Najkrajše poti v grafih

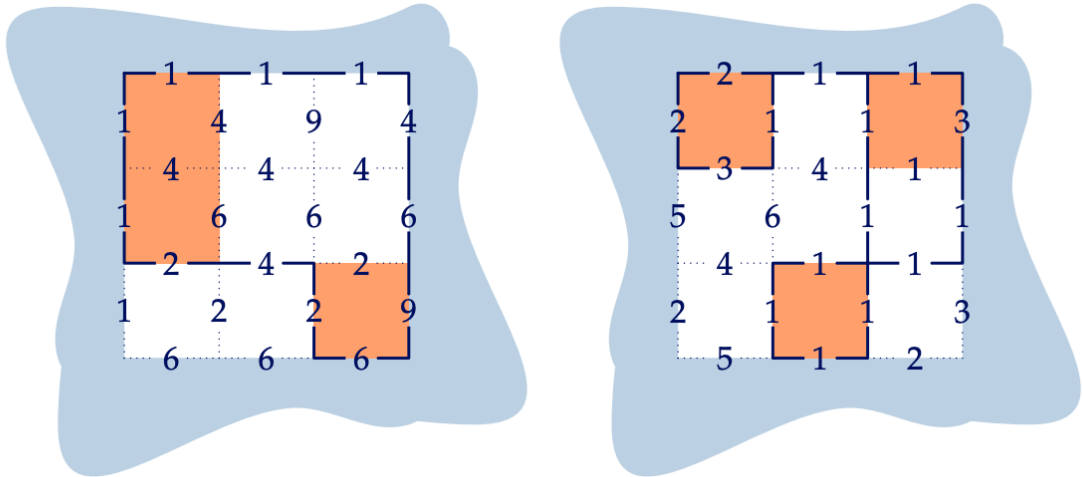
Filip Koprivec

Fakulteta za Matematiko in Fiziko

Februar 2019

The Wall - IOI 2014

- Na mreži imamo vasice, ki jih želimo zaščititi, tako da okoli njih napeljemo zid, ki je povezan s prestolnico (zgoraj levo)
- Cena grajenja zidu je različna, iščemo najcenejšega

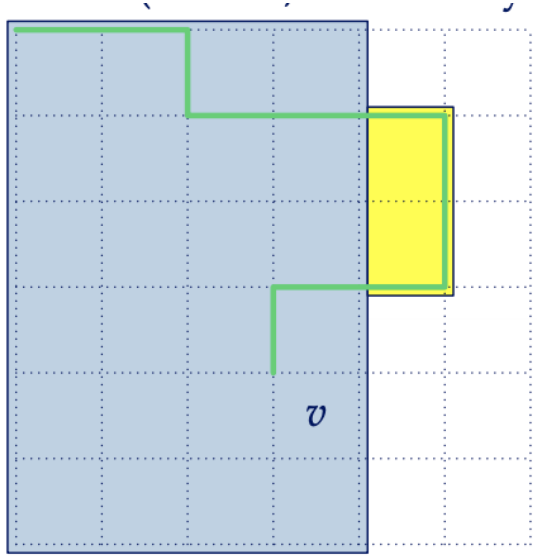


Poenostavimo problem

- Kaj narediti, če so vse povezave enako težke/lahke
- Napnemo (čudno) elastiko

Poenostavimo problem

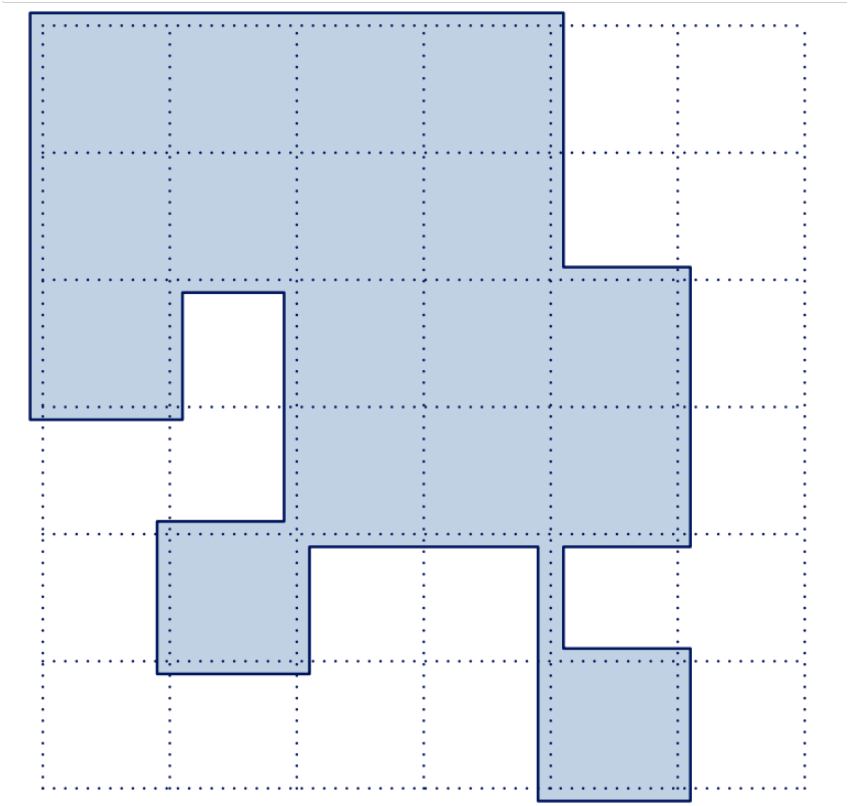
- Kaj narediti, če so vse povezave enako težke/lahke
- Napnemo (čudno) elastiko



- Enaka ideja velja pri uteženih poteh

Rešitev

- Iščemo najkrajšo pot, ki ne seka najkrajših poti



Najkrajša pot

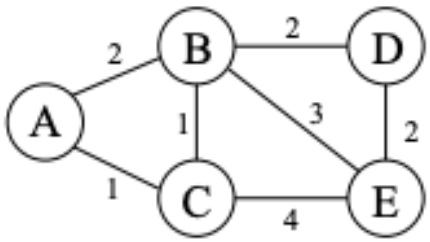
- Do sedaj poznamo BFS

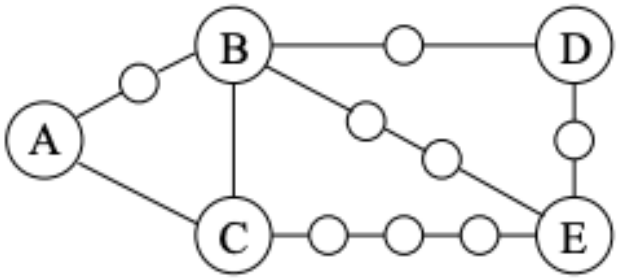
Najkrajša pot

- Do sedaj poznamo BFS
- Zakaj deluje, ga lahko kako popravimo?

Najkrajša pot

- Do sedaj poznamo BFS
- Zakaj deluje, ga lahko kako popravimo?
- Namesto uteži w postavimo $w - 1$ dodatnih vozlišč
- Nadaljujemo z bfs





Dijkstra

- Koda

Dijkstra

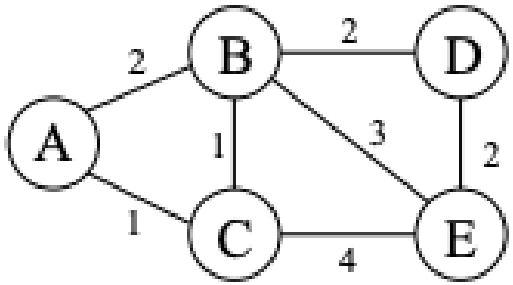
- Koda
- Kaj uporabimo za iskanje naslednjega elementa?

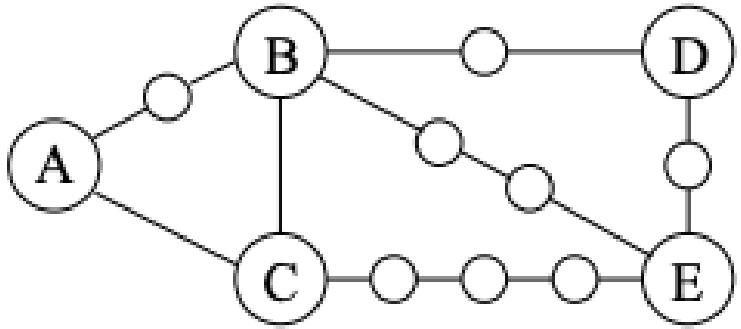
Dijkstra

- Koda
- Kaj uporabimo za iskanje naslednjega elementa?
- Priority queue bo dovolj dober
- `if cur < dist[`v`]` ali pa vodimo že obiskane
- Časovna zahtevnost

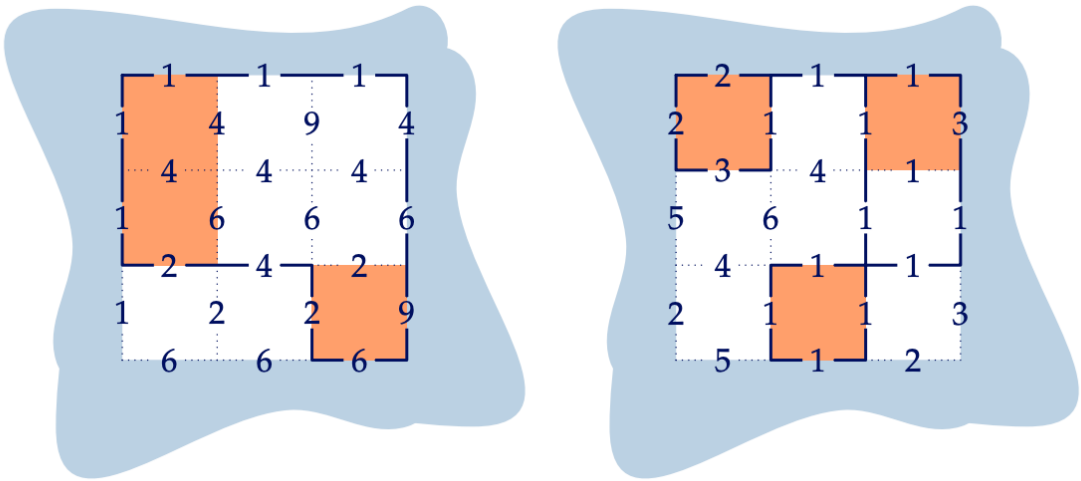
Dijkstra

- Koda
- Kaj uporabimo za iskanje naslednjega elementa?
- Priority queue bo dovolj dober
- `if cur < dist[`v`]` ali pa vodimo že obiskane
- Časovna zahtevnost
- $|\mathrm{E}| \log |\mathrm{V}|$

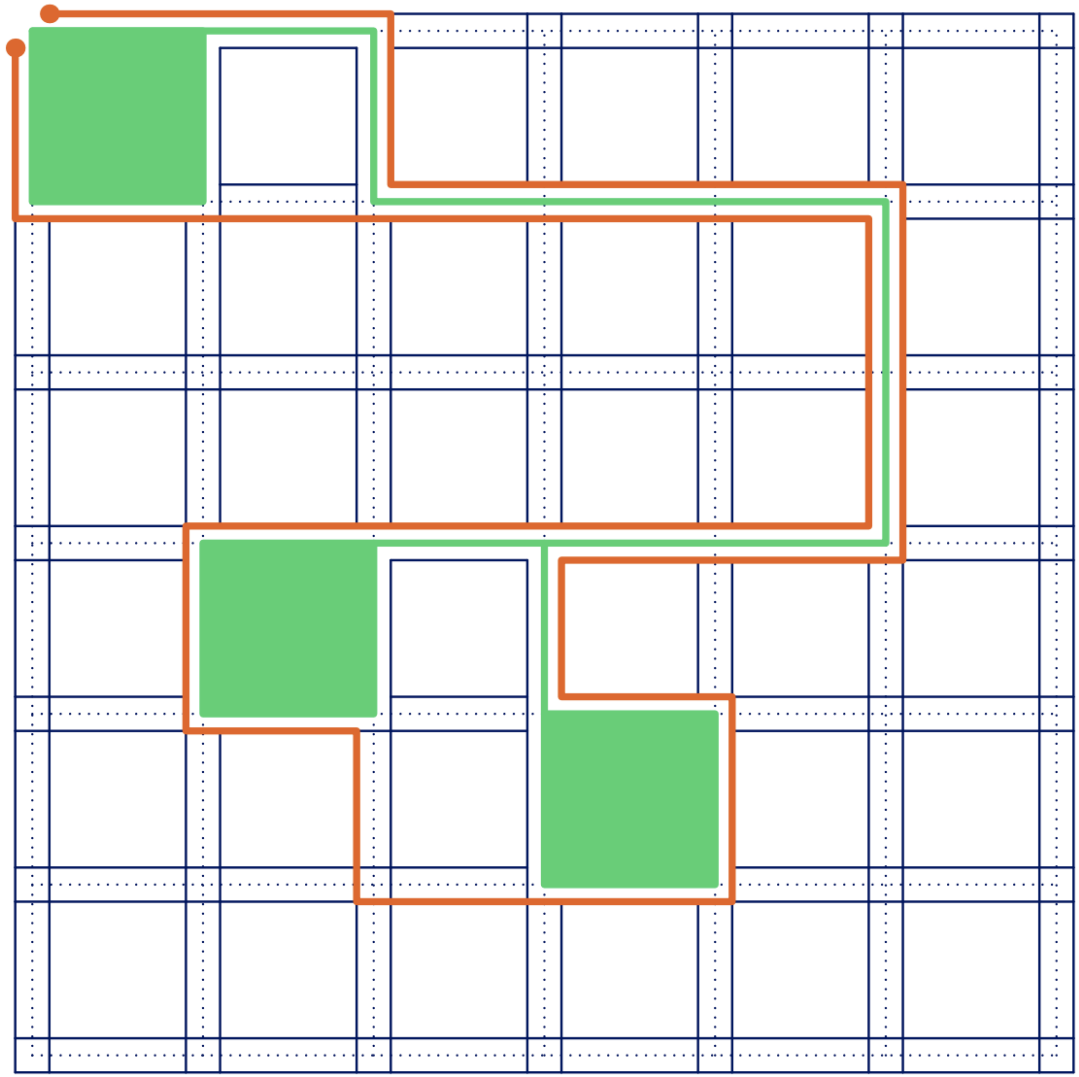

A
B
C
D
E
2
2
1
3
2
1
4

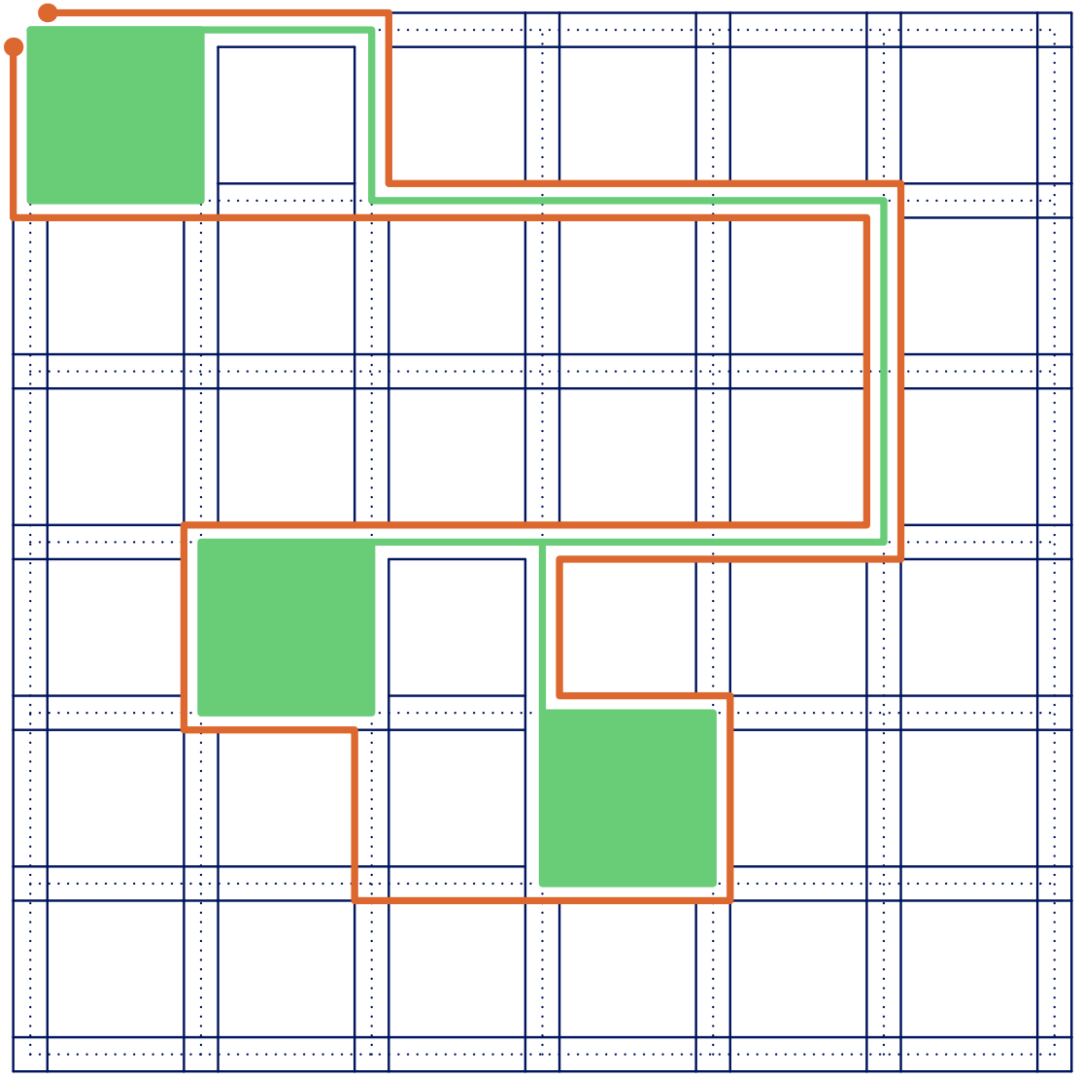

A
B
C
D
E

Naloga



- Najdemo najkrajšo pot od prestolnice do vrha vsake vasi
- Popravimo graf in najdemo najkrajšo pot spet nazaj, ki ne seka te poti





- Kolikokrat poženemo Dijkstro ?
- $nm \log(nm)$

Ključna opazka

- Teže so pozitivne (nenegativne)
- Kaj pa če dovoljujemo negativne uteži

Bellman-Ford

- Ne smemo računati, da je najkrajša razdalija tista, po kateri smo prišli prej
- Popravljanje moramo narediti malo bolj zvito...
- Razmišljamo o *dolžini* povezav
- Najlažjo povezavo iščemo izmed vseh dolžine največ k

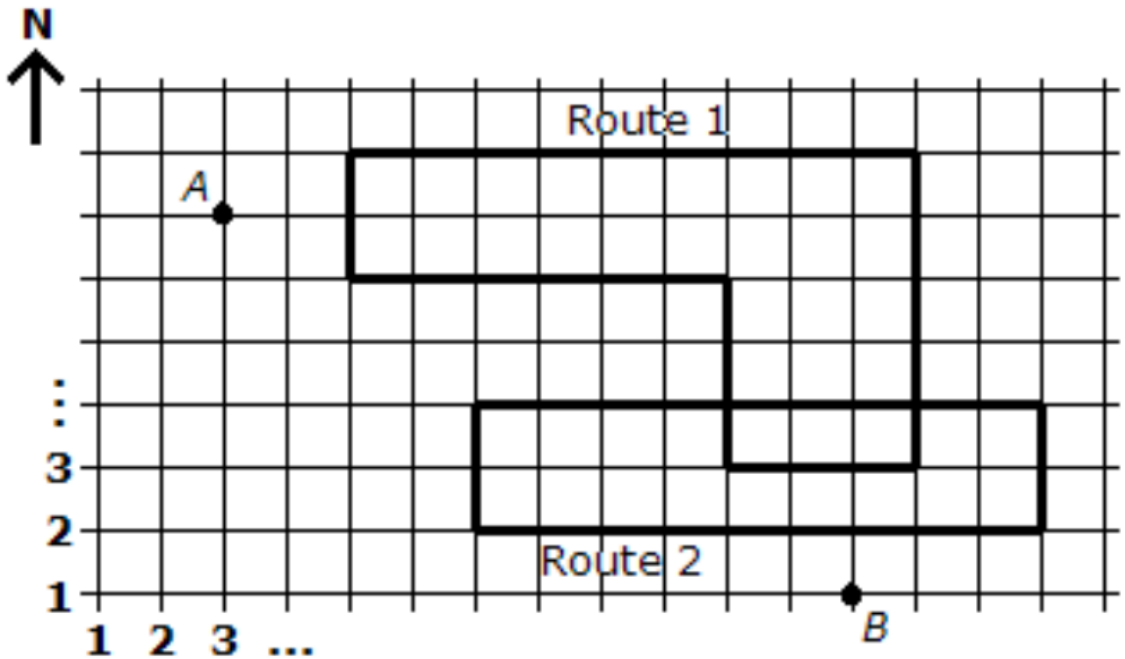
- Koda
- Kakšno težavo nam lahko dajo negativne povezave?

- Koda
- Kakšno težavo nam lahko dajo negativne povezave?
- Poženemo še enkrat
- Časovna zahtevnost

- Koda
- Kakšno težavo nam lahko dajo negativne povezave?
- Poženemo še enkrat
- Časovna zahtevnost
- $|V||E| \rightarrow$ Zagotovo slabše kot Dijkstra

Busses - IOI 2008 (Pripravljalna)

- Imamo posamezne avtobusne povezave na (veliki) mreži (in ceno vožnje z njimi), $R \leq 100, 4 \leq N_i \leq 50$
- Lahko se tudi malo sprehodimo, a največ do razdalije $D \leq 300$
- Kako lahko najceneje pridemo do cilja?



- Bi šlo, če ne smemo hoditi, in se povezave ustrezno sekajo?
- Ali znamo za posamezni progi preveriti, ali smemo prestopati brez hoje

- Bi šlo, če ne smemo hoditi, in se povezave ustrezno sekajo?
- Ali znamo za posamezni progi preveriti, ali smemo prestopati brez hoje
- Dijkstra glede na ceno prog

- Bi šlo, če ne smemo hoditi, in se povezave ustrezno sekajo?
- Ali znamo za posamezni progi preveriti, ali smemo prestopati brez hoje
- Dijkstra glede na ceno prog
- Ali znamo izračunati njuno razdalijo
- $(RN)^2$

Trik

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), v \in \mathcal{G}, 0 \leq i \leq D\}$
- Vozlišči povežemo če sta v i narazen ravno za njuno oddaljenost na mreži
- Število vozlišč: RD
- Števiko povezav

Trik

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), v \in \mathcal{G}, 0 \leq i \leq D\}$
- Vozlišči povežemo če sta v i narazen ravno za njuno oddaljenost na mreži
- Število vozlišč: RD
- Števiko povezav
- $R^2D \rightarrow$ vsako vozlišče je povezano z R+1 drugimi (lahko v različnih nivojih)

Trik

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), v \in \mathcal{G}, 0 \leq i \leq D\}$
- Vozlišči povežemo če sta v i narazen ravno za njuno oddaljenost na mreži
- Število vozlišč: RD
- Števiko povezav
- $R^2D \rightarrow$ vsako vozlišče je povezano z R+1 drugimi (lahko v različnih nivojih)
- Pogost trik: Odklepanje sob, iskanje ključev...
- Množenje povezav

C. Pony Express - Code Jam 2017 Round 1B

- Imamo N mest in dolžine povezav med njimi
- V vsakem mestu nas čaka svež konj, ki teče z določeno hitrostjo in se zgrudi od izčrpanosti po določeni dolžini poti
- Konje lahko zamenjamo v mestu (ali pa ne)

C. Pony Express - Code Jam 2017 Round 1B

- Imamo N mest in dolžine povezav med njimi
- V vsakem mestu nas čaka svež konj, ki teče z določeno hitrostjo in se zgrudi od izčrpanosti po določeni dolžini poti
- Konje lahko zamenjamo v mestu (ali pa ne)
- Seveda nas zanima najhitrejši način kako pridemo iz enega v drugo mesto

Floyd-Warshall

- Jap te algoritmi imajo vedno imena po ljudeh....
- Se še spomnimo dinamičnega programiranja?

Floyd-Warshall

- Jap te algoritmi imajo vedno imena po ljudeh....
- Se še spomnimo dinamičnega programiranja?
- Delajmo po kosih
- Najkrajšo pot do vozlišča posodobimo z najkrajšimi potmi njenih sosedov

Floyd-Warshall

- Jap te algoritmi imajo vedno imena po ljudeh....
- Se še spomnimo dinamičnega programiranja?
- Delajmo po kosih
- Najkrajšo pot do vozlišča posodobimo z najkrajšimi potmi njenih sosedov
- Počasi povečujemo katere sosede vzamemo

```
sp[i, j, k] = min(
    sp[i, j, k-1],
sp[i, k, k-1] + sp[k, j, k-1] )
```

Negativni cikli?

Direktno prekopirano

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
let next be a |V| × |V| array of vertex indices initialized to null

procedure FloydWarshallWithPathReconstruction ()
   for each edge (u,v)
      dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
      next[u][v] ← v
   for k from 1 to |V| // standard Floyd-Warshall implementation
      for i from 1 to |V|
         for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j] then
               dist[i][j] ← dist[i][k] + dist[k][j]
               next[i][j] ← next[i][k]

procedure Path(u, v)
   if next[u][v] = null then
       return []
   path = [u]
   while u ≠ v
       u ← next[u][v]
       path.append(u)
   return path
```

Nazaj k nalogi

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti $\rightarrow$ FW

Nazaj k nalogi

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti $\rightarrow$ FW
- Kako pa je s konji

Nazaj k nalogi

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti $\rightarrow$ FW
- Kako pa je s konji
- Ali lahko kako popravimo graf

Nazaj k nalogi

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti $\rightarrow$ FW
- Kako pa je s konji
- Ali lahko kako popravimo graf
- Naredimo nov graf, tako da povezave utežimo kot čas potovanja kjer to smemo
- Ponovno uporabimo FW

Crocodile - IOI 2011

- Nahajamo se v labirintu podzemnih rovov(različnih dolžin), in soban. Iz nekaterih soban lahko pobegnemo na površje
- Zlobni vratar lahko po želji odpira in zapira poljuben rov in nam s tem preprečiti (ali pa vsaj zavlačevati) pobeg
- Zanima nas najhitrejši mogoč čas pobega (če se to sploh da)

- Prvi namig pride iz manjših testnih primerov
- Kaj če ima graf drevesno strukturo

- Prvi namig pride iz manjših testnih primerov
- Kaj če ima graf drevesno strukturo
- Če želimo zavlačevati, želimo pokriti najhitrejši vhod
- Kaj pa če zadeva ni drevo?

- Prvi namig pride iz manjših testnih primerov
- Kaj če ima graf drevesno strukturo
- Če želimo zavlačevati, želimo pokriti najhitrejši vhod
- Kaj pa če zadeva ni drevo?
- Premikamo se z Dijkstro od izhoda in sproti posodabljamo tudi drugi najboljši čas
- Pognati je potrebno celotno Dijkstro

Domača naloga

1. Dijkstra?
   Točno to, kar piše v imenu, poskusite najti najhitrejšo implementacijo, kar se tiče kode
   Če še niste, vsaj enkrat v življenju se spodobi, da kopico implementirate sami
   Za malo igranja s kazalci, implementirajte tudi `decrease_key`
2. Roads in Berland
   Ali je kateri izmed predstavljenih algoritmov poskušal preveriti, kako posamezna povezava 'skrajša' trenutno najkrajšo pot?
3. Jzzhu and Cities
   Zvito glej poti nazaj in pazi, kdaj lahko $<$ spremeniš v $\leq$