Graf $G = (V, E)$

↳ konečné množiny

neorientovaný

$E = \{\{u, v\} \mid u \in V \wedge v \in V\}$

orientovaný

$E = \{(u, v) \mid u \in V \wedge v \in V\}$

Reprezentace grafu (tj. hran)

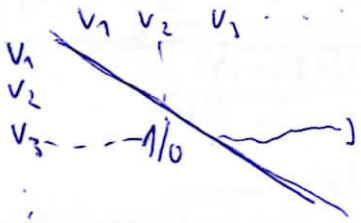
	prostor	dotaz na hranu mezi u a v	následníci u
seznam hran	$\lvert E\rvert$	$\lvert E\rvert$	$\lvert E\rvert$
matice sousednosti	$\lvert V\rvert^2$	1	$\lvert V\rvert$
seznam následníků	$\lvert E\rvert + \lvert V\rvert$	$\deg(v)$	$\deg(v)$
množina následníků	$\lvert E\rvert + \lvert V\rvert$	$\log(\deg(v))$	$\deg(v)$

seznam hran

	orient
$h_1 = \{u_1, v_1\}$	(u_1, v_1)
$h_2 = \{u_2, v_2\}$	(u_2, v_2)
⋮	
$h_n = \{u_n, v_n\}$	(u_n, v_n)

prostor: $\lvert E\rvert \times$ dotaz na hranu mezi u a v: $\lvert E\rvert \times$ následníci u: $\lvert E\rvert \times$

matice sousednosti



1 – je hrana

0 – není hrana

pro neorientovaný G je matice symetrická

prostor: $\lvert V\rvert^2 \times$ dotaz na hranu mezi u a v: 1 $\times$ následníci u: $\lvert V\rvert \times$

seznam následníků

$v_1 \rightarrow [u_1 | u_2 | u_3 | \ldots]$ ~ následníci v_1

v_2 :

v_3 :

⋮

stupeň v = počet následníků

prostor: $\lvert E\rvert + \lvert V\rvert$ ✓ dotaz na hranu mezi u a v: $\deg(v) \leq \lvert V\rvert$ ✓ následníci u: $\deg(v) \leq \lvert V\rvert$ ✓

nejpoužívanější reprezentace

množina následníků

$v_1 \rightarrow \{u_1, u_2 \ldots u_j\}$

v_2 :

⋮

prostor: $\lvert E\rvert + \lvert V\rvert$ dotaz na hranu mezi u a v: $\log(\deg(v))$ následníci u: $\deg(v)$

$\log(\deg(v))$ ↳ reprezentace množiny pomocí stromů

eventuelně konstantní přístup (hashování)

Neorientované grafy

sled (může se opakovat hrany i vrchol)

$sled(v_1, v_n) \Leftrightarrow v_1 e_1 v_2 e_2 \dots v_n \wedge \boxed{e_i = \{v_i, v_{i+1}\}}$

tah (může se opakovat jen vrchol)

$tah(v_1, v_n) \Leftrightarrow sled(v_1, v_n) \wedge \boxed{\forall 1 \le i, j \le n : i \ne j \Rightarrow e_i \ne e_j}$

cesta (nic se nemůže opakovat)

$cesta(v_1, v_n) \Leftrightarrow tah(v_1, v_n) \wedge \boxed{\forall 1 \le i, j \le n : i \ne j \Rightarrow v_i \ne v_j}$

kružnice (cyklus)

$kružnice(V_1) \Leftrightarrow$ tah $(v_1, v_n) \wedge \boxed{V_1 = V_n \wedge \forall 1 \le i, j \le n-1 : i \ne j \Rightarrow V_i \ne V_j}$

myšlenka: sudá množina vrcholů dosažitelných z S

souvislost $(G) \Leftrightarrow \forall u, v \in V$: cesta (u, v) — celkově $O(|E|+|V|)$ lépe ~ ala BFS

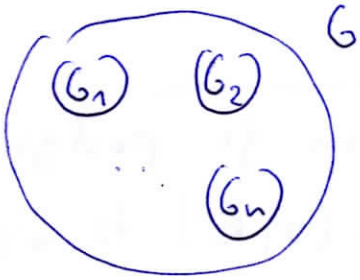
```
souvislost(G)                                   O(|V|)
S := random_node(G)
Reach := {S}
do
    new := false                                O(|E|)
    foreach {u,v} ∈ E do        seznam hran
        if |{u,v} ∩ Reach| = 1 then
            Reach := Reach ∪ {u,v}              uniká v O(1)
            new := true
while new
if |Reach| = |V| return true
else return false
```

Celkově $O(|V| \cdot |E|)$!!!

V.c – barva vrcholu:
- white – vrchol nebyl navštíven
- black – vrchol byl navštíven

```
souvislost(G)                                   O(|V|)
foreach v ∈ V v.c = white
S := random_node(G); Reach = ∅
S.c := black ~> byl navštíven
Q = ∅ ~> fronta vrcholů ke zpracování
Enqueue(Q,s) ~> přidej s
while Q ≠ ∅ do          každou hranu právě jednou!
    u := Dequeue(Q) ~> vrjmi v O(E)
    foreach v ∈ Adj(u) ~> seznam následníků
        if v.c = white ~> nebyl navštíven
            v.c := black
            Enqueue(Q,v) ~> zpracuj v
foreach v∈V do if v.c = black then
    Reach := Reach ∪ {v}
```

Rozklad na souvislé komponenty



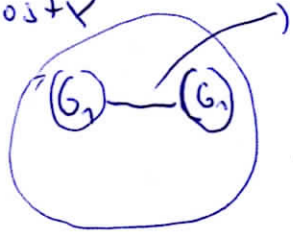
modifikujeme alg. pro souvislost tak, že vrací množinu Reach.

```
Rozklad (G)
  i := 1
  while V ≠ ∅ do
      G_i := souvislost (G)
      V := V \ V_i   ~> odstraním vrcholy z G_i
      i := i+1
  return { G_j | 1 ≤ j < i }
```

složitost: $O(|E|+|V|)$

Projdeme každou hranu a vrchol jen jednou

Mosty



most: odebráním hrany porušíme souvislost v G_1

```
Most (e, G) - je e mostem v G?
  n = |Rozklad (G)|
  E = E \ {e}
  m = |Rozklad (G)|
  return ! n = m
```

$O(|E|+|V|)$

Most (G) -> najdi všechny mosty

- opakovaně voláním Most (e, G) pro všechny hrany

složitost $O(|E| \cdot (|E|+|V|))$!!! umíme v $O(|E|+|V|)$ pomocí DFS

LES(G) $\Leftrightarrow$ $\nexists$ cyklus v G

STROM(G) $\Leftrightarrow$ LES(G) $\wedge$ souvislost(G)

---

kostra: pro souvislý $G = (V, E)$ kostra je podgraf $G' = (V, E')$ tž. G' je strom

ξ

dá se modifikovat pro nesouvislé

alg. pro hledání kostry: modifikujeme souvislost(G)

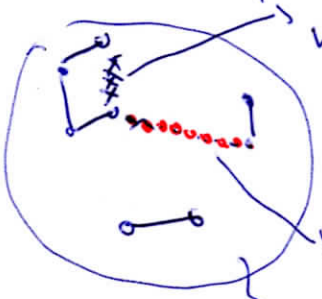
pokud najdeme hranu tž.
$v \in Adj(u) \wedge v.c = black$
(t.j. našli jsme cyklus)
tuto ji odstraníme

---

minimální kostra máme ohodnocení hran $c: E \to \mathbb{R}$

Dva základní přístupy: Kruskal vs. Prim

Kruskal
- setřídíme hrany do neklesající posloupnosti
- postupně přidáváme hrany co nevytvoří cyklus

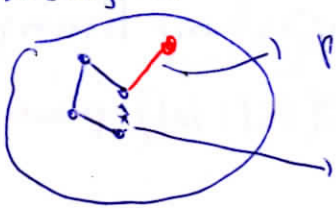


nepřidáme hranu (tvoří cyklus)

přidáme hranu (propojuje komponenty)

budujeme strom spojování komponent lesu

Prim
- budujeme strom přidáváním hran k existujícímu stromu



přidáme

nepřidáme (tvoří cyklus)

Efektivní implementace Kruskala

Foreach $v \in V$ do make_set(v) ⇝ vytvoříme les lidí, kdy každý vrchol je v vlastní komponentě

$E' := \emptyset$

$E' := sort(E')$ → $O(|E| \log(|E|))$

For each $\{u,v\} \in E$

  If find_set(u) ≠ find_set(v) then

    $E' = E' \cup \{\{u,v\}\}$

    union(u,v) ⇝ spojíme obě komponenty ⇝ hranou spojíme dvě různé komponenty → přidáme do kostry

Return E

implementace pomocí union-find struktury

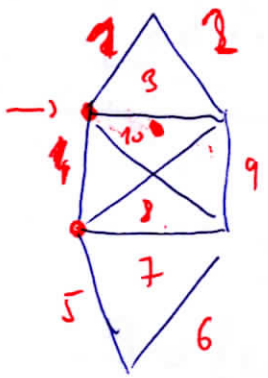
$\log(|V|)$ ⇝ dá se dále zlepšit!

celkově dominuje řazení

celkově $O(|E| \cdot \log(|V|))$

Efektivní Prim viz slajd + vizualizace

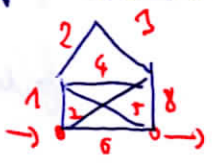
Eulerovský graf: ∃ uzavřený tah, který obsahuje každou hranu



Efektivní algoritmus: (nutná podmínka)

∀ v ∈ V: deg(v) je sudý

Poloeulerovský graf: ∃ tah, který obsahuje každou hranu

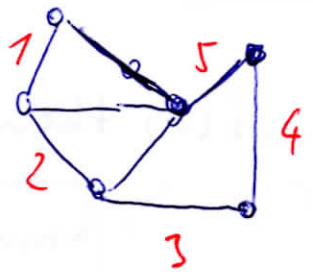


nutná podmínka: právě dva vrcholy mají lichý stupeň

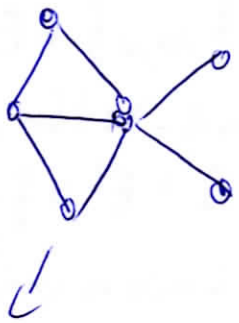
∀ v ∈ V: deg(v) je sudý nebo

Hamiltonská kružnice: prochází každým vrcholem právě jednou

Hamiltonský graf má Hamiltonskou kružnici



vs



nemá H. k.

Ne existuje efektivní algoritmus
(tj. lepší než vyzkoušet všechny kružnice)
je to **NP-complete problem**

Ale existují postačující podmínky
($\Leftarrow$) G je hamiltonský

nutná vs. postačující podmínka

Batvení:
B je množina barev
funkce $f: V \to B$ je obarvení $\Leftrightarrow \forall \{u,v\} \in E: f(u) \neq f(v)$

Typické úlohy

- je graf k-obarvitelný ⇒ NP-complete pro $k \geq 3$

  ⇕

  ∃ obarvení k barvami

- najdi nejmenší k t.ž G je k-obarvitelný

  ⇕

  též NP-complete

Existuje několik heuristik na hledání min. obarvení např. barvení na základě stupně vrcholu

- seřadím vrcholy sestupně podle jejich stupně
- postupně jim přiřazuju vždy nejnižší možnou barvu (t.j. takovou, která neporuší podmínku barvení)

- vlastně viz slajdy

.

souvislost idea

(2.5)

už ve frontě byl, takže ho tam nedáváme / případně nevoláme

new := true

dáváme do fronty

⇝ budujeme množinu Reach

O-notace

(1.5)

$2 \cdot |E| \in O(|E|)$ ale $|E|^2 \notin O(|E|)$

více na STI v úterý

```
1  PRIM((V, H), w, r)
2  Q ← V
3  foreach v ∈ Q do key[v] ← ∞ od
4  key[r] ← 0
5  π[r] ← NIL
6  while Q ≠ ∅ do
7      u ← EXTRACT-MIN(Q)
8      foreach v takový, že (u, v) ∈ H do
9          if v ∈ Q ∧ w(u, v) < key[v]
10             then π[v] ← u
11                 DECREASE-KEY(Q, v, w(u, v)) fi od od
12 return {(π[v], v) | v ∈ V, v ≠ r}
```

celkově $O(|V| \cdot \log(|V|) + |E|)$

$|V|$-krát

$\log(|V|)$

$O(1)$

$|E|$-krát

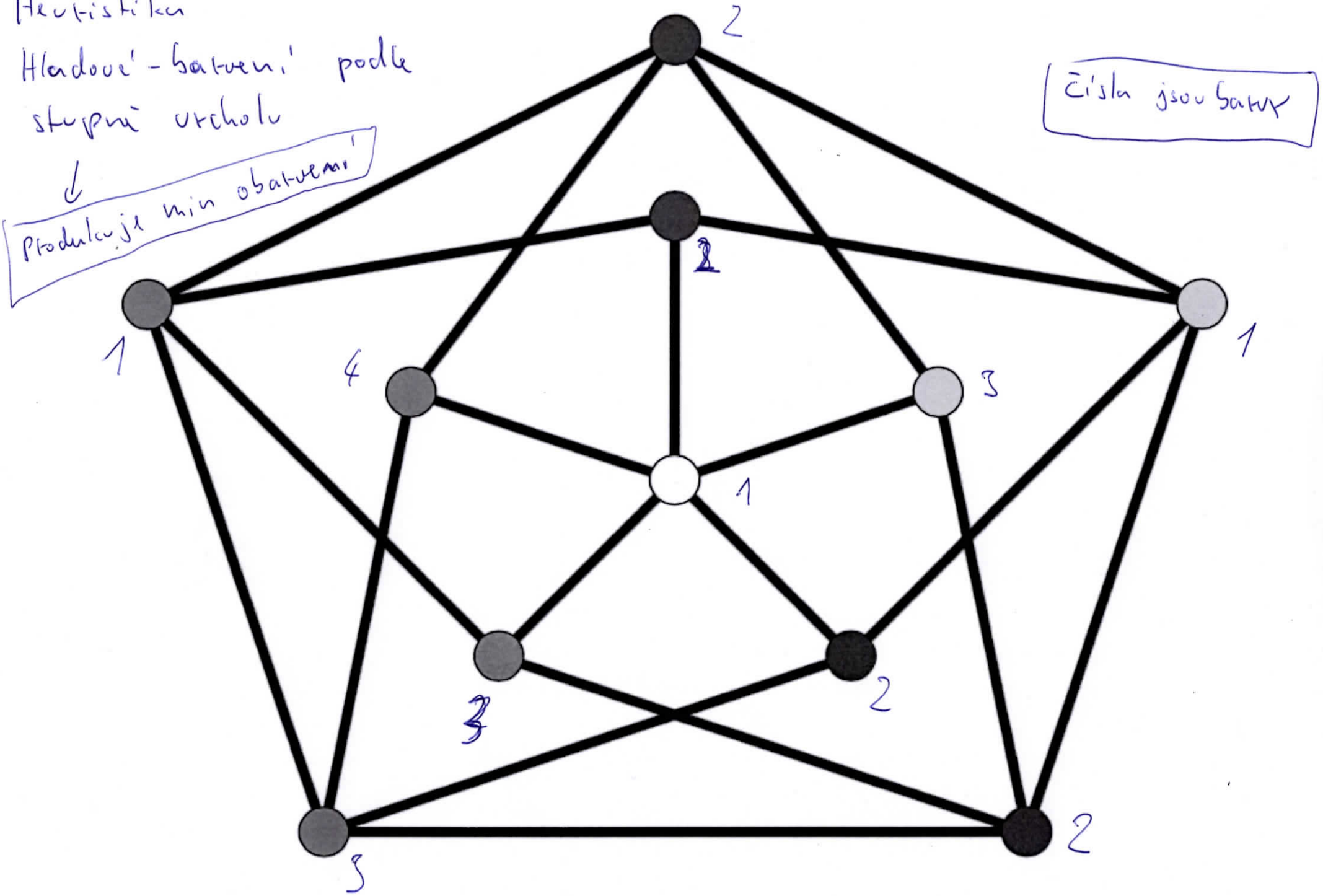
Q je implementována jako Fibonacciho halda ⇝ požadovaná složitost

Heuristika

Hladové obarvení podle stupně vrcholu

↓

Prodlužuje min obarvení

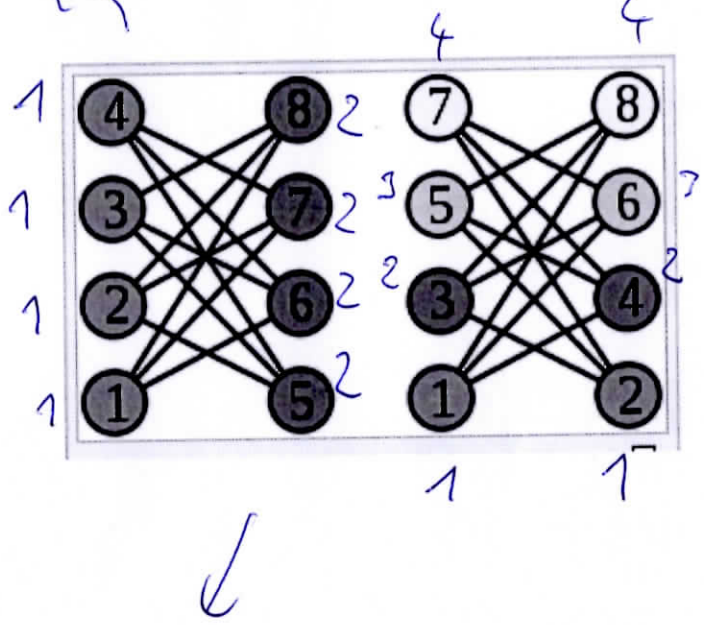


Čísla jsou barvy

Ukázka kde nefungují hladové barvení podle stupně

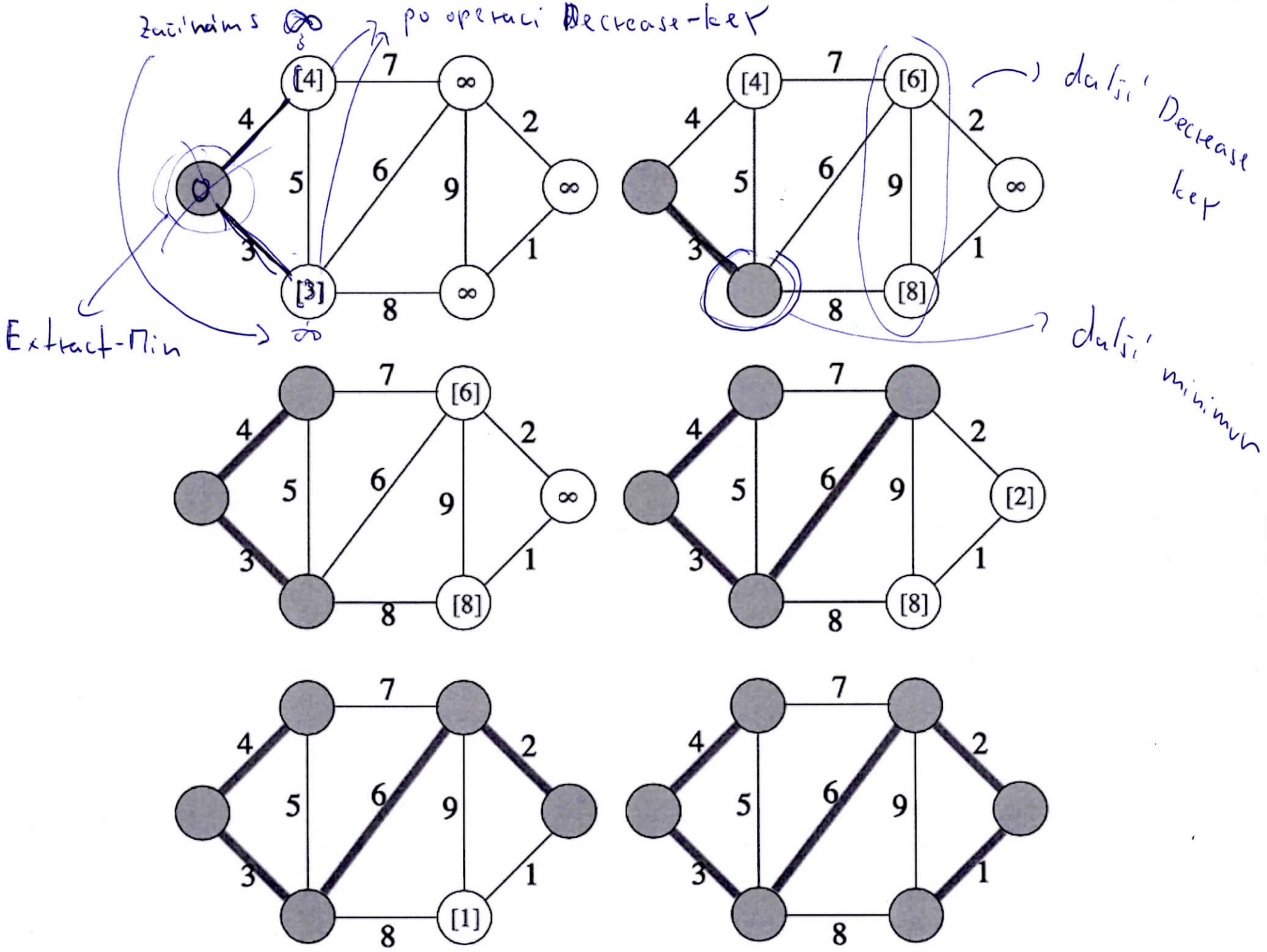
"moje" čísla jsou barvy

minimální obarvení



špatné obarvení

všechny vrcholy mají stejný stupeň!


Začínáme s
po operaci Decrease-key
Extract-Min
další Decrease key
další minimum
[4]
[3]
[6]
[8]
[2]
[1]
∞
0
7
4
5
6
9
2
3
1
8