

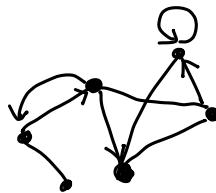
3. Razstavljanje grafov

3.1. Zakaj so grafi koristni

3.1.1. Predstavitev grafov

$$G = (V, E)$$

$$E \subseteq V \times V$$



- slovar seznamov sosedov (adjacency list)

$$\{A: [B, C, D], B: [A], C: [A, D], D: [A, C], E: []\}$$

$$D - C$$

$$\begin{matrix} | \\ A - B \end{matrix}$$

E

- matrika sosednosti

$$\begin{array}{c} A \quad B \quad C \quad D \quad E \\ \begin{pmatrix} & 1 & 1 & 1 & \\ 1 & & & & \\ 1 & & & & \\ 1 & & & & \\ 1 & & & & \end{pmatrix} \end{array}$$

- slovar (hash-tables PSAZ) mnogic sosedov

$$\{A: \{B, C, D\}, B: \{A\}, C: \{A, D\}, D: \{A, C\}, E: \emptyset\}$$

- po definiciji

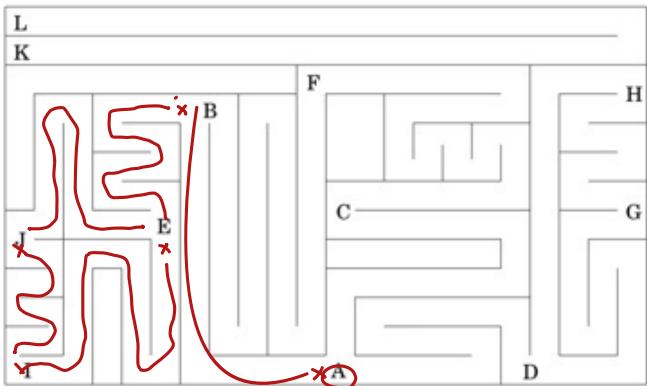
$$\{A, B, C, D, E\}, \{(A, B), (A, C), \dots, (D, C)\}$$

	slovar sez.,	matr. sos.	slovar mnog.	po definiciji
seznam vozilic	$O(V)$	$O(V)$	$O(V)$	$O(V)$
seznam povezav	$O(V + E)$	$O(V ^2)$	$O(V + E)$	$O(E)$
seznam sosedov	$O(1)$	$O(V)$	$O(\deg_G(u))$	$O(E)$
ali sta vozilci M, N sosednji	$O(\deg_G(u))$	$O(1)$	$O(1)$	$O(1)$
prostorska zahiterost	$O(V + E)$	$O(V ^2)$	$O(V + E)$	$O(V + E)$

3.2. Iskanje v globino

depth-first-search DFS

3.2.1. Raziskovanje grafov



Trémaux

*1859 Toulon,
Francija

+1882 Tipasa,
Alžirija

3.2.2. DFS

vhod graf G , vozlišče $v \in V(G)$

izhod označene vozlišča, dosegljiva iz v + ...
natanko

def razisci (G, v):

označi (v)

pripravi (v)

za vsakega sosedja $(v, w) \in E(G)$:

če w ni označen:

razisci (w)

pospravi (w)

① v dosegljiv iz $v \Leftrightarrow v$ označen

(\Rightarrow) Recimo, da je v dosegljiv iz v in v označen

$v - \dots - w - w' = M$

Naj bo w zadnje označeno vozlišče na poti.

Tedaj smo paklicali razisci (G, w). Tedaj smo paklicali tudi razisci (G, w') in s tem označili w' .

(\Leftarrow) označujemo le vozlišča, ki jih raziskujemo, raziskujemo pa le sosedce.
+ ...

② & ③ Odvisni od velikosti komponente, zato se bomo k vprašanjim vrnila kasneje.

vход graf G

изход označene vse vozlišča + ...

def dfs(G):

za vse $v \in V(G)$:

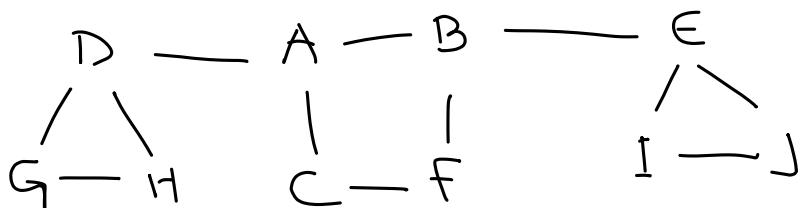
če v ni označen:

razišci(G, v)

① ✓

② $O(|V| + |E|)$

③ Hitreje se ne da, saj rabimo toliko časa, da preberemo graf.

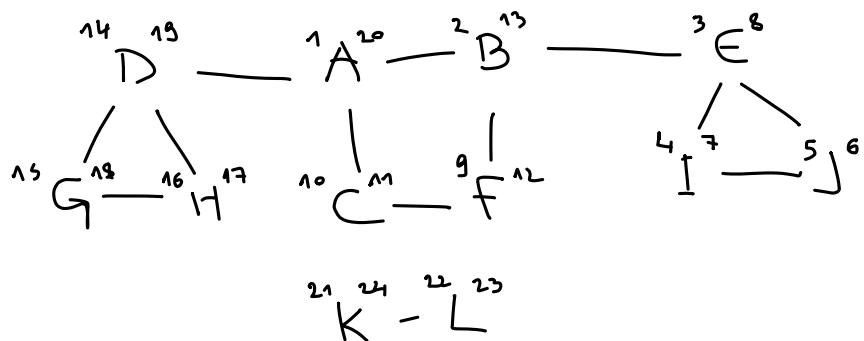


K - L

3.2.3 Primer uporabe

Računanje komponent za povezanost.
Vsakič, ko v dfs(...) zbiščemo novo vozlišče, povečamo stevec komponent.
Vsakič, ko raziščemo vozlišče, ga pridobimo trenutni komponenti.

3.2.4. Predogledna in poogledna urejenost



def pripravi(v):

korak += 1

predoznaka[v] = korak

def pospravi(n):

korak += 1

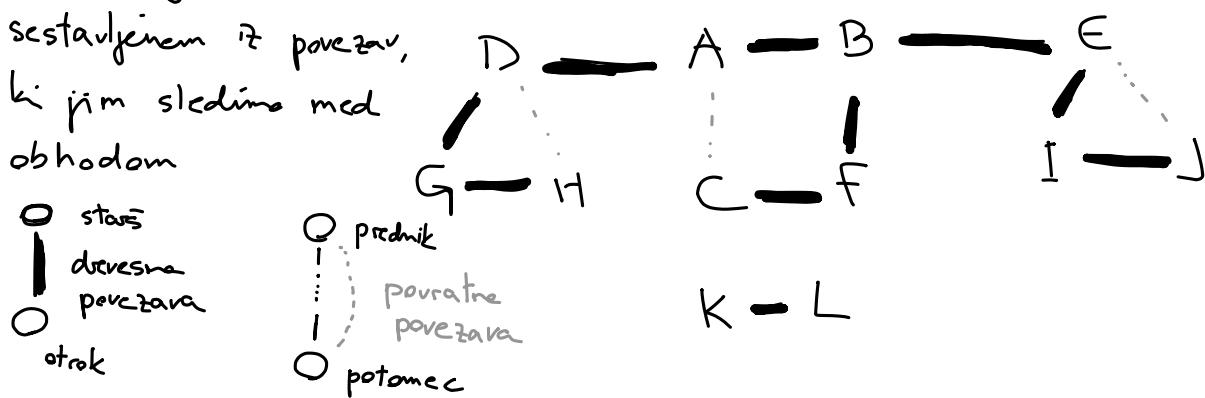
poznake[n] = korak

za poljubni vozliscí m, n vrijedi

- $[pred[u], po[u]] \supseteq [pred[v], po[v]]$
če smo do v prišli med raziskovanjem u
 - $[\dots] \subseteq [\dots]$
 $\dots \dots m \dots \dots \dots \sim$
 - $[\dots] \cap [\dots] = \emptyset$

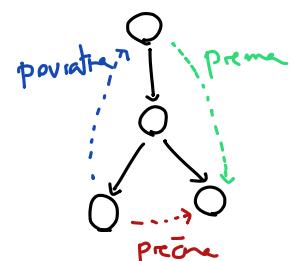
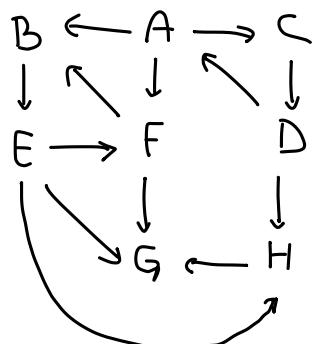
Odnos med vezlisci si lahko predstavimo na DFS gozdu,

sestaļjēnum ir povezā
ki jām sledīns med
obhodam



3.3. DFS v usmerjenih grafih

Po stopce je identičen.



3.3.1. Vrste povezav

Povezave lahko klasificiramo med obhodom.

Te inamo $m \rightarrow n$ in valje:

- $n \leq m$ obiskali - drevesna \leftarrow $po[n] > po[m]$
 - $pred[n] < pred[m]$:
 - $po[n] < pred[m]$ - prečna \leftarrow
 - $po[n]$ ni nastavljena - povratna \times
 - $po[n] > pred[m]$ \rightarrow \leftarrow \times
 - $pred[n] < pred[m]$ - prema \leftarrow

u z 2 8

3.3.2 Usmerjeni grafi brez ciklov (DAG)

zelo uporabni ✓

Trditve Usmerjen graf nima ciklov \Leftrightarrow DFS ne odkrije paratne povezave.

Dokaz

(\Leftarrow) Umejimo cikel $n_1 \xrightarrow{n_2} \dots \xrightarrow{n_n} n_1$.

Naj bo n_1 prvo vozlišče, ki ga obišče DFS.
Naj bo n_1 prvo vozlišče, ki ga obišče DFS.

$\text{pred}[n_1] < \dots < \text{pred}[n_n]$

Ko raziskujemo n_n najdemo povezavo $n_n \rightarrow n_1$.

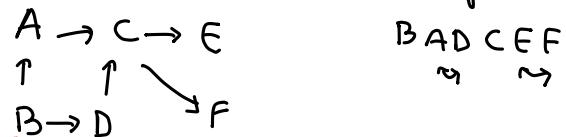
Ker n_1 še ni zaključen, imamo povratno povezavo.

(\Rightarrow) Recimo, da imamo povratno povezavo $m \rightarrow n$.

Ker je povezava povratna, je n potomec m .

Torej imamo pot $n \rightarrow \dots \rightarrow m$, zato imamo cikel $n \rightarrow \dots \rightarrow m \rightarrow n$

Def Ureditri vozlišč n_1, \dots, n_n pravimo topološke, če iz $n_i \rightarrow n_j$ sledi $i < j$.



Ce V DAGu G velja $u \rightarrow v \in E$ ~~natanko tisto, ker je $po[u] > po[v]$~~

Dokaz Sistematično pogledamo vse možnosti, pri čemer upoštevamo, da ni paratnih povezav.

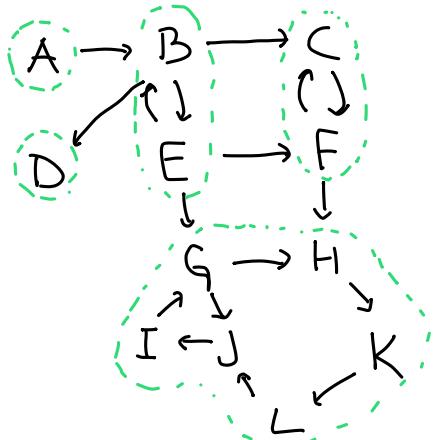
Topološko ureditve dobimo tako, da ureditimo padajoče po po-oznakah.
To lahko naredimo tako, da v DFS vozlišč ob zaključku damo na sklad.



Tarjan
*1948
Pomona,
CA, ZDA

3.4. Krepko povezane komponente

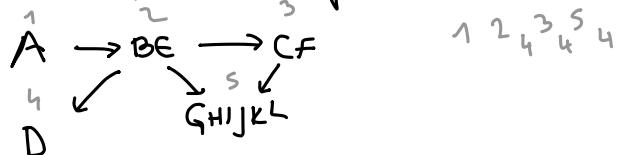
3.4.1. Povezanost v usmerjenih grafih



M in N sta povezani,
če obstaja pot $M \rightarrow \dots \rightarrow N$
in pot $N \rightarrow \dots \rightarrow M$.

To je ekvivalentna relacija
in njenim razredom pravimo
krepko povezane komponente

Iz vsakega usmerjenega grafa lahko tvorimo kvocientni graf,
ki ima za vozlišča KPK C_1, \dots, C_k in povezave $C_i \rightarrow C_j$.
natančno takot, ko obstajata $m \in C_i$ in $n \in C_j$, da velja $m \rightarrow n$.



Trditve Kvocientni graf je DAG.

Dokaz Če bi imel cikel $C_1 \rightarrow \dots \rightarrow C_l \rightarrow C_1$, bi bila $C_1 \cup C_2 \cup \dots \cup C_l$ KPK.

3.4.2. Določanje KPK

Ideja Začnemo v enem od pohrov kvocientnega grafa in na njem izvedemo raziskovanje. Tako dobimo eno komponento, ki jo odstranimo iz nadaljnjega raziskovanja.

Trditve Če v kvoc. grafu velja $C \rightarrow C'$, potem je $\max_{u \in C} po[u] > \max_{u \in C'} po[u]$

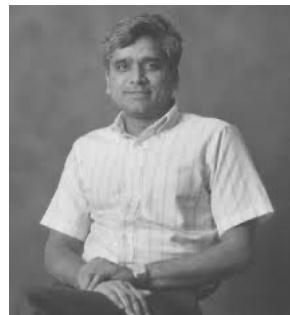
Dokaz 1) Če DFS najprej obišče C , bo obiskal vse $u \in C'$, preden bo zaključil v C .

2) Če DFS najprej obišče C' , ga bo zaključil, preden bo splet zacet s C .

Vidimo, da je vozlišče z max po-oznako v nekem izvoru kvoc. grafa. Ker potrebujemo pohore, obrnemo povezave v grafu in na obrnjencem grafu izvedemo DFS. Nato padajoče wedrino po po-oznakah in v tem vrstnem redu izvedemo DFS.



Shriram
*1950 Tel Aviv, Izrael



Ketan Mehta
*1943, Guntur, Indija