

Najkrajše poti v grafih

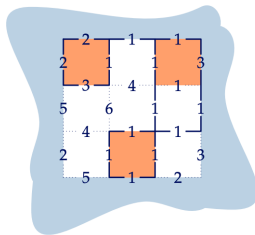
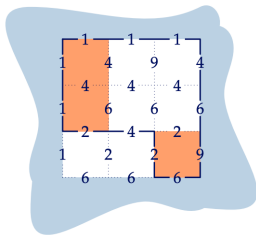
Filip Koprivec

AFLabs, FMF

Jan 2023

The Wall - CEOI 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

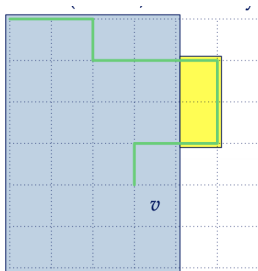


Poenostavimo problem

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

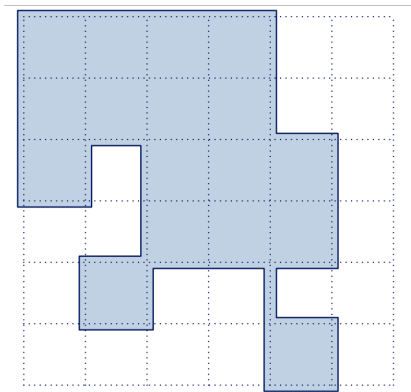
Poenostavimo problem

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



- Enaka ideja velja pri uteženih poteh

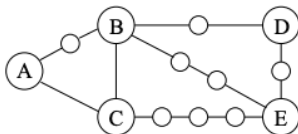
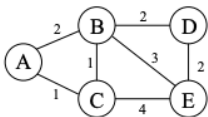
- Iščemo najkrajšo pot, ki ne seka najkrajših poti



- Do sedaj poznamo BFS

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

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

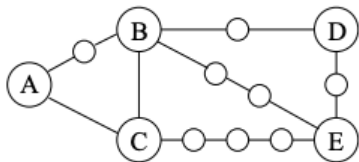
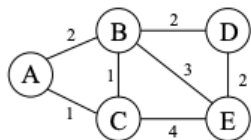


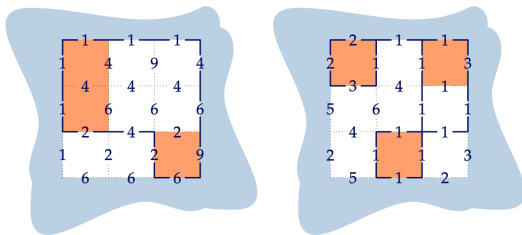
- Koda

- Koda
- Kaj uporabimo za iskanje naslednjega elementa?

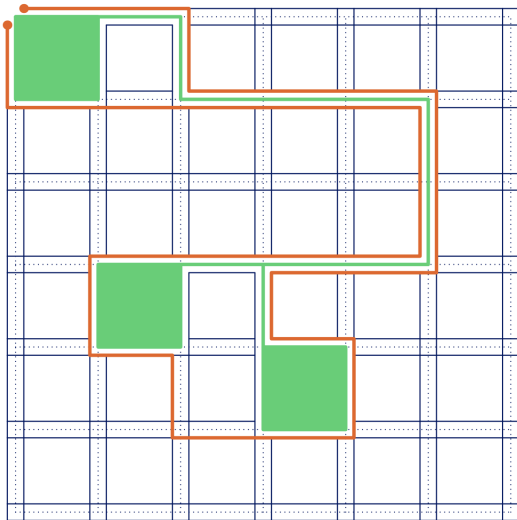
- Koda
- Kaj uporabimo za iskanje naslednjega elementa?
- Priority queue bo dovolj dober
- `if cur < dist[v]` ali pa vodimo že obiskane
- Časovna zahtevnost

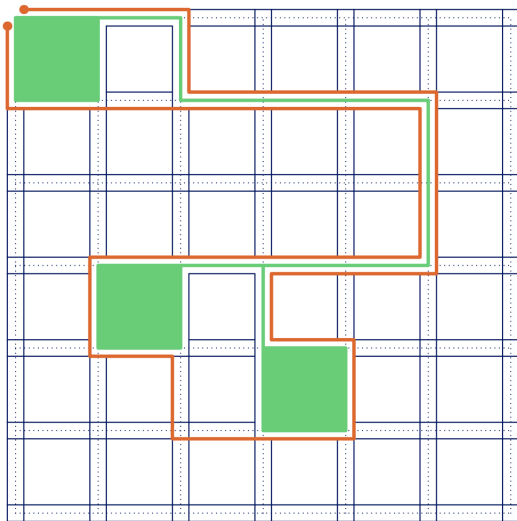
- Koda
- Kaj uporabimo za iskanje naslednjega elementa?
- Priority queue bo dovolj dober
- `if cur < dist[v]` ali pa vodimo že obiskane
- Časovna zahtevnost
- $|E| \log |V|$





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





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

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

- 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

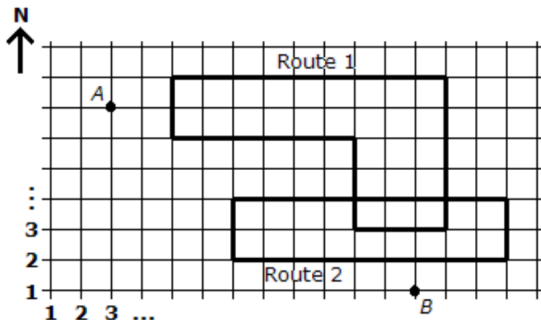
- Koda
- Kakšno težavo nam lahko dajo negativne povezave?

- 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 progji 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 prog preveriti, ali smemo prestopati brez hoje
- Dijkstra glede na ceno prog
- Ali znamo izračunati njuno razdalijo
- $(RN)^2$

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), -1 \leq v \leq R + 1, 0 \leq i \leq D\}$
- Vozlišči povežemo če sta v in i narazen ravno za njuno oddaljenost na mreži
- Število vozlišč: RD
- Število povezav

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), -1 \leq v \leq R + 1, 0 \leq i \leq D\}$
- 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)

- Lahko prehodimo največ D cest
- Grafe zložimo enega nad drugega
- $V' = \{(v, i), -1 \leq v \leq R + 1, 0 \leq i \leq D\}$
- Vozlišči povežemo če sta v i narazen ravno za njuno oddaljenost na mreži
- Število vozlišč: RD
- Število 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

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

- 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

- 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

$$\begin{aligned} sp[i, j, k] = \min(\\ & sp[i, j, k-1], \\ & sp[i, k, k-1] + sp[k, j, k-1]) \end{aligned}$$

Negativni cikli?

```
let dist be a  $|V| \times |V|$  array of minimum distances initialized to  $\infty$  (infinity)
let next be a  $|V| \times |V|$  array of vertex indices initialized to null

procedure FloydWarshallWithPathReconstruction ()
  for each edge (u,v)
    dist[u][v]  $\leftarrow$  w(u,v) // the weight of the edge (u,v)
    next[u][v]  $\leftarrow$  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]  $\leftarrow$  dist[i][k] + dist[k][j]
          next[i][j]  $\leftarrow$  next[i][k]

procedure Path(u, v)
  if next[u][v] = null then
    return []
  path = [u]
  while u  $\neq$  v
    u  $\leftarrow$  next[u][v]
    path.append(u)
  return path
```

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti → FW

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

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

- Ali znamo hitro izračunati najkrajše povezave med posameznimi mesti → 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 poljubni 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 posodabljammo tudi drugi najboljši čas
- Pognati je potrebno celotno Dijkstro

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