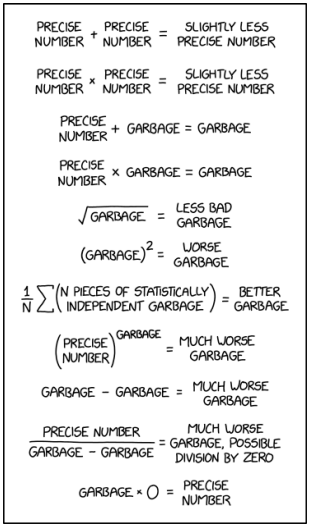
Uporabe DFS

Filip Koprivec

Fakulteta za Matematiko in Fiziko

April 2020

Od prejšnjič



https://xkcd.com/2295/

Današnje teme

Uporabe DFS
!Vsi grafi so usmerjeni!

1. Topološko urejanje
2. Krepko povezane komponente
3. Mostovi in prerezna vozlišča

Motivacija

https://www.spoj.com/problems/RPLA/

- Eloy is a hard worker man, however, he is constantly bullied by his superiors, molested by this, one day he was wondering in what "rank" you are, so you can bully the people with lower ranks, also to discover who can really bully Eloy!.
- Now, given the number of employees and the number of relations between them, Eloy need you to output the "rank" which employee is in, being 1 the "boss" (not bullied by anybody) and the employee who are in these ranks
- INPUT:
  test case starts with two integers N and R, the number of employees and the number of relations between them, the next R lines consists in two integers R1 and R2, meaning that "employee R1 is lower than employee R2's rank".
- OUTPUT: N lines, for each line you will output the rank of the employee and the employee itself, if there is the same rank for several employees, then output them lexicographically ordered (the first is the lower)

DFS

```
1: for u ∈ v.neigh do
2:     if !u.visited then
3:         explore(u)
4:     end if
5: end for
```

DFS 2.0

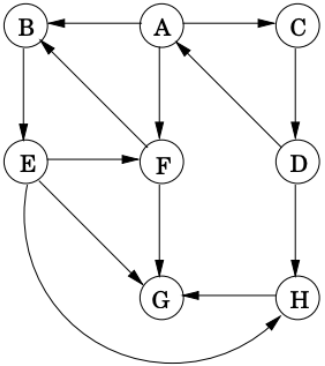
```
1: for u ∈ v.neigh do
2:     if !u.visited then
3:         previsit(u)
4:         explore(u)
5:         postvisit(u)
6:     end if
7: end for
```

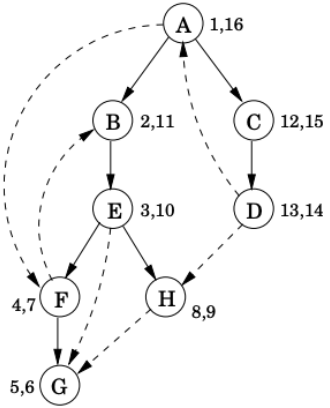
DFS 2.0

```
1: for u ∈ v.neigh do
2:     if !u.visited then
3:         previsit(u)
4:         explore(u)
5:         postvisit(u)
6:     end if
7: end for
```

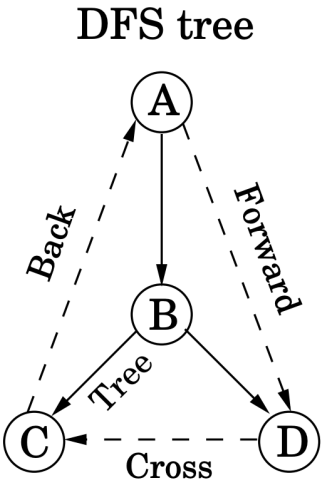
- Pred in po obisku lahko počnemo cel kup stvari
- Za začetek si zapomnimo vhodni (previsit) in izhodni (postvisit) čas

Figure 3.7 DFS on a directed graph.



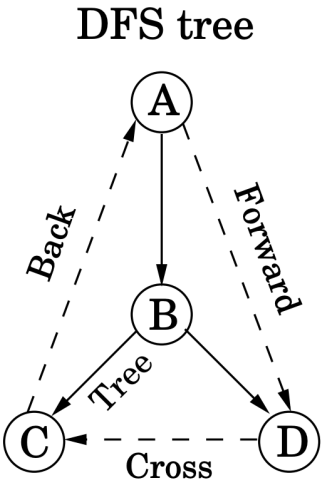


[DPV06]



Klasifikacija vozlišč

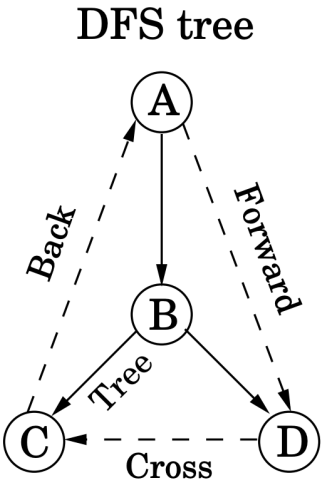
$$[\,]_u \quad [\,]_u \quad [\,]_v \quad [\,]_v$$



Klasifikacija vozlišč

$$[\,]_u \quad [\,]_u \quad [\,]_v \quad [\,]_v$$

$$[\,]_u \quad [\,]_v \quad [\,]_v \quad [\,]_u$$



Klasifikacija vozlišč

$$[\,]_u \quad [\,]_u \quad [\,]_v \quad [\,]_v$$

$$[\,[\,]_u \quad [\,]_v \quad]_v \quad]_u$$

$$[\,]_u \quad [\,]_v \quad [\,]_v \quad]_u$$

Topološko urejanje (toposort)

- Opraviti moramo množico opravil $[1, n]$.
- Nekatera opravila so pogoj za opravljanje ostalih.
- Odvisnost prikažemo z usmerjenim grafom (odpremo vrata, preden vstopimo).

Topološko urejanje (toposort)

- Opraviti moramo množico opravil $[1, n]$.
- Nekatera opravila so pogoj za opravljanje ostalih.
- Odvisnost prikažemo z usmerjenim grafom (odpremo vrata, preden vstopimo).
- Cikli?
- DAG (Directed acyclic graph)

Topološko urejanje (toposort)

- Opraviti moramo množico opravil $[1, n]$.
- Nekatera opravila so pogoj za opravljanje ostalih.
- Odvisnost prikažemo z usmerjenim grafom (odpremo vrata, preden vstopimo).
- Cikli?
- DAG (Directed acyclic graph)
- Vozlišča postavimo v tako zaporedje, da opravila opravimo v pravilnem vrstnem redu (ali povemo, da se ne da).

Urejanje

DFS

- Kdaj pokličemo *postvisit*?

Urejanje

DFS

- Kdaj pokličemo *postvisit*?
- Takoj, ko obiščemo vse neobiskane otroke.
- Katero vozlišče bo prvo poklicalo *postvisit*, katero naslednje, katero zadnje?
- Obratni vrstni red

Urejanje

DFS

- Kdaj pokličemo *postvisit*?
- Takoj, ko obiščemo vse neobiskane otroke.
- Katero vozlišče bo prvo poklicalo *postvisit*, katero naslednje, katero zadnje?
- Obratni vrstni red
- cikli

Urejanje

DFS

- Kdaj pokličemo *postvisit*?
- Takoj, ko obiščemo vse neobiskane otroke.
- Katero vozlišče bo prvo poklicalo *postvisit*, katero naslednje, katero zadnje?
- Obratni vrstni red
- cikli

postvisit(u) = $topo\_order.push\_back(u)$

Urejanje

DFS

- Kdaj pokličemo *postvisit*?
- Takoj, ko obiščemo vse neobiskane otroke.
- Katero vozlišče bo prvo poklicalo *postvisit*, katero naslednje, katero zadnje?
- Obratni vrstni red
- cikli

postvisit(u) = $topo\_order.push\_back(u)$
Več možnosti?

Urejanje

DFS

- Kdaj pokličemo *postvisit*?
- Takoj, ko obiščemo vse neobiskane otroke.
- Katero vozlišče bo prvo poklicalo *postvisit*, katero naslednje, katero zadnje?
- Obratni vrstni red
- cikli

postvisit(u) = $topo\_order.push\_back(u)$
Več možnosti?
$O(V + E)$, v praksi $O(E)$, pazi, E je lahko $O(V^2)$

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo
- Cikli?
- Več možnosti?

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo
- Cikli?
- Več možnosti?

DAG natanko tedaj, ko ima topološko ureditev

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo
- Cikli?
- Več možnosti?

DAG natanko tedaj, ko ima topološko ureditev
Topološko in leksikografsko

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo
- Cikli?
- Več možnosti?

DAG natanko tedaj, ko ima topološko ureditev
Topološko in leksikografsko
Najdaljše poti v drevesu

Khan's algorithm

Požrešno

- Katero opravilo lahko opravimo takoj?
- Za vsako vozlišče poznamo $v.indeg$ (število vhodnih povezav)
- Vzamemo prvo tako opravilo, ki je prosto $indeg == 0$
- Za vsakega soseda zmanjšamo $indeg$
- Ponovimo
- Cikli?
- Več možnosti?

DAG natanko tedaj, ko ima topološko ureditev
Topološko in leksikografsko
Najdaljše poti v drevesu
Ali je podoben kateremu izmed znanih algoritmov za urejanje?

Nazaj na problem

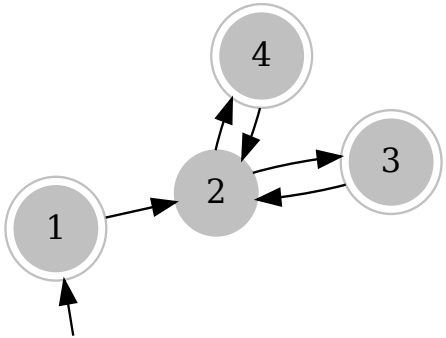
Topološko uredimo uslužbence
Dodatno jih uredimo še po abecedi (Khan)

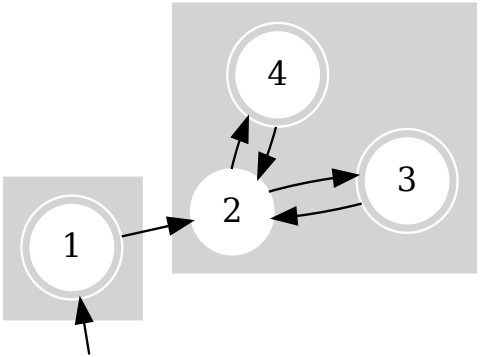
Nazaj na problem

Topološko uredimo uslužbence
Dodatno jih uredimo še po abecedi (Khan)
Najdaljše/najkrajše poti v DAG (topološko uredimo in se premikamo, pot je samo ena).

Nedeterministični Büchijev avtomat

https://putka-upm.acm.si/tasks/2018/2018_2kolo/buchi
Dolenjec Polde je ujel zlato ribico, ki uresničuje želje. Vedno si je želel, da bi večno hodil po gostilnah. Prijazna zlata ribica mu je željo tudi uresničila. Na tej točki pa se je Polde zamislil nad tem, kar je storil. Sedaj je nesmrten, preostanek večnosti pa bo hodil od ene gostilne do druge. V nobeni se ne bo mogel ustaviti. Lahko bo le vzel pijačo in šel dalje. Poldetova vas ima n stavb, označenih s števili od 1 do n, ki jih povezuje m enosmernih cest. V vasi je d gostiln, ki se nahajajo v stavbah z oznakami $f_1, f_2, \ldots, f_d$. Denimo, da Polde svojo pot začne pri stavbi q_0. Ali lahko vekomaj obiskuje gostilne, če hodi le po cestah? Bolj formalno, Poldetova pot po cestah mora biti neskončna in mora vsebovati neskončno obiskov ene ali več gostiln. Ni pomembno, katere gostilne obiskuje ali koliko drugih stavb obišče vmes.


4
3
2
1



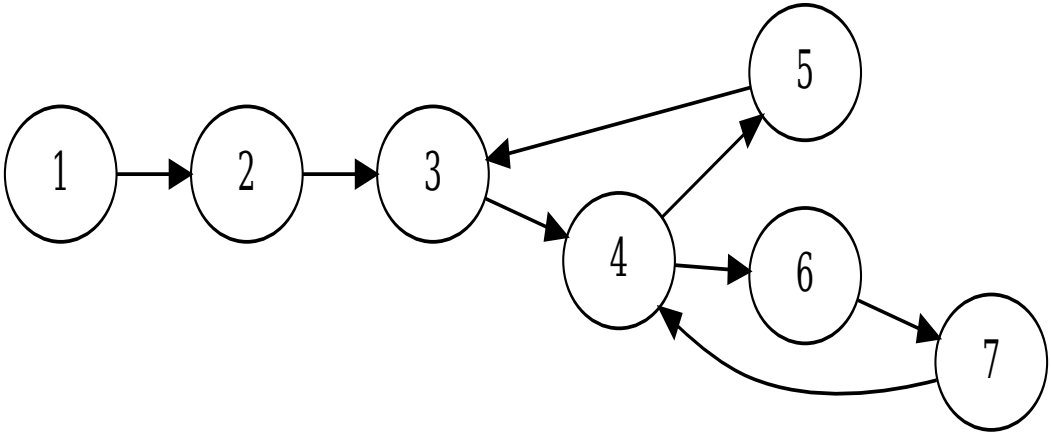
Polde se lahko večno giblje v (nekaterih) sivih območjih.
Siva območja so močno povezane komponente

SCC

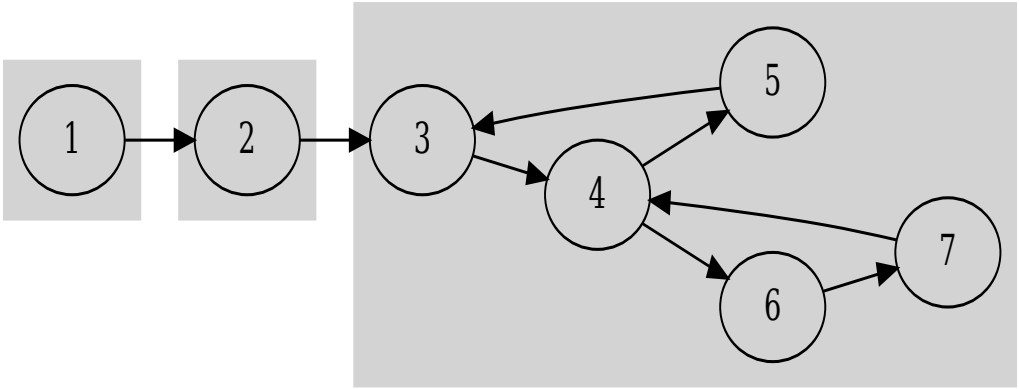
Definicija: Vsa vozlišča v komponenti so dostopna iz vseh ostalih (osamelci?, povezave nazaj?)
Iskanje močno povezanih komponent (Tarjanov algoritem)

SCC

Definicija: Vsa vozlišča v komponenti so dostopna iz vseh ostalih (osamelci?, povezave nazaj?)
Iskanje močno povezanih komponent (Tarjanov algoritem)
DFS drevo in vračanje nazaj



(1); (2); (3,4,5,6,7)


1
2
3
4
5
6
7

- Z DFS rekurzivno raziskujemo graf.
- Za vsako vozlišče si beležimo vhodni čas in najnižji vhodni čas vozlišč, ki se ga med DFS dotakne (lowlink).
- Glava komponente je nepomembna.
- Vozlišča, ki bodo v povezani komponenti so otroci lahko le otroci glave komponente (ali pa tvorijo svoje komponente).
- Če je moj vhodni čas enak najnižjemu času -> glava komponente (med preiskovanjem se ne vrnem nazaj).

- Z DFS rekurzivno raziskujemo graf.
- Za vsako vozlišče si beležimo vhodni čas in najnižji vhodni čas vozlišč, ki se ga med DFS dotakne (lowlink).
- Glava komponente je nepomembna.
- Vozlišča, ki bodo v povezani komponenti so otroci lahko le otroci glave komponente (ali pa tvorijo svoje komponente).
- Če je moj vhodni čas enak najnižjemu času -> glava komponente (med preiskovanjem se ne vrnem nazaj).
- Vsi DFS otroci so v moji komponenti.

- Z DFS rekurzivno raziskujemo graf.
- Za vsako vozlišče si beležimo vhodni čas in najnižji vhodni čas vozlišč, ki se ga med DFS dotakne (lowlink).
- Glava komponente je nepomembna.
- Vozlišča, ki bodo v povezani komponenti so otroci lahko le otroci glave komponente (ali pa tvorijo svoje komponente).
- Če je moj vhodni čas enak najnižjemu času -> glava komponente (med preiskovanjem se ne vrnem nazaj).
- Vsi DFS otroci so v moji komponenti. Ne čisto, nekateri so del svoje komponente.
- Sklad obiskanih otrok, ki ga praznim, ko najdem komponento.

- Z DFS rekurzivno raziskujemo graf.
- Za vsako vozlišče si beležimo vhodni čas in najnižji vhodni čas vozlišč, ki se ga med DFS dotakne (lowlink).
- Glava komponente je nepomembna.
- Vozlišča, ki bodo v povezani komponenti so otroci lahko le otroci glave komponente (ali pa tvorijo svoje komponente).
- Če je moj vhodni čas enak najnižjemu času -> glava komponente (med preiskovanjem se ne vrnem nazaj).
- Vsi DFS otroci so v moji komponenti. Ne čisto, nekateri so del svoje komponente.
- Sklad obiskanih otrok, ki ga praznim, ko najdem komponento.
- Ponovim za neobiskana vozlišča.

- Z DFS rekurzivno raziskujemo graf.
- Za vsako vozlišče si beležimo vhodni čas in najnižji vhodni čas vozlišč, ki se ga med DFS dotakne (lowlink).
- Glava komponente je nepomembna.
- Vozlišča, ki bodo v povezani komponenti so otroci lahko le otroci glave komponente (ali pa tvorijo svoje komponente).
- Če je moj vhodni čas enak najnižjemu času -> glava komponente (med preiskovanjem se ne vrnem nazaj).
- Vsi DFS otroci so v moji komponenti. Ne čisto, nekateri so del svoje komponente.
- Sklad obiskanih otrok, ki ga praznim, ko najdem komponento.
- Ponovim za neobiskana vozlišča.
  $O(E + V)$ zgolj en DFS in nekaj dodatnega spomina

Koda (Courtesy of Jure Slak)

`low = [0 for _ in range(n)]; dfs_num = [0 for _ in range(n)]; component = [-1 for _ in range(n)]`
`tarjan(u): low[u] = dfs_num[u] = time++` # začetni lowlink in čas obiska
`for` v `in` $G[u]$
    if $dfs\_num[v] == 0$ # Neobiskano v DFS
        tarjan(v) # Običajen DFS
    `endif` # Konec DFS za enega otroka
    if $dfs\_num[v] != -1$ # Pod nami in še ni del komponente
        $low[u] = min(low[u], low[v])$ # Pogledamo kako daleč sežemo s tem otrokom
    `endif`
`endfor`
`if` $low[u] == dfs\_num[u]$ # Ne sežemo bolj nazaj (nikjer se nismo popravili)
Poberemo iz sklada vse, do vključno u in jim nastavimo dfs_num = -1 (niso več v skladu, a so obiskani).
Vsi te so v povezani komponenti

Animacija

Visualgo SCC

Nazaj k Poldetu

Iz originalnega grafa dobimo krepko povezane komponente

Nazaj k Poldetu

Iz originalnega grafa dobimo krepko povezane komponente
Pogledamo, če se lahko sprehodi od začetka do neke katerekoli komponente, ki vsebuje gostilno

Nazaj k Poldetu

Iz originalnega grafa dobimo krepko povezane komponente
Pogledamo, če se lahko sprehodi od začetka do neke katerekoli komponente, ki vsebuje gostilno
Ali je dobro, če je komponenta osamelec?

Motivacija

Neusmerjeni grafi
Razdelijo graf na več komponent
Ključno pri problemih deli in vladaj

UVA 315

A Telephone Line Company (TLC) is establishing a new telephone cable network. They are connecting several places numbered by integers from 1 to N. No two places have the same number. The lines are bidirectional and always connect together two places and in each place the lines end in a telephone exchange. There is one telephone exchange in each place. From each place it is possible to reach through lines every other place, however it need not be a direct connection, it can go through several exchanges. From time to time the power supply fails at a place and then the exchange does not operate. The officials from TLC realized that in such a case it can happen that besides the fact that the place with the failure is unreachable, this can also cause that some other places cannot connect to each other. In such a case we will say the place (where the failure occured) is critical. Now the officials are trying to write a program for finding the number of all such critical places. Help them.

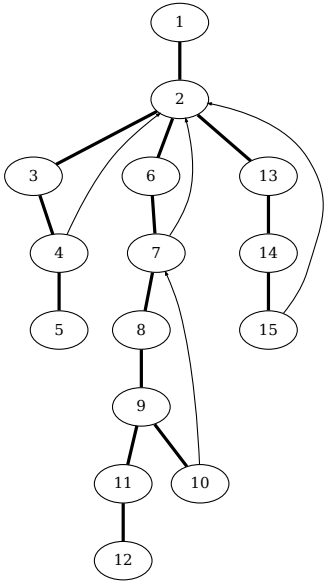
Naloga

Poiskati moramo prerezna vozlišča, vozlišča, ki jih odstranimo in zaradi tega graf razpade na več komponent

Naloga

Poiskati moramo prerezna vozlišča, vozlišča, ki jih odstranimo in zaradi tega graf razpade na več komponent
V DFS drevesu gledamo povratne povezave
Klasifikacija:

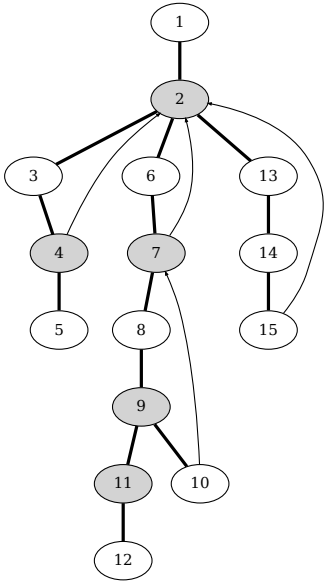
- Vse poti med u in v vodijo čez vozlišče c
- Vozlišče c je koren DFS drevesa, ki ima dve ločeni poddrevesi.



Naloga

Poiskati moramo prerezna vozlišča, vozlišča, ki jih odstranimo in zaradi tega graf razpade na več komponent
V DFS drevesu gledamo povratne povezave
Klasifikacija:

- Vse poti med u in v vodijo čez vozlišče c
- Vozlišče c je koren DFS drevesa, ki ima dve ločeni poddrevesi.

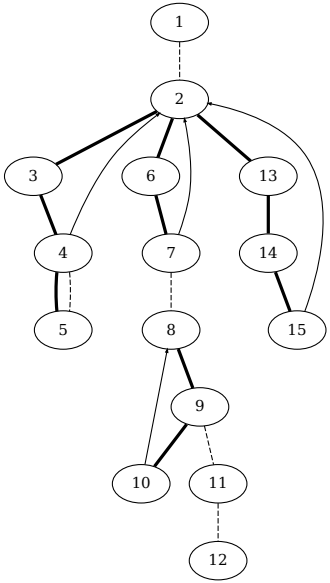


Algoritem

Beležimo si najbolj zgodno doseženo vozlišče `low = [0 for _ in range(n)];`
`dfs_num = [-1 for _ in range(n)];parent[-1 for _ in range(n)]` # začetni lowlink in čas obiska
`articualtion(u): low[u] = dfs_num[u] = time++; ++children;`
`for` v `in` $G[u]$
    if $dfs\_num[v] == -1$ # Neobiskano v DFS
        $parent[v] = u; ++children$
        $articulation(v)$ # Običajen DFS
        $low[u] = min(low[u], low[v])$ # posodobimo low
        `if` $parent[u] != -1 \&\& children > 1$ art = True
        `if` $parent[u] != -1 \&\& dfs_num[u] < low[v]$ art= True # Če ne pridem višje
        `endif`
    `elseif` $v != parent[v]$ # Predhodnik (backlink), vendar ne direktni
        $low[u] = min(low[u], dfs\_num[v])$ # Ampak zgolj dfs_num (že obiskan)
    `endif`
`endfor`

Mostovi

Mostovi
Povezave, ki jih odstranimo, da graf razpade na več delov
Ista ideja, le malo bolj strogi smo pri neenačaju
$dfs\_num[u] <= low[v]$ (Vsak most ima dve presečni vozlišči)
V resnici se to ponavadi implementira kar v isti metodi (preveriš oboje)
$O(E + V)$ zgolj en DFS in nekaj dodatnega spomina



IOI 2017 Simurgh (https://ioinformatics.org/files/ioi2017problem5.pdf)

In Persia there are cities, labeled from 0 to n, and m two-way roads, labeled from 0 to $m-1$. Each road connects a pair of distinct cities. Each pair of cities is connected by at most one road. Some of the roads are royal roads used for travels by royals. Zal's task is to determine which of the roads are the royal roads.

Zal has a map with all the cities and the roads in Persia. He does not know which of the roads are royal, but he can get help from Simurgh, the benevolent mythical bird who is Zal's protector. However, Simurgh does not want to reveal the set of royal roads directly. Instead, she tells Zal that the set of all royal roads is a golden set. A set of roads is a golden set if and only if:

it has exactly n roads, and

for every pair of cities, it is possible to reach one from the other by traveling only along the roads of this set.

Furthermore, Zal can ask Simurgh some questions. For each question: 1. Zal chooses a golden set of roads, and then 2. Simurgh tells Zal how many of the roads in the chosen golden set are royal roads. Your program should help Zal find the set of royal roads by asking Simurgh at most q questions. The grader will play the role of Simurgh.

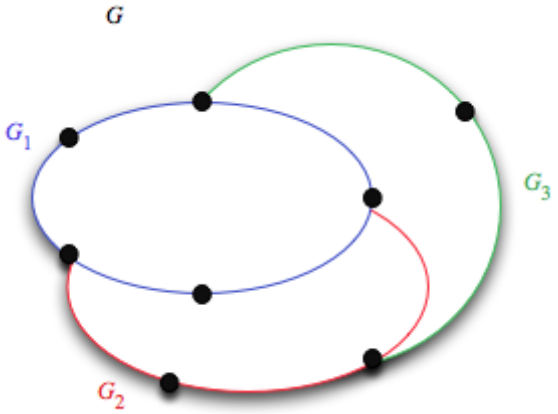
Subtaske se da rešiti relativno požrešno

Subtaske se da rešiti relativno požrešno
Najdemo mostove v grafu -> te so gotovo del lepih cest.

Subtaske se da rešiti relativno požrešno
Najdemo mostove v grafu -> te so gotovo del lepih cest.
Kaj vemo o ostalih komponentah?

Subtaske se da rešiti relativno požrešno
Najdemo mostove v grafu -> te so gotovo del lepih cest.
Kaj vemo o ostalih komponentah?
Ušesna dekompozicija

Subtaske se da rešiti relativno požrešno
Najdemo mostove v grafu -> te so gotovo del lepih cest.
Kaj vemo o ostalih komponentah?
Ušesna dekompozicija



[Wik20]

Dodatno

Dvosmerni BFS

Dodatno

Dvosmerni BFS
Gremo iz obeh strani

Dodatno

Dvosmerni BFS
Gremo iz obeh strani

$$1 + B + B^2 + B^3 + \dots B^k$$

$$2 + 2B + 2B^2 + 2B^3 + \dots 2B^{\frac{k}{2}}$$

In nekaj iskanja po množici

Dodatno

Najdaljša pot po drevesu

Domača naloga

Saj bo šlo, če se kjer zatakne, napišite

- Fox and names: Ali smo vzeli kaj kar omogoča drugačno urejanje?
- Checkposts: Ali so checkposti kako povezani?
- Bosses: Na katera mesta gotovo postavimo mostove?
- Table compression: Pametno sestavimo graf, v pravilnem vrstnem redu ...
- Tourist reform: Ali so katere ceste ključne?

 Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh Vazirani, *Algorithms*, 1 ed., McGraw-Hill, Inc., USA, 2006.

 Wikipedia contributors, *Ear decomposition — Wikipedia, the free encyclopedia*, 2020, [Online; accessed 18-April-2020].