Najlacnejšia kostra

Kostra grafu: Podmnožina hrán T grafu $G = (V, E)$ taká, že:

- graf (V, T) je súvislý
- neobsahuje žiadne cykly

Kostra má práve $n - 1$ hrán

Úloha: Daný súvislý neorientovaný ohodnotený graf $G = (V, E)$
Nájdite kostru grafu s minimálnou váhou $w(T) = \sum_{e \in T} w(e)$

Kruskalov algoritmus (minimálna kostra)

```
Sort edges in order of increasing weight
so that w[f[1]] <= w[f[2]] <= ... <=w[f[m]]

T=empty set
for i:=1 to m do
  let u,v be the endpoints of edge f[i]
  if there is no path between u and v in T then (**)
    add f[i] to T
return T
```

Implementácia (**) pomocou UNION/FIND-SET:
$O(\log n)$ na jedno volanie (**) $\Rightarrow O(m \log n)$

Dôkaz správnosti

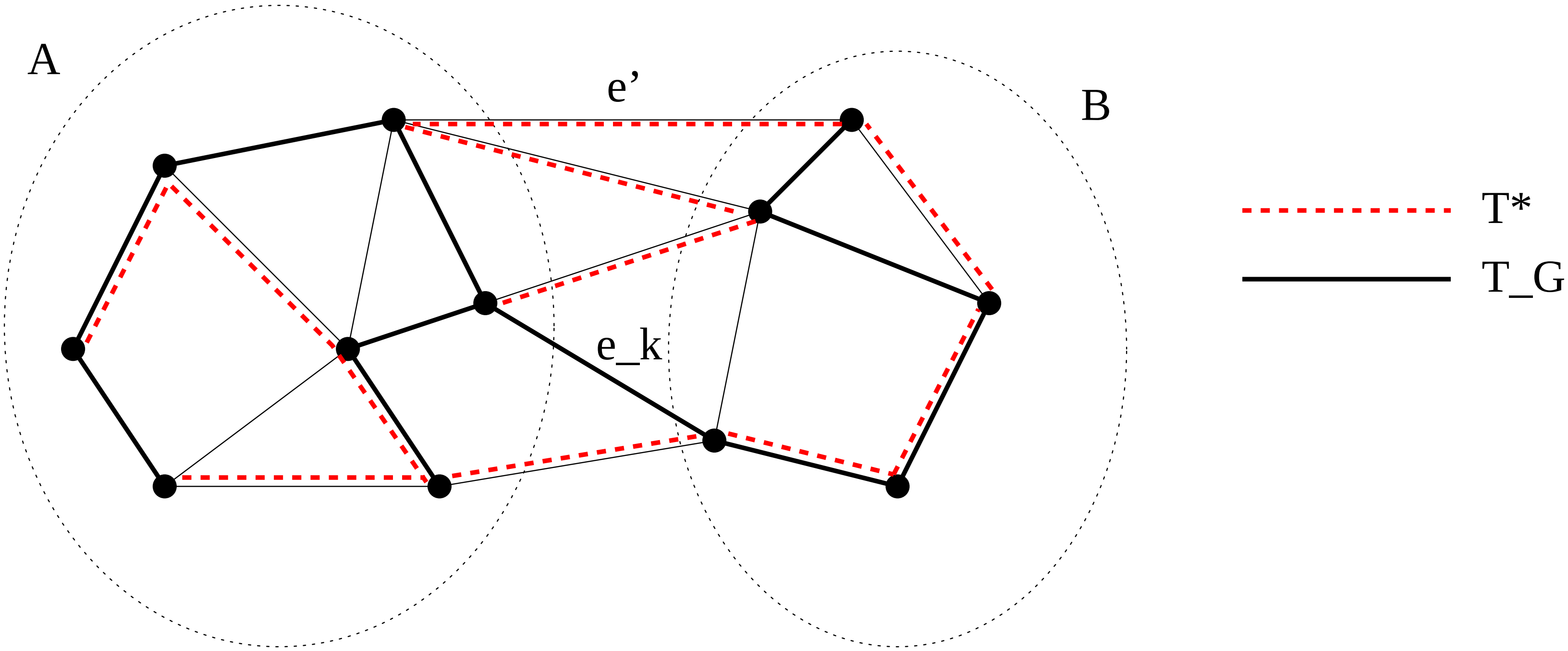
Tvrdenie: Nech riešenie T_G vypočítané pomocou Kruskalovho algoritmu obsahuje hrany $e_1, \ldots, e_{n-1}$ (v poradí pridávania) Pre každé $0 \leq k \leq n-1$ **existuje minimálna kostra** obsahujúca hrany $e_1, \ldots, e_k$

Komplikácia: hrany s rovnakou váhou

Dôkaz indukciou: pre $k = 0$ platí triviálne

Platí pre $k-1 \Rightarrow$ existuje min. kostra T^* obsahujúca $e_1, \ldots, e_{k-1}$
Dokážeme: existuje aj min. kostra T' obsahujúca $e_1, \ldots, e_k$

Platí pre $k-1 \Rightarrow$ existuje min. kostra T^* obsahujúca $e_1, \ldots, e_{k-1}$
Dokážeme: existuje aj min. kostra T' obsahujúca $e_1, \ldots, e_k$



$$w(T') = w(T^*) + \underbrace{w(e_k) - w(e')}_{\leq 0}$$

$$w(T') \leq w(T^*)$$

Dokázali sme:

Tvrdenie: Nech riešenie T_G vypočítané pomocou Kruskalovho algoritmu obsahuje hrany $e_1, \ldots, e_{n-1}$ (v poradí pridávania) Pre každé $0 \leq k \leq n-1$ **existuje minimálna kostra** obsahujúca hrany $e_1, \ldots, e_k$

Postupne "pretvárame" (akúkoľvek) minimálnu kostru na T_G, pričom v žiadnom ktorku **nezmeníme váhu kostry**.

$\boxed{\text{Pre } k = n-1\text{: } T_G \text{ je minimálna kostra}}$

Na cvičeniach: Primov algoritmus

Hľadanie artikulácií v grafoch

Artikulácia: Vrchol, ktorý keď odoberieme z grafu, tak sa niektorý komponent súvislosti rozpadne.

Úloha: Daný je neorientovaný graf $G = (V, E)$
Nájdite v ňom všetky artikulácie

Triviálny prístup: odober vrchol, spočítaj komponenty
($n\times$ spusti prehľadávanie do hĺbky)
Časová zložitosť: $O(n(m + n)) = O(mn)$

Ide to aj lepšie?

Prehľadávanie do hĺbky—rekurzívna funkcia

```
function dfs-visit(v,cnum)
  // pre-condition: v is WHITE vertex
  // find all vertices that are reachable from v
  // by path going through white vertices only
  status[v]:=gray;
  num[v]:=cnum;
  for each w in out(v)
    if status[w]=white
       dfs-visit(w,cnum)
  status[v]:=black;
```

(biely=nepreskúmaný; čierny=ukončený; sivý=v procese)

Prehľadávanie do hĺbky—hlavný program

```
// --- main program ---
status of all vertices is white
cnum=0; // component number
for all vertices v in V
  if status[v]=white
    // all vertices in v's component are white;
    // explore v's component
    dfs-visit(v,cnum);
    cnum:=cnum+1;
```

Ako sa artikulácie správajú pri prehľadávaní do hĺbky?

Pozorovanie: Ak je vrchol artikulácia, tak z jeho podstromu nevyjdeme, kým ho celý neprehľadáme.

Prehľadávanie do hĺbky—pamätáme si viac

```
function dfs-visit(v,cnum)
  status[v]:=gray;
* time:=time+1; d[u]:=time;
  num[v]:=cnum;
  for each w in out(v)
    if status[w]=white
*      edge (v,w) is a tree edge;
       dfs-visit(w,cnum)
  status[v]:=black;
* time:=time+1; f[u]:=time;
```

stromová hrana: hrana, ktorou sme objavili nový vrchol

⇒ **strom prehľadávania**

spätná hrana: všetky ostatné hrany

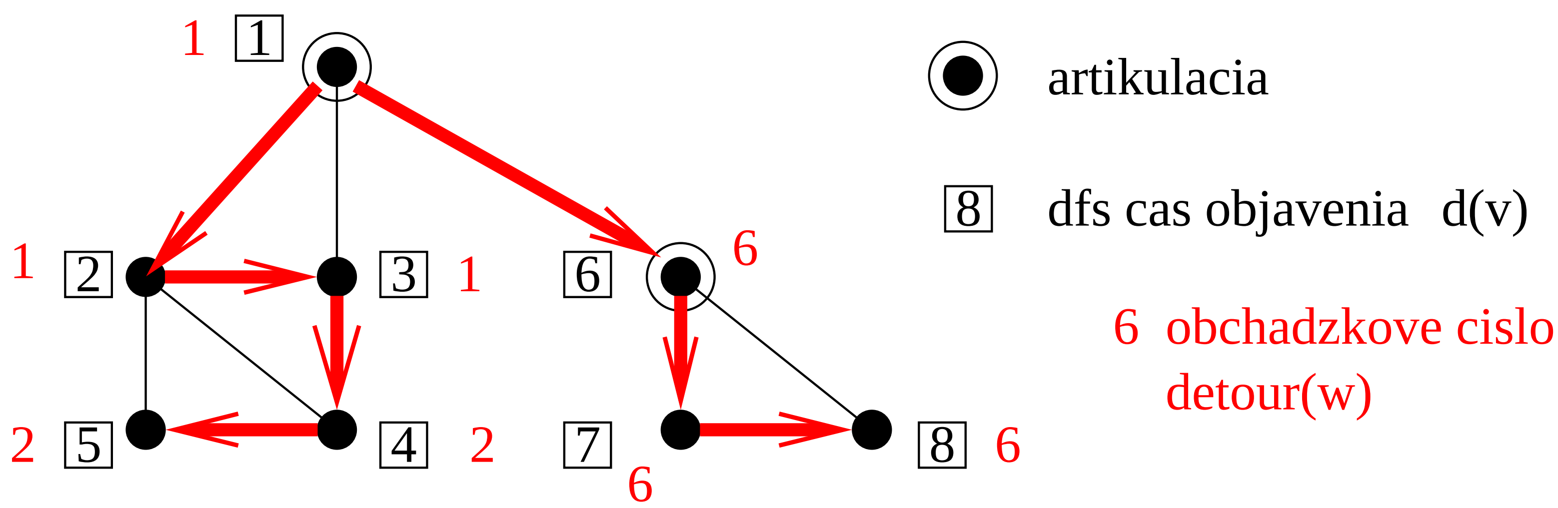
Základná vlastnosť prehľadávania do hĺbky

Lema: Nech $e = (u, v)$ je **spätná hrana**, buf $d(u) < d(v)$
Potom u je predok v v strome prehľadávania T

Dôkaz:

$d(u)$: vrchol u sa zmení z bieleho na sivý
$f(u)$: vrchol u sa zmení zo sivého na čierny
Môže byť v čase $f(u)$ vrchol v biely? NIE!
preto $d(u) < d(v) \leq f(u)$
VŠETKY vrcholy objavené medzi $d(u)$ a $f(u)$ sú potomkami u v strome prehľadávania

Dôsledok: Ak u a v nie sú vo vzťahu predok–potomok
potom **medzi potomkami u a potomkami v nevedú žiadne hrany**



Def: Obchádzkové číslo vrchola v je najmenšie DFS číslo $d(w)$ vrchola w, ktorý môžeme dosiahnuť z vrchola v tak, že najskôr prejdeme niekoľko stromových hrán a potom jednu spätnú hranu.

Lema: Vrchol v, ktorý nie je koreňom DFS stromu, NIE JE artikulácia práve vtedy, keď pre všetky jeho deti w v DFS strome platí

$$\mathrm{detour(w)} < d(v).$$

Lema: Koreň DFS stromu je artikulácia práve vtedy, keď má v DFS strome viac ako jedno dieťa.

Lema: Vrchol v, ktorý nie je koreňom DFS stromu, NIE JE artikulácia práve vtedy, keď pre všetky jeho deti w v DFS strome platí

$$\mathrm{detour(w)} < d(v).$$

$(\Rightarrow)$

Lema: Vrchol v, ktorý nie je koreňom DFS stromu, NIE JE artikulácia práve vtedy, keď pre všetky jeho deti w v DFS strome platí

$$\mathrm{detour(w)} < d(v).$$

$(\Leftarrow)$

Artikulácie: modifikácia prehľadávanie do hĺbky (nekoreň)

```
function dfs-visit(v,cnum)
  status[v]:=gray;
  time:=time+1; d[v]:=time;
* detour[v]:=v;
  for each w in out(v)
    if status[w]=white
       //--- (v,w) is a TREE edge
       dfs-visit(w,cnum); // detour[w] is now computed!
*      if detour[w]>=d[v] then vertex v is an articulation!
*      if detour[w]<detour[v] then detour[v]:=detour[w];
*   else
*      //--- (v,w) is a BACK edge
*      if d[w]<detour[v] then detour[v]:=d[w];
  status[v]:=black;
```

Časová zložitosť: $O(n + m)$

Zhrnutie

- Problém najlacnejšej kostry možno riešiť v čase $O(m \log n)$ pomocou jednoduchého greedy algoritmu (Kruskalov algoritmus)
- Precvičili sme si techniku dokazovania greedy algoritmov
- Ďalší greedy algoritmus (Primov algoritmus) na cvičeniach
- Rozšírený pohľad na prehľadávanie do hĺbky:
  - strom prehľadávania
  - čas objavenia vrcholu
  - vlastnosti stromu prehľadávania
- Modifikáciou prehľadávanie do hĺbky sme efektívne vyriešili problém hľadania artikulácií