

# NTIN060 | Homework 06/6

Milan Wikarski (milan@wikarski.sk)

## Zadanie

**6. Revoluce u Bobří řeky.** Chcete zavést KaraNET i ve své domovské vesnici Kravařích. Sít' možných spojení je bohužel trochu řidší a spojení odpovídá rovinnému grafu. Jak dlouho bude trvat nalézt páteřní síť v tomto rozložení?

## Riešenie a dôkaz správnosti

Majme súvislý rovinný graf  $G = (V, E)$ , zobrazenie  $E \rightarrow w$ , ktoré každej hrane priradí nejakú váhu. Hľadáme minimálnu kostru grafu  $G$ .

Pre nájdenie minimálnej kostry grafu  $G$  použijeme Jarníkuv algoritmus. Vyberieme si nejaký vrchol  $v_0$  (náhodne) a začneme so stromom  $T = (\{v_0\}, \{\})$ , ktorý obsahuje iba vrchol  $v_0$  a žiadne hrany.

### Stav vrcholov

Každý vrchol  $v \in V$  sa bude nachádzať v jednom z troch stavov:

1. **IN** - ak  $v$  patrí do  $T$ .
2. **NEIGHBOR** - ak  $v$  nepatrí do  $T$  a zároveň existuje hrana  $e = \{u, v\} \in G$  medzi  $T$  a zvyškom grafu  $G$  (tzn.  $u \in V(T)$ ,  $v \in V(G) \setminus V(T)$ ).
3. **OUT** - v ostatných prípadoch.

Tieto stavy budú uložené v poli `state`.

### Ohodnotenie vrcholov

Každý vrchol  $v \in V$  budú mať ohodnotenie `eval[v]` - to odpovedá:

- $+\infty$ , ak `state[v] = OUT`.
- najnižšej váhy z hrán, ktoré vedú z vrchola  $v$  do nejakého vrchola  $u \in V(T)$ , ak `state[v] = NEIGHBOR`.
- ak `state[v] = IN`, hodnota `eval[v]` nás nezáujíma a nebude uložená

Ohodnotenie hrán si uložíme do minimálnej haldy, ktorá bude mať operácie ExtractMin a Decrease.

### Predchodcovia

Ďalej si budeme držať pole predchodcov `pred[v]`. Bude platiť, že ak `state[v] = NEIGHBOR`, potom hrana  $e = \{v, \text{pred}[v]\}$  je hrana s najnižším ohodnotením, ktorá vedie z vrchola  $v$  do nejakého vrchola  $u \in V(T)$ .

### Beh algoritmu

V každom kroku algoritmu si vyberieme vrchol  $v \in V$  s najnižším ohodnotením `eval[v]` z haldy (pomocou operácie ExtractMin), zmeníme stav vrchola  $v$  na **IN** a pripojíme ho ku grafu  $T$  hranou  $e = \{v, \text{pred}[v]\}$ . Ďalej prepočítame hodnotu `eval[u]` pre všetkých susedov  $u$  vrchola  $v$ , ktorí nepatria do  $T$  (tzn. platí `state[u] != IN`); konkrétne ju znížime, ak hrana medzi vrcholmi  $u$  a  $v$  má nižšiu váhu, ako je hodnota `eval[u]`.

### Správnosť

Tu je dôležité, že ani hrana  $e$ , ani vrchol  $v$  nepatria do grafu  $T$ , a teda ku  $T$  pripájame listy. Z toho dostávame, že graf  $T$  bude počas celého behu algoritmu strom.

Algoritmus sa po najviac  $n$  krokoch zastaví, keďže v každom kroku sme ku stromu  $T$  pridali jeden vrchol.

V každom kroku vyberáme najľahšiu hranu elementárneho rezu medzi stromom  $T$  a zvyškom grafu. Všetky vybrané hrany teda ležia v minimálnej kostre, a teda strom  $T$  je minimálnou kostrou grafu  $G$ .

## Algoritmus

ALGORITMUS: Jarníkův

INPUT: graf  $G = (V, E)$ , zobrazenie  $E \rightarrow w$

OUTPUT: Strom  $T$ , ktorý je minimálnou kostrou grafu  $G$

```
1. for ( $\forall v \in V$ ):
2.   eval[v] <-  $+\infty$ 
3.   pred[v] <- null
4.   state[v] <- OUT
5.  $v_0 \leftarrow$  random vertex from  $V$ 
6. eval[v0] <- 0
7. state[v0] <- NEIGHBOR
8. while (exists  $v \in V$  such that state[v] = NEIGHBOR):
9.    $v \leftarrow$  ExtractMin(eval)            $\triangleleft$  Vyberieme vrchol s najnižšou hodnotou eval z haldy
10.  state[v] <- IN
11.  if (pred[v] is not null):              $\triangleleft$  Nutná kontrola, pretože pred[v0] is null
12.     $V(T) \leftarrow V(T) \cup \{v\}$        $\triangleleft$  Pridáme vrchol v
13.     $E(T) \leftarrow E(T) \cup \{v, \text{pred}[v]\}$   $\triangleleft$  Pridáme hranu {v, pred[v]}
14.  for ( $\forall u$ ;  $u$  is neighbour of  $v$ ):     $\triangleleft$  Každý vrchol u spojený hranou s vrcholom v
15.    if (state[u] != IN and eval[u] > w({u, v})):
16.      state[u] <- NEIGHBOR
17.      eval[u] <- w({u, v})
18.      pred[u] <- v
19. return T
```

## Časová a priestorová zložitosť

Označme si  $n = |V|$  počet vrcholov a  $m = |E|$  počet hrán.

### Časová zložitosť

Jarníkův algoritmus nájde minimálnu kosť pri použití halde v čase  $O(m \cdot \log(n))$ . Pre rovinné grafy platí  $m \leq 3n - 6$ , z čoho dostávame  $O(m \cdot \log(n)) = O((3n - 6) \cdot \log(n)) = O(n \cdot \log(n))$ .

### Priestorová zložitosť

Algoritmus pracuje s pomocnými poľami `pred` a `state` a haldou `eval` ktoré majú dĺžku  $n$ . Ďalej potrebujeme pamäť na ukladanie kostry  $T$ , čo je najviac  $O(n + m)$ . Z hore spomenutého odhadu ale máme  $O(n + m) = O(n + 3n - 6)$ . Dokopy to všetko je  $O(3n + n + 3n - 6) = O(n)$ .