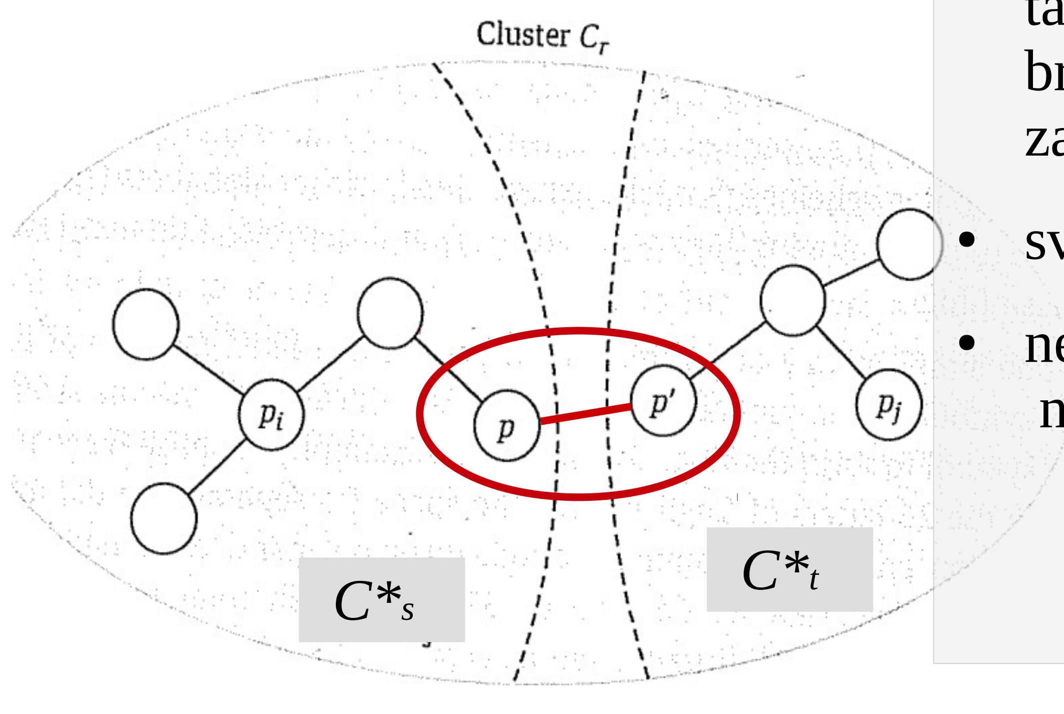
- jer p_i i p_j pripadaju istoj komponenti Cr → tada je Kruskalov algoritam dodao sve bridove na putu P od p_i do p_j prije zaustavljanja
- svaki brid na tom putu je duljine najviše d*

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t



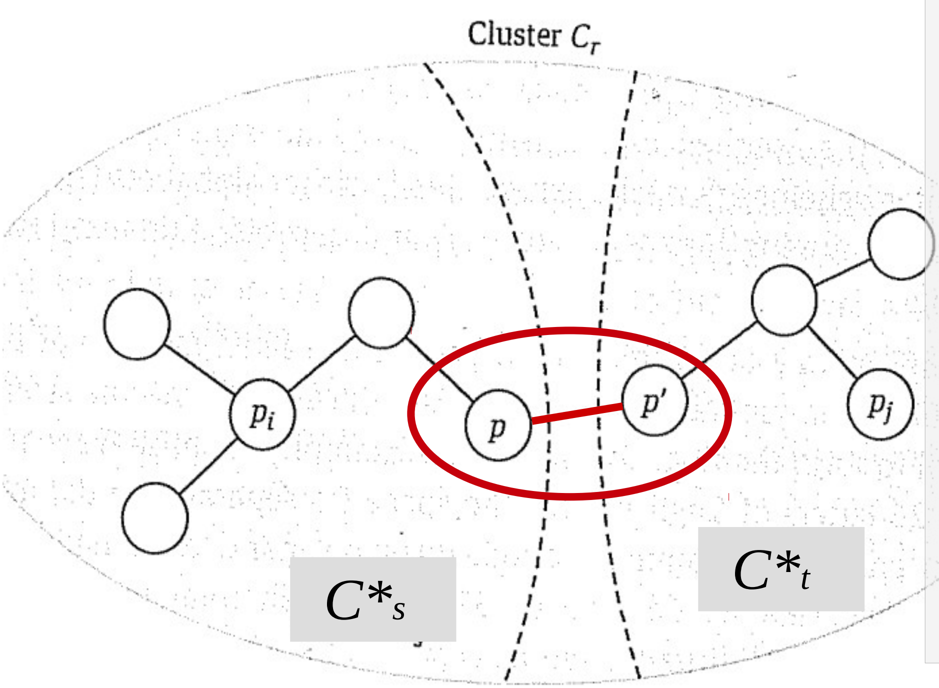
- jer p_i i p_j pripadaju istoj komponenti Cr → tada je Kruskalov algoritam dodao sve bridove na putu P od p_i do p_j prije zaustavljanja
- svaki brid na tom putu je duljine najviše d*
- neka je p prvi vrh u P koji ne pripada C^*t i neka je p' prvi takav koji ne pripada C^*s

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t



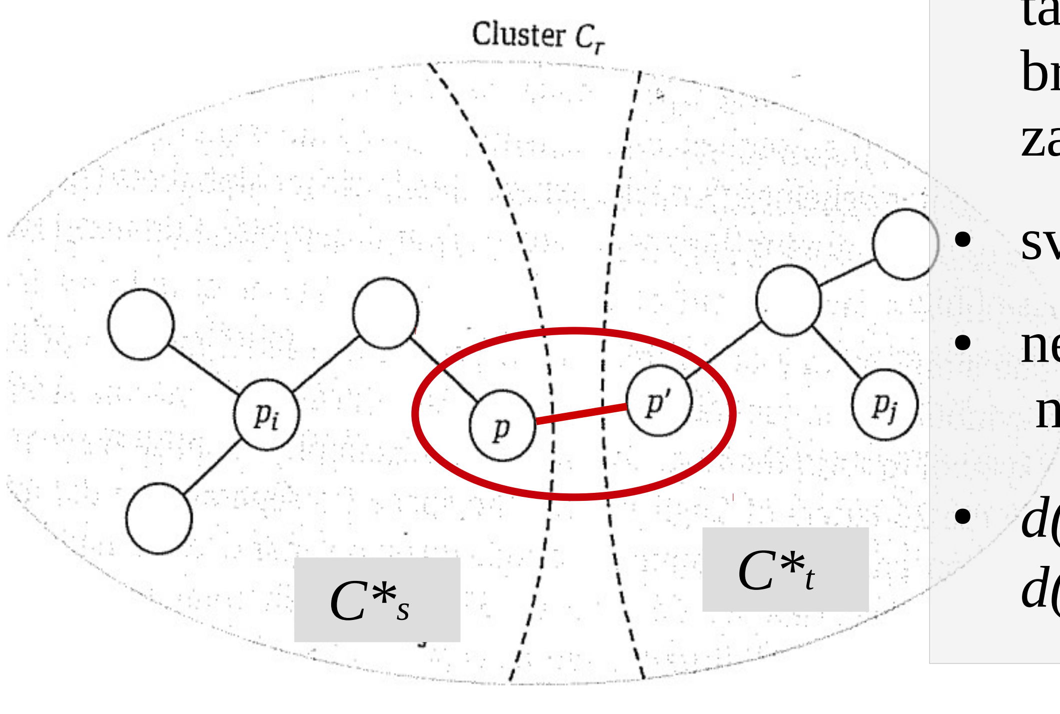
- jer p_i i p_j pripadaju istoj komponenti Cr → tada je Kruskalov algoritam dodao sve bridove na putu P od p_i do p_j prije zaustavljanja
- svaki brid na tom putu je duljine najviše d*
- neka je p prvi vrh u P koji ne pripada C^*t i neka je p' prvi takav koji ne pripada C^*s
- $d(p,p') \leq d^*$ → razmak u $\mathcal{C}^*$ je najviše $d(p,p') \leq d^*$

KRUSKALOV ALGORITAM

Primjena u klasteriranju

Dokaz

- p_i i p_j u C_r koje pripadaju **različitim** klasterima u $\mathcal{C}^*$ (BSO) u redom C^*_s i C^*_t



- jer p_i i p_j pripadaju istoj komponenti Cr → tada je Kruskalov algoritam dodao sve bridove na putu P od p_i do p_j prije zaustavljanja
- svaki brid na tom putu je duljine najviše d*
- neka je p prvi vrh u P koji ne pripada C^*t i neka je p' prvi takav koji ne pripada C^*s
- $d(p,p') \leq d^*$ → razmak u $\mathcal{C}^*$ je najviše $d(p,p') \leq d^*$

$\mathcal{Q}.\mathcal{E}.\mathcal{D}.$

STRUKTURA PODATAKA

Union Find

STRUKTURA PODATAKA

Union Find

- omogućava efikasnu implementaciju komponenata povezanosti u grafu

STRUKTURA PODATAKA

Union Find

- omogućava efikasnu implementaciju komponenata povezanosti u grafu

Karakteristike:

STRUKTURA PODATAKA

Union Find

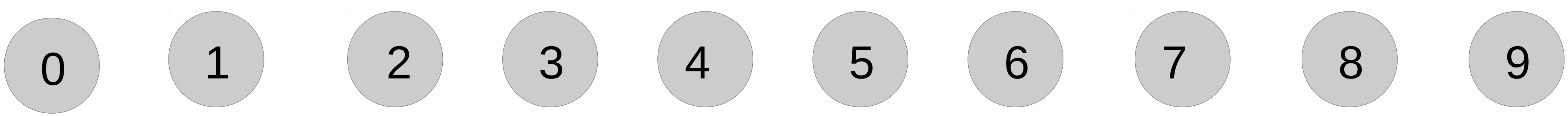
- omogućava efikasnu implementaciju komponenenata povezanosti u grafu

Karakteristike:

- **Find(u)** → za dani vrh u vraća ime komponente u kojoj se vrh u nalazi (**korijen komponente**)
- **Union(A,B)** → spaja dva skupa A i B u jedan skup
- **poboljšanje strukture** → Union(A,B) spaja kraći skup s većim

STRUKTURA PODATAKA

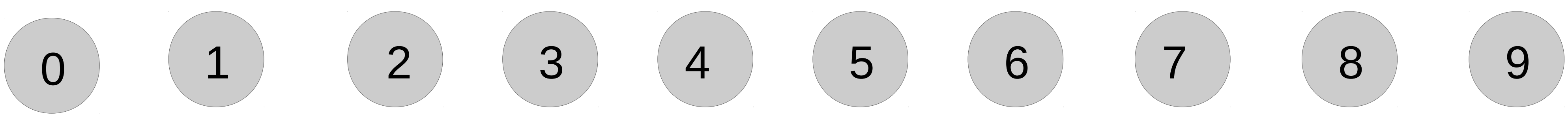
Union Find



STRUKTURA PODATAKA

Union Find

Union(4,3)



STRUKTURA PODATAKA

Union Find

Union(4,3)

STRUKTURA PODATAKA

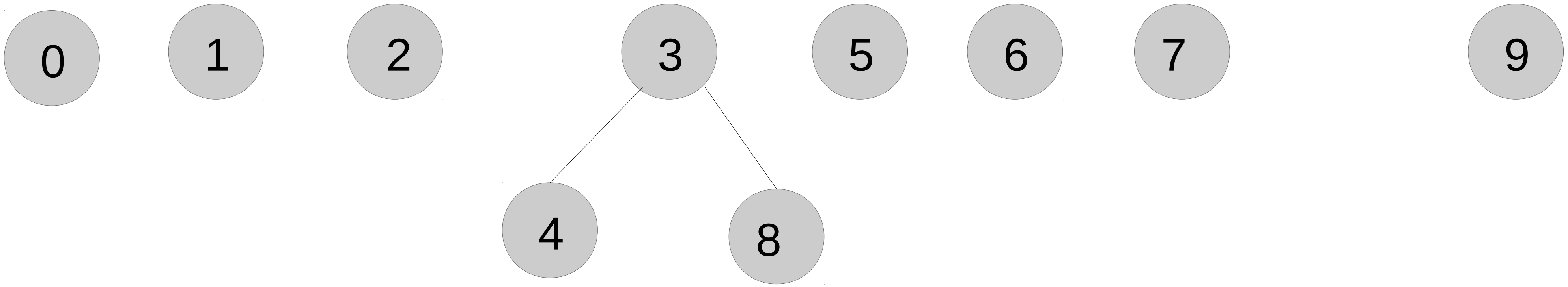
Union Find

Union(3,8)

STRUKTURA PODATAKA

Union Find

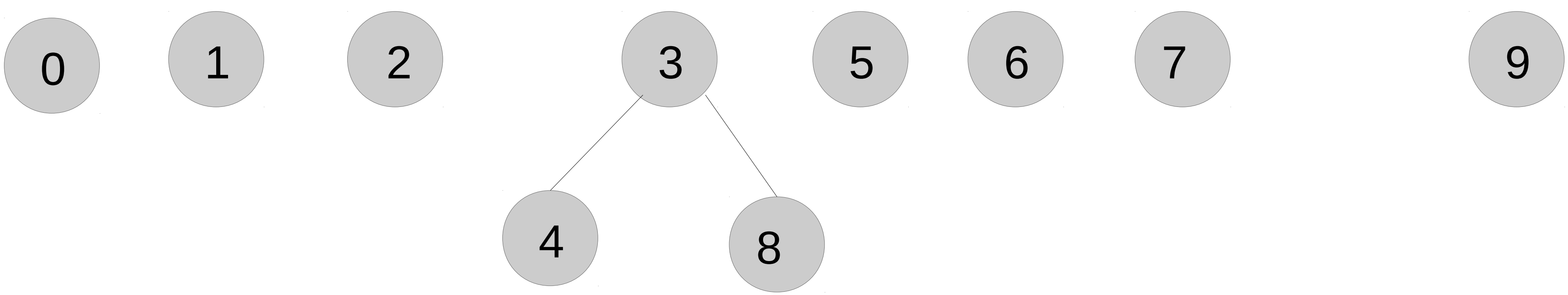
Union(3,8)



STRUKTURA PODATAKA

Union Find

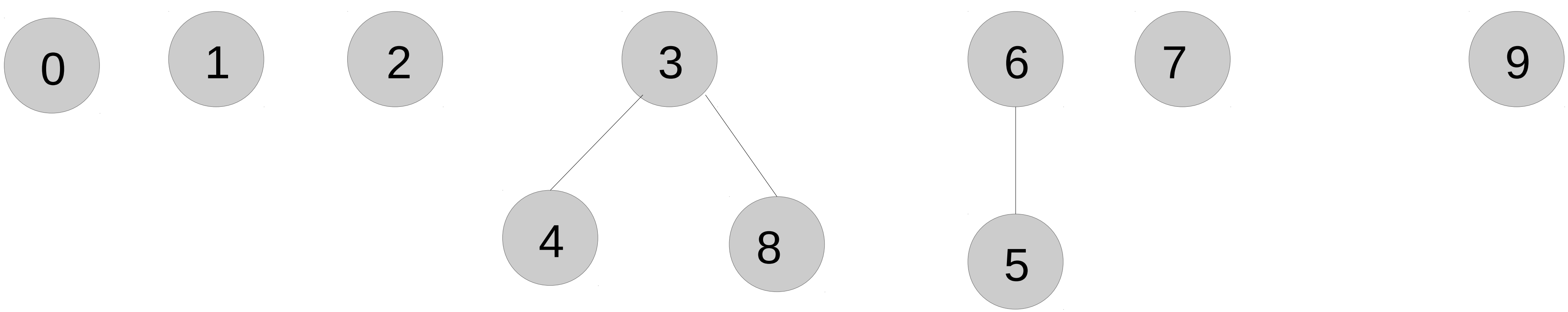
Union(6,5)



STRUKTURA PODATAKA

Union Find

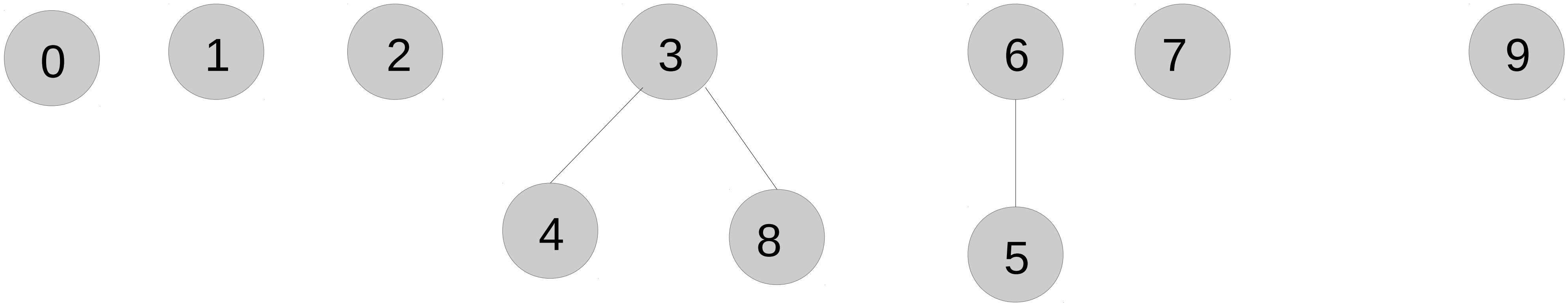
Union(6,5)



STRUKTURA PODATAKA

Union Find

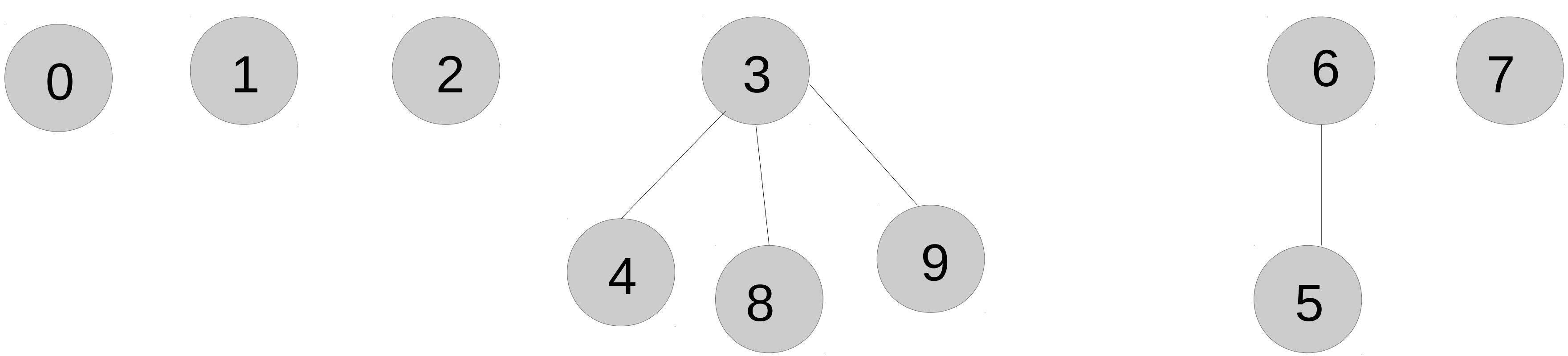
Union(9,4)



STRUKTURA PODATAKA

Union Find

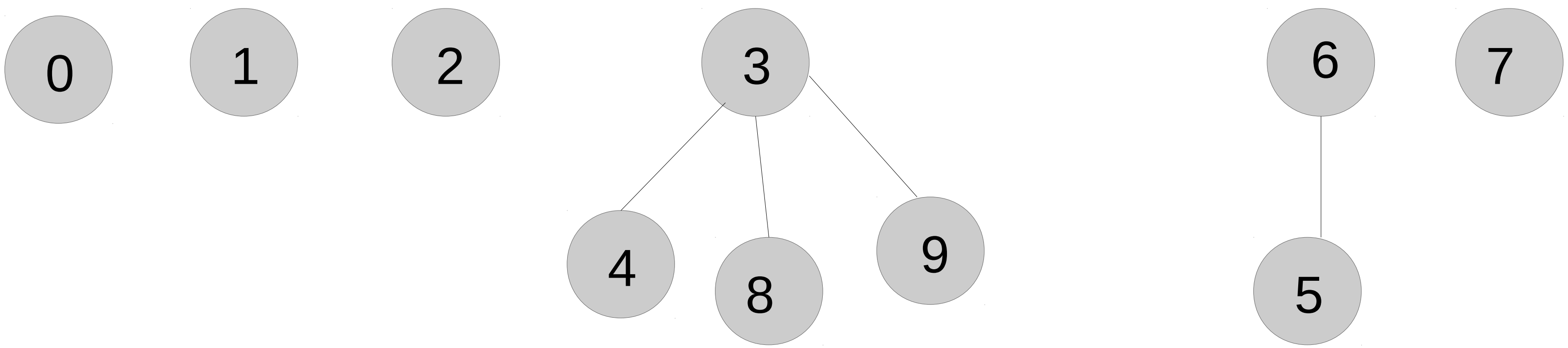
Union(9,4)



STRUKTURA PODATAKA

Union Find

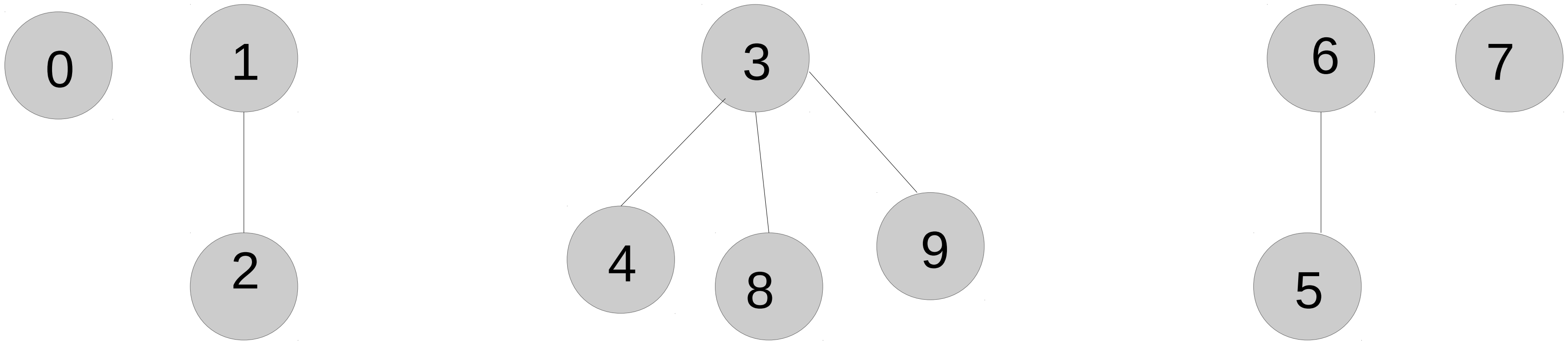
Union(1,2)



STRUKTURA PODATAKA

Union Find

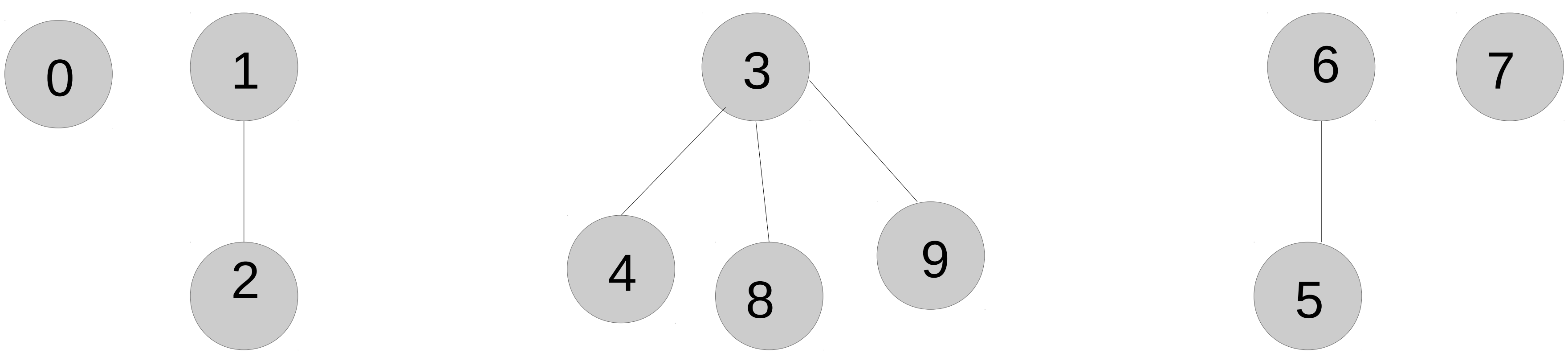
Union(1,2)



STRUKTURA PODATAKA

Union Find

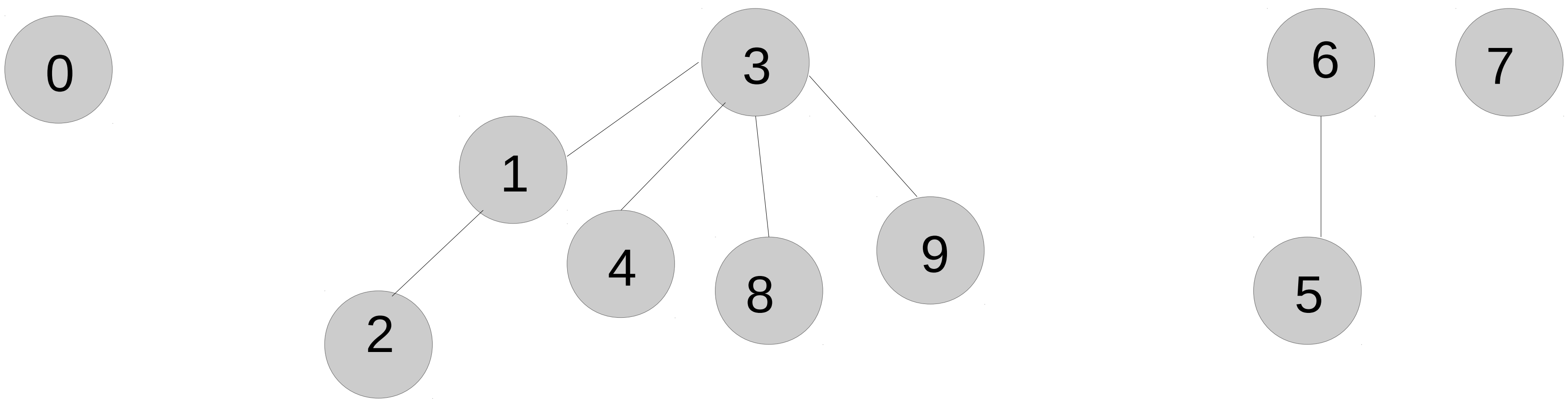
Union(2,3)



STRUKTURA PODATAKA

Union Find

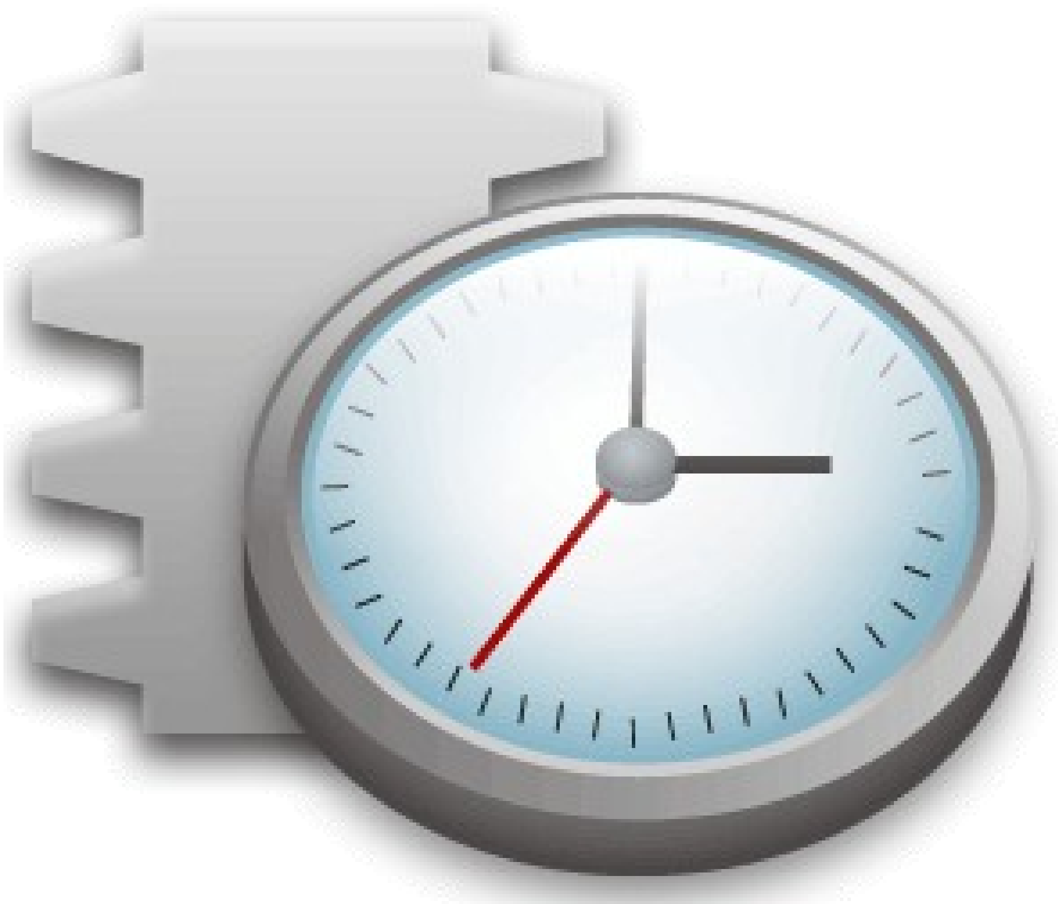
Union(2,3)



STRUKTURA PODATAKA

Union Find

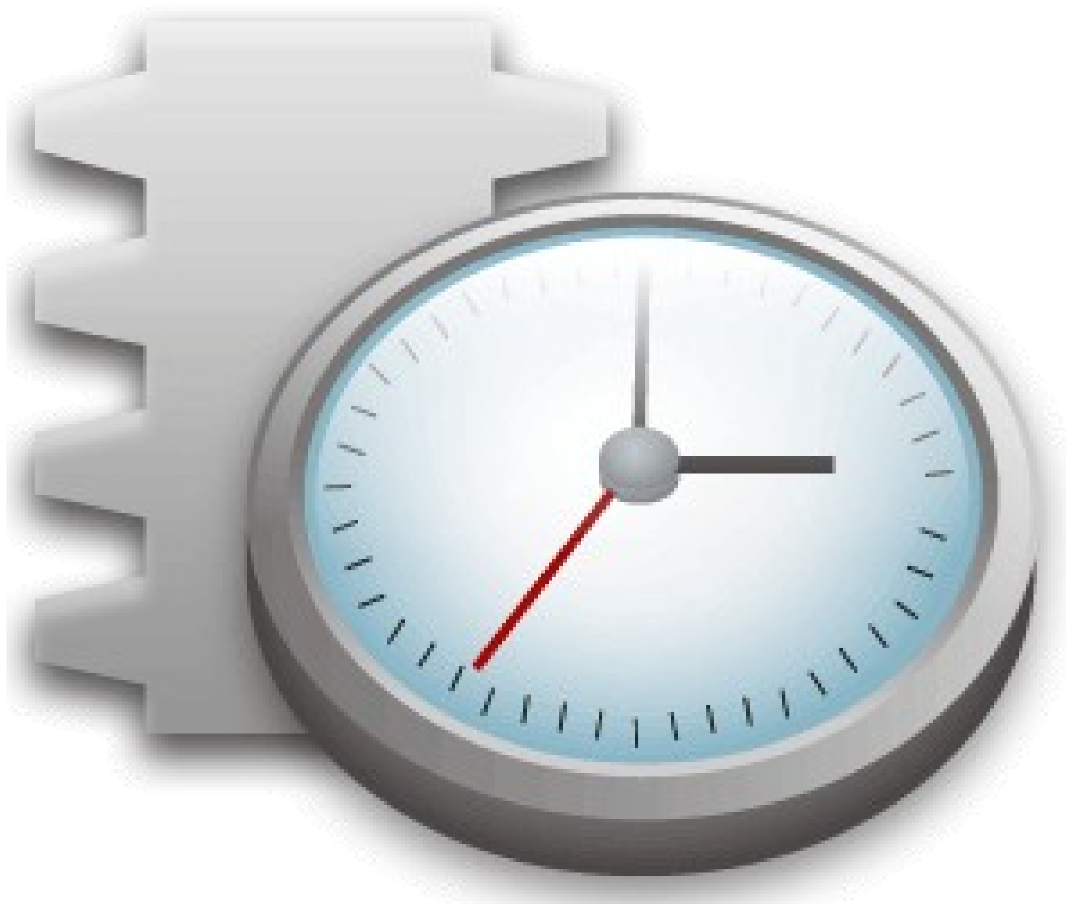
SLOŽENOST?



STRUKTURA PODATAKA

Union Find

SLOŽENOST?

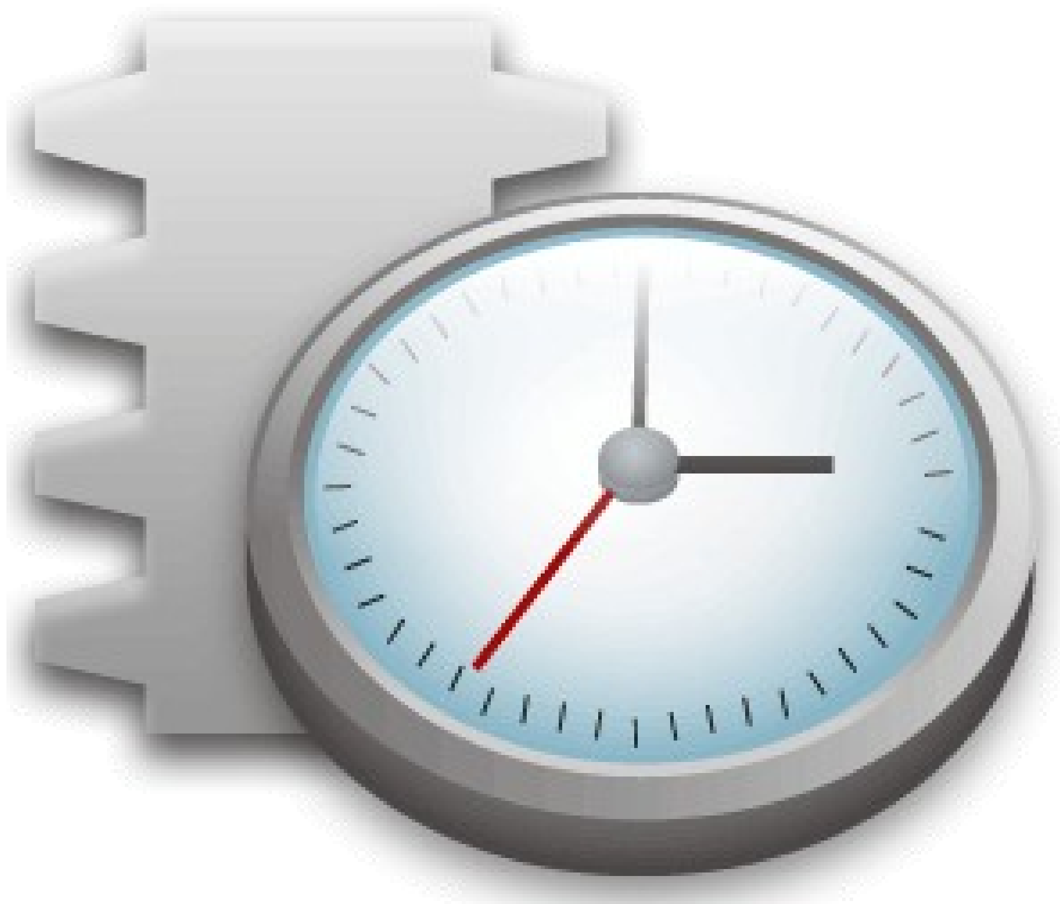


- Union(A,B) → **O(1)**

STRUKTURA PODATAKA

Union Find

SLOŽENOST?



- Union(A,B) → **O(1)**
- Find(u) → **O(logn)**

STRUKTURA PODATAKA

`Union Find` → primjena u Kruskalovom algoritmu

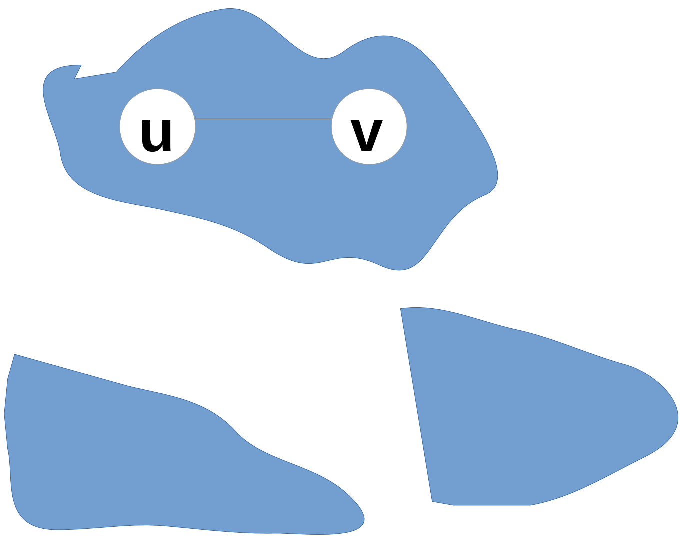
- problem zatvara li brid *(u,v)* ciklus se svodi na provjeru pripadaju li u i v istoj komponenti povezanosti
- *Find(u) == Find(v)?*

STRUKTURA PODATAKA

Union Find → primjena u Kruskalovom algoritmu

- problem zatvara li brid *(u,v)* ciklus se svodi na provjeru pripadaju li u i v istoj komponenti povezanosti
- *Find(u) == Find(v)?*

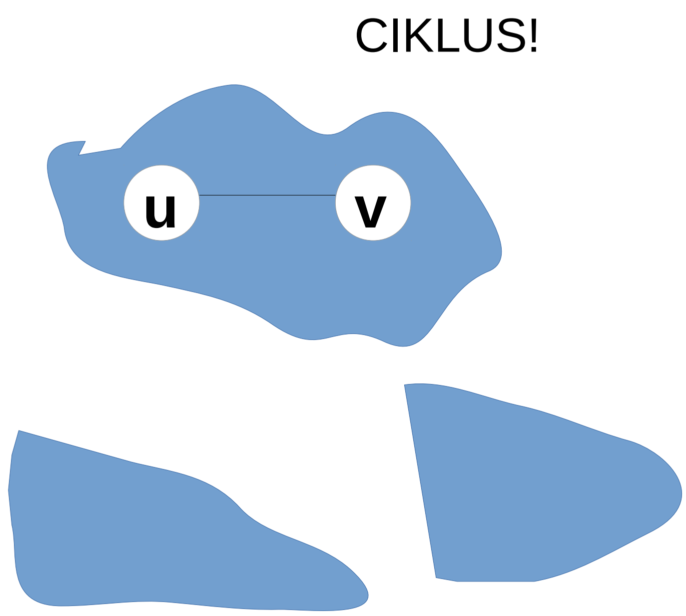
CIKLUS!

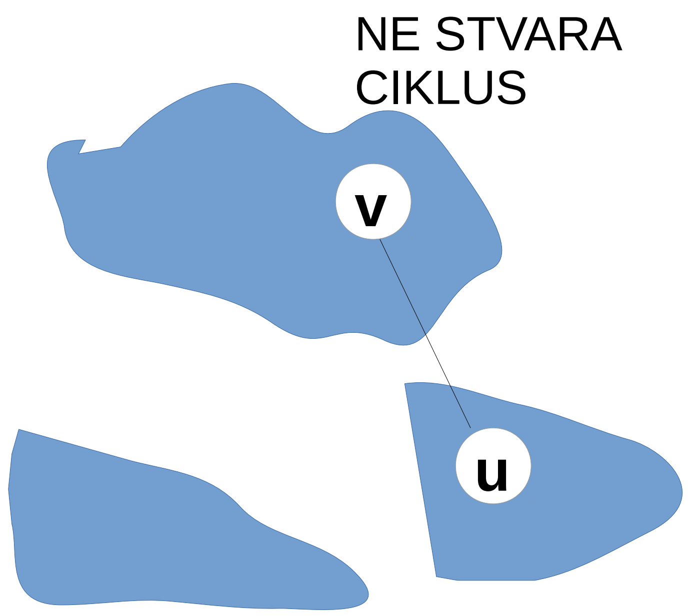


STRUKTURA PODATAKA

`Union Find` → primjena u Kruskalovom algoritmu

- problem zatvara li brid *(u,v)* ciklus se svodi na provjeru pripadaju li u i v istoj komponenti povezanosti
- *Find(u) == Find(v)?*



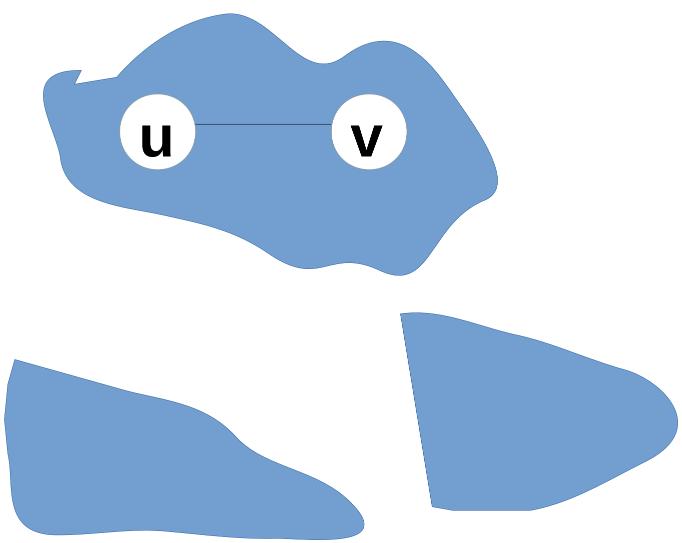


STRUKTURA PODATAKA

`Union Find` → primjena u Kruskalovom algoritmu

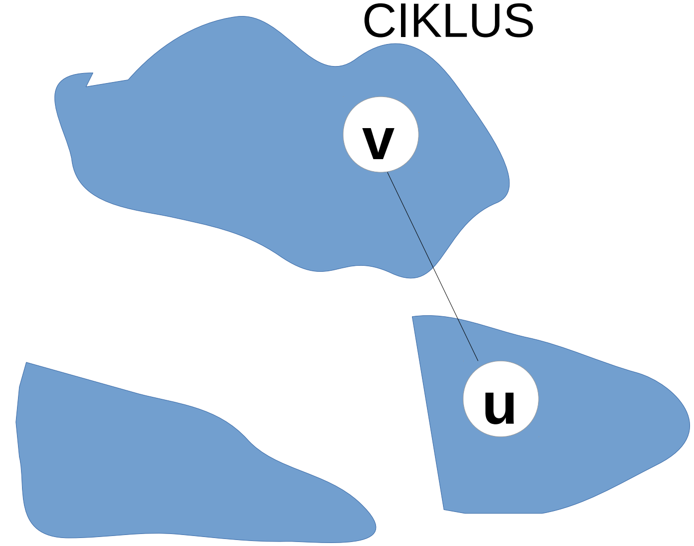
- problem zatvara li brid *(u,v)* ciklus se svodi na provjeru pripadaju li u i v istoj komponenti povezanosti
- *Find(u) == Find(v)?*

CIKLUS!



O(logn)!

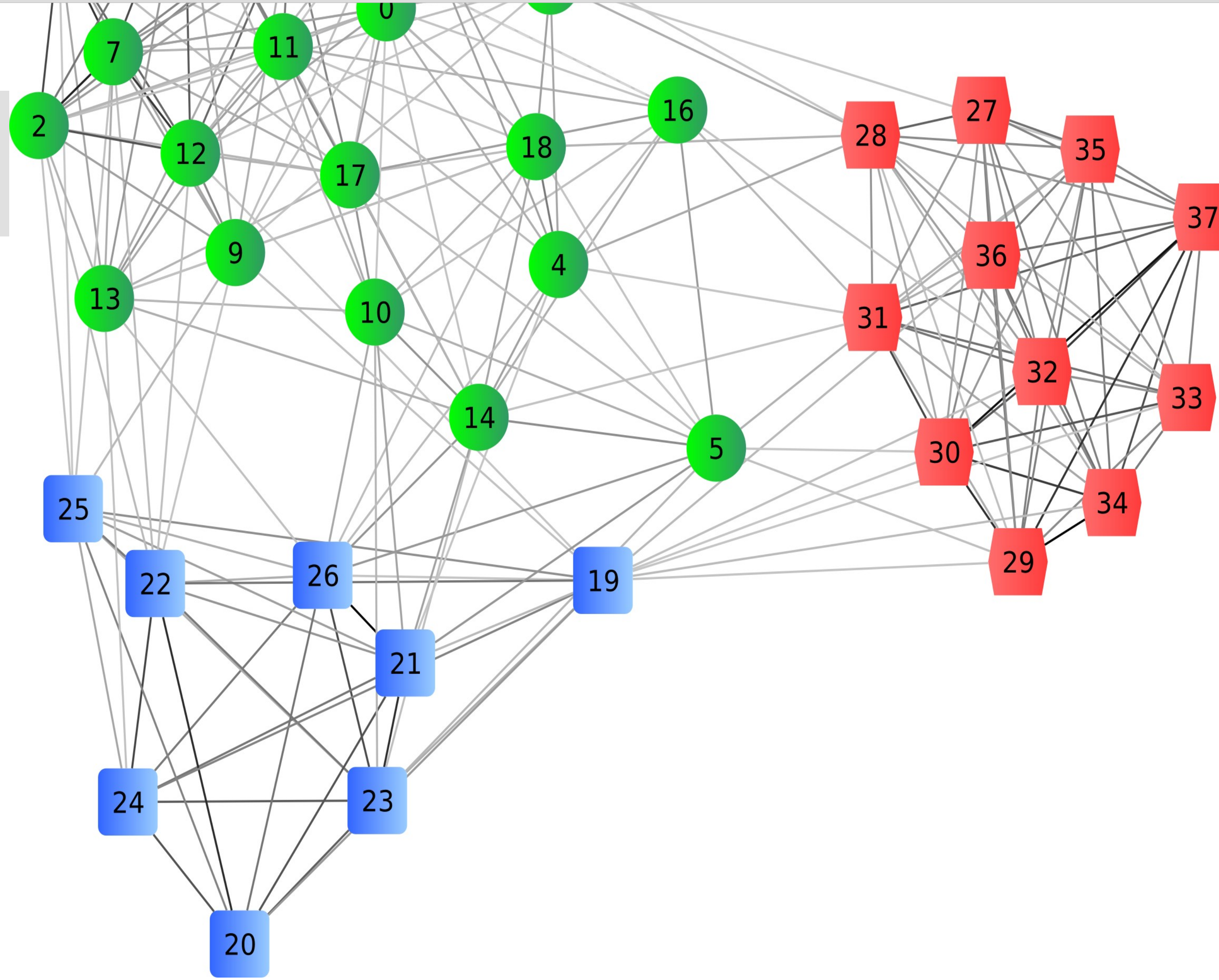
NE STVARA CIKLUS



STRUKTURA PODATAKA

Analiza složenosti k - klasteriranja

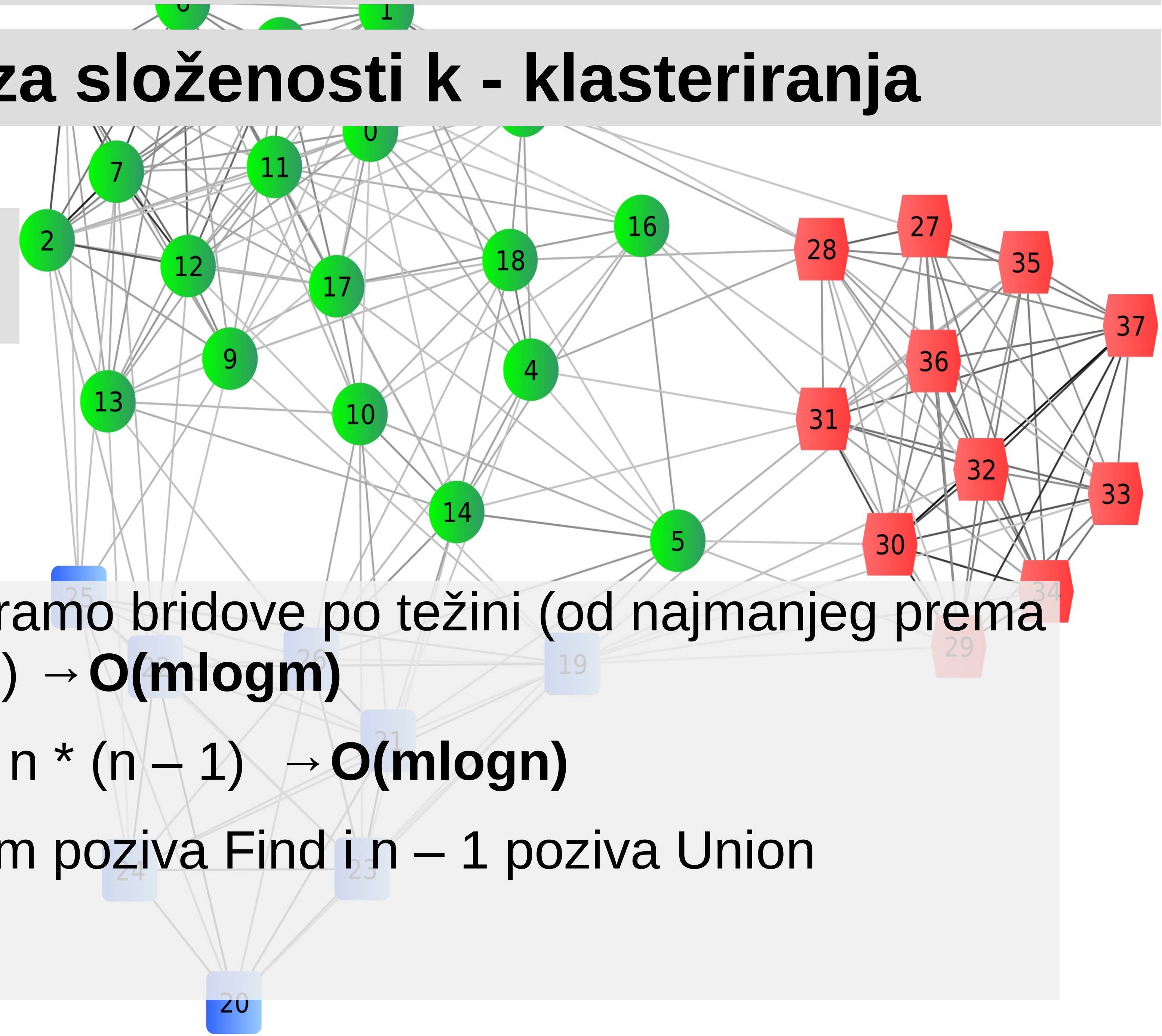
- **m** → broj bridova
- **n** → broj vrhova



STRUKTURA PODATAKA

Analiza složenosti k - klasteriranja

- **m** →broj bridova
- **n** → broj vrhova



- prvo sortiramo bridove po težini (od najmanjeg prema najvećem) →**O(mlogm)**
- jer je m < n * (n – 1) →**O(mlogn)**
- ukupno 2m poziva Find i n – 1 poziva Union operacija

STRUKTURA PODATAKA

Analiza složenosti k - klasteriranja

- **m** →broj bridova
- **n** → broj vrhova

- prvo sortiramo bridove po težini (od najmanjeg prema najvećem) →**O(mlogm)**
- jer je m < n * (n – 1) →**O(mlogn)**
- ukupno 2m poziva Find i n – 1 poziva Union operacija →<u>ukupna složenost:</u> **O(mlogn)**

RAZMATRANJE

Kako odabrati k?

RAZMATRANJE

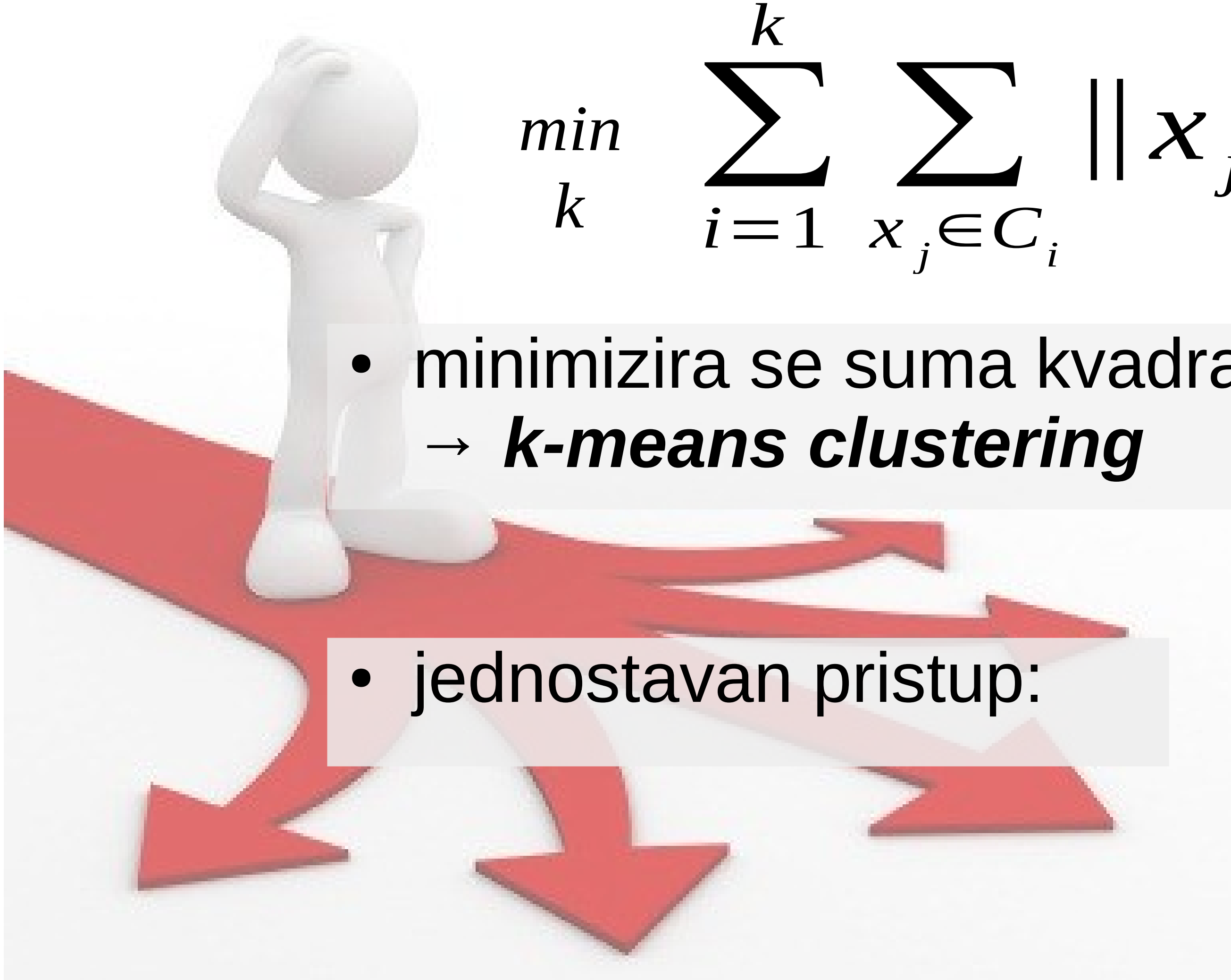
Kako odabrati k?

$$\min_{k} \sum_{i=1}^{k} \sum_{x_j \in C_i} \| x_j - \mu_j \|^2$$

- minimizira se suma kvadrata unutar klastera
  → ***k-means clustering***

RAZMATRANJE

Kako odabrati k?



$$\min_{k} \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_j||^2$$

- minimizira se suma kvadrata unutar klastera → ***k-means clustering***

- jednostavan pristup: $k \approx \sqrt{\frac{n}{2}}$

REZULTATI

REZULTATI

opisno

REZULTATI

opisno

Ulaz:

- 100 slučajno generiranih težinskih neusmjerenih grafova s n vrhova i n * (n – 1) / 2 bridova
- težine su slučajno generirane (*double* preciznosti)
- n slučajan cijeli broj iz skupa [20, 520]

REZULTATI

opisno

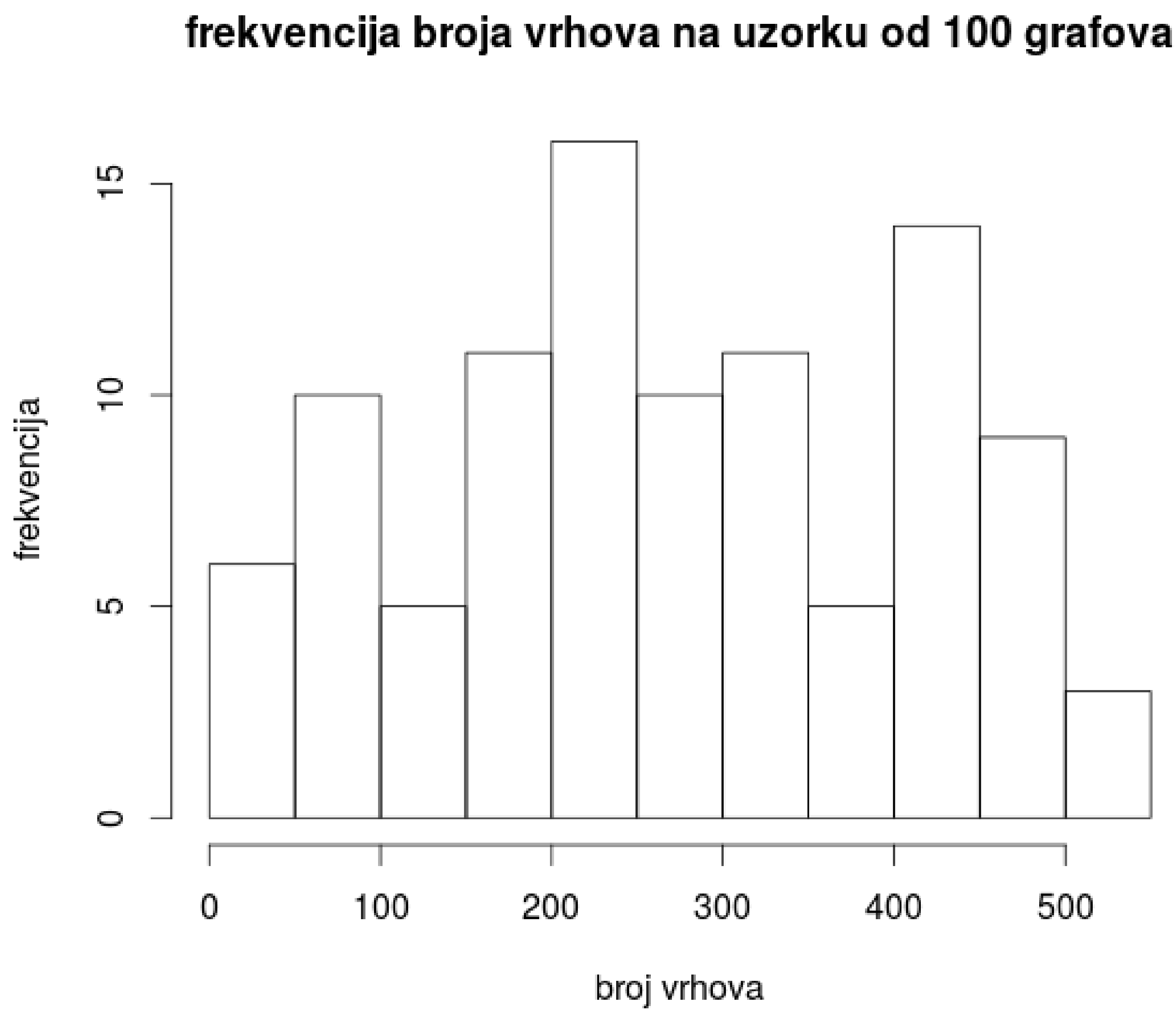
Ulaz:

- 100 slučajno generiranih težinskih neusmjerenih grafova s n vrhova i n * (n – 1) / 2 bridova
- težine su slučajno generirane (*double* preciznosti)
- n slučajan cijeli broj iz skupa [20, 520]
- za svaki graf mjeri se vrijeme izvršavanja k-klasteriranja (500 iteracija) → uzima se prosjek

$$k = \lfloor \frac{n}{2} \rfloor$$

REZULTATI

opisno



REZULTATI

opisno

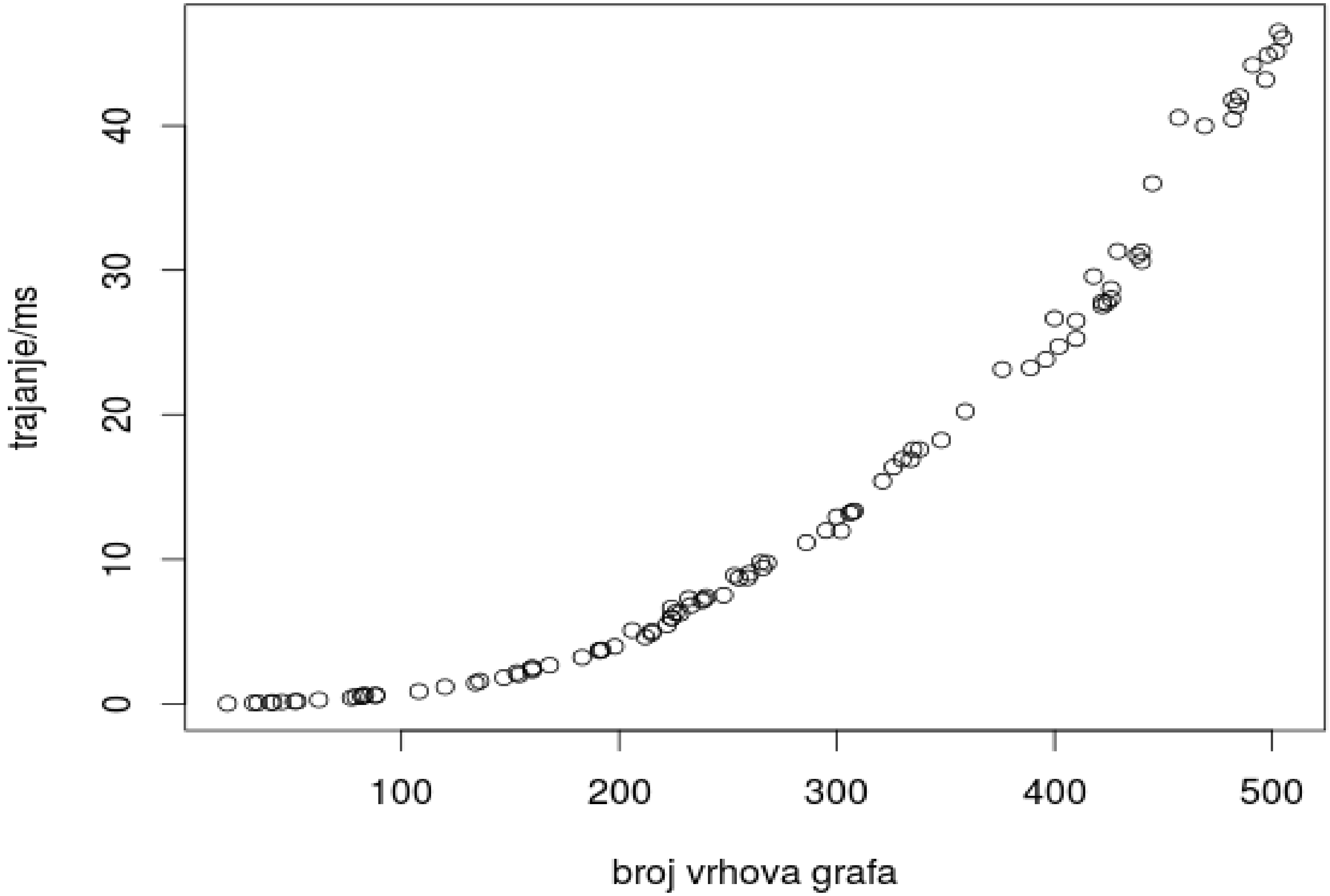
Izlaz:

- prosječno vrijeme u *ms* potrebno za izvršavanje k-klasteriranja za svaki od 100 grafova
- usporedba prosječnog vremena izvršavanja u ovisnosti o
  - broju vrhova u grafu
  - k
  - broju bridova u grafu
  - mlogn

REZULTATI

prikaz

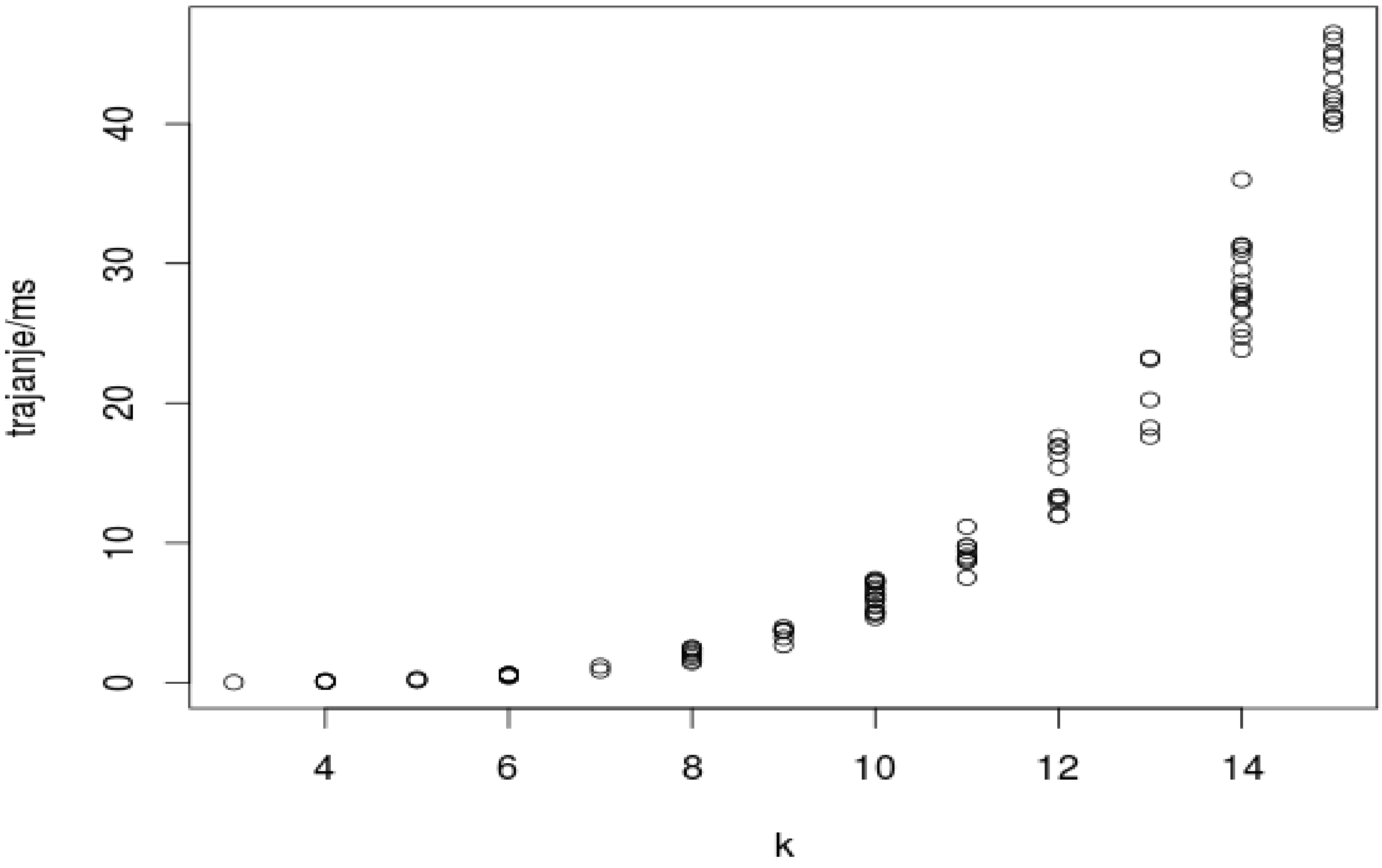
ovisnost prosječnog trajanja traženja k-klastera u ovisnosti o broju vrhova



REZULTATI

prikaz

ovisnost prosječnog trajanja traženja k-klastera u ovisnosti o k

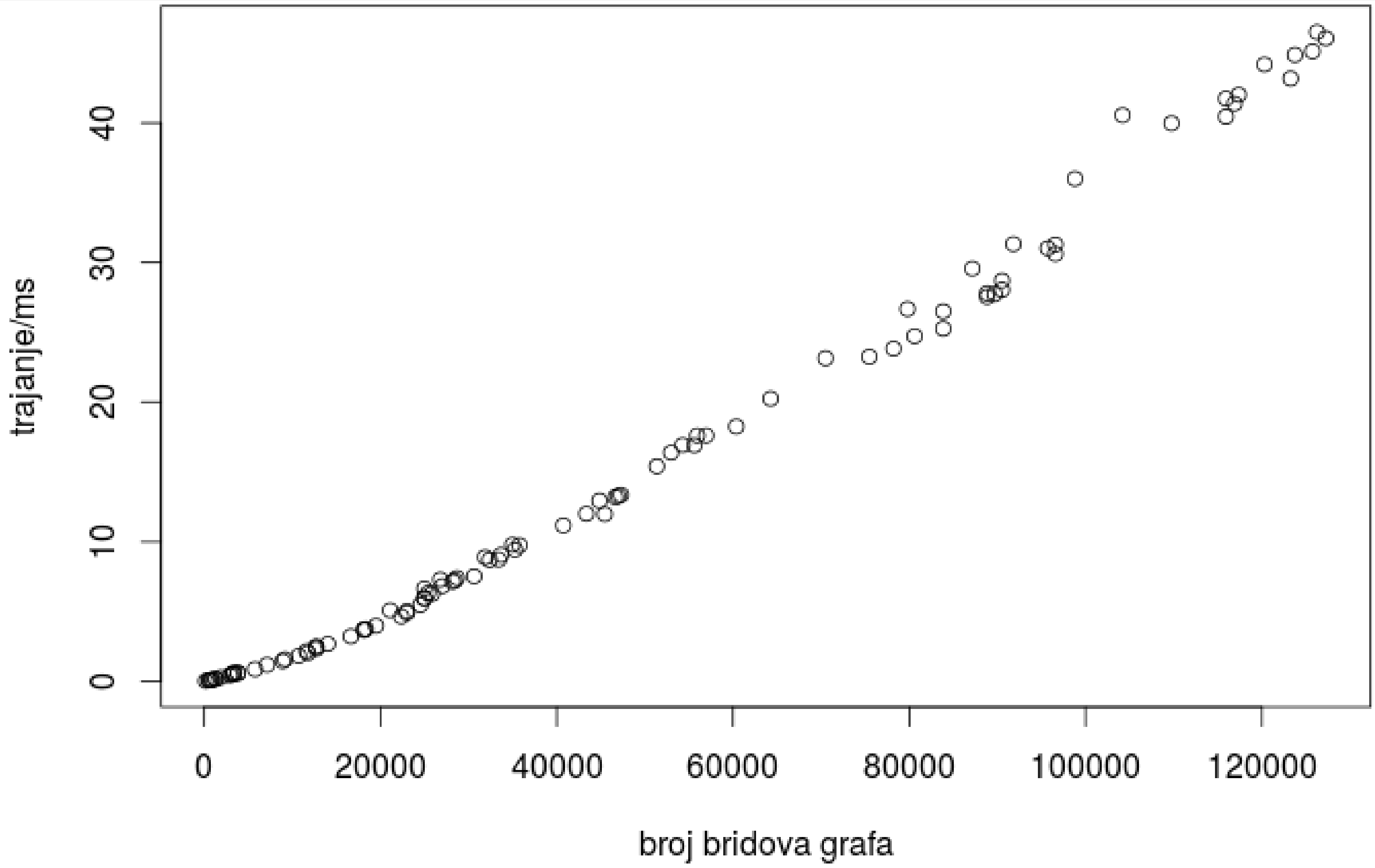


$$k = \lfloor \frac{n}{2} \rfloor$$

REZULTATI

prikaz

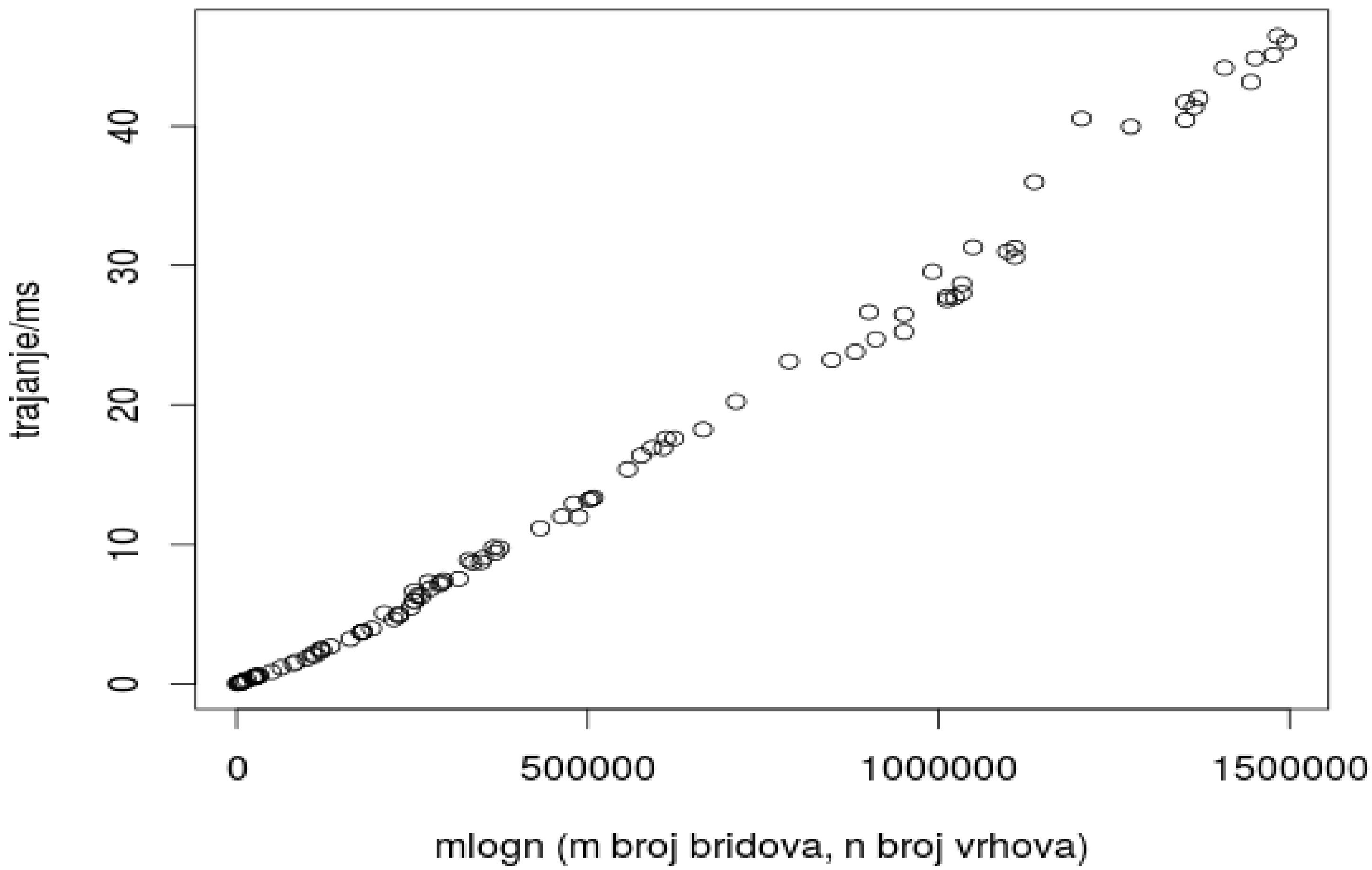
ovisnost prosječnog trajanja traženja k-klastera u ovisnosti o broju bridova



REZULTATI

prikaz

ovisnost prosječnog trajanja traženja k-klastera u ovisnosti o *mlogn* (m broj bridova, n broj vrhova)



LITERATURA

- *J.Kleinberg, E.Tardos, Algorithm Design, Pearson Education, 2006.*
- *R.Sedewick, K.Wayne, Algorithms, 4th edition,* dostupno na *http://algs4.cs.princeton.edu/home/* , (siječanj 2016.)