NAJKRAJŠE POTI

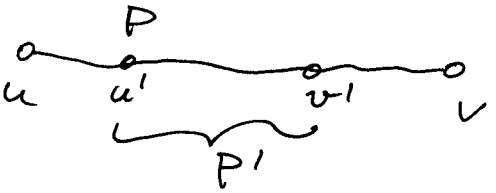
shortest paths

G utežen, usmerjen graf

P pot v G, dolžina poti P: $w(P) := \sum_{e \in E(P)} w(e)$ (utež, dolžina povezave)

lahko predstavljajo razdaljo, čas, ceno, hrami, ...

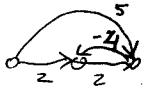
Lastnosti:

① 

P najkrajša pot od u do v

Potem je tudi P' najkrajša pot od u' do v'.

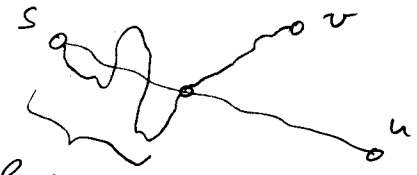
PAZI: NE SME BITI NEG. CIKLOV



②

s ... source (izvor)

najkrajše poti od ene do vseh ostalih točk lahko predstavimo z drevesom drevo najkrajših poti

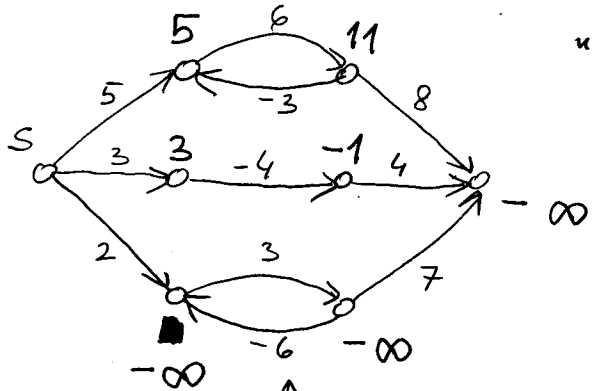


Drevo vsebuje točke, dosegljive iz s.

Uporabljamo za predstavitev najkrajših poti.

lahko nadomestimo ... (V splošnem med parom točk obstaja več najkrajših poti.)

Negativne uteži: v splošnem imajo nekatere povezave lahko negativne uteži



"Težave": negativni cikli

∃ palj. "kratke" poti

↑ cikel negativne dolžine, lahko palj. "zakrožimo"

(govorimo o "poteh", saj so najkrajši sprehodi, če ni neg. ciklov, nes poti)

Najkrajše poti od izvora do vseh ostalih točk

single source shortest paths

~~Neusmerjen~~, neutežen graf:

naredimo pregled BFS z začetkom v izvoru

Usmerjen, utežen graf, nenegativne uteži (Dijkstrov postopek)

Ideja algoritma: podobno kot Primov postopek (požrešno)

Požrešno gradimo drevo najkrajših poti. Začnemo z izvorom in na vsakem koraku dodamo tč, ki je prek drevesa in ene povezave zunaj njega najbližja izvoru (le kriterij dodajanja je drugačen kot pri Primu).

Dijkstra(G, s)

(s ← začetna točka)

w ... uteži na povezavah (morajo biti ≥ 0)

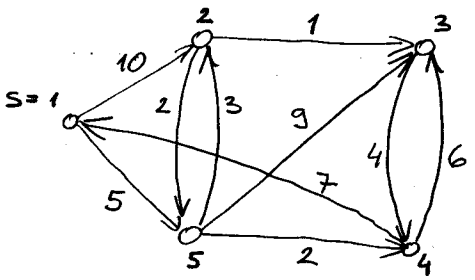
d ... razdalje od s do drugih točk (med postopkom: razd. po drevesu + 1 povezava)

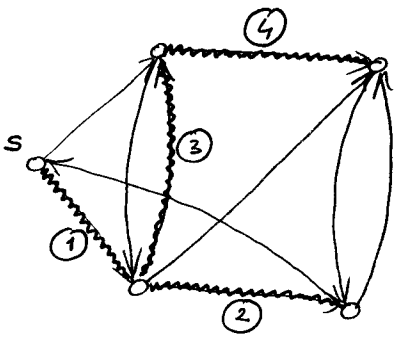
π ... oče v drevesu najkrajših poti

```
for u ∈ V(G) do            }
   d[u] = ∞                }
   π[u] = nil              }  inicializacija
d[s] = 0                   }
Q = V(G)                      struktura, v kateri hranimo točke, ki še niso v drevesu
while Q ≠ ∅ do
   u = min Q                  u je tč z najmanjšim d v Q; odstranimo jo iz Q
   for v ∈ Adj(u) do          pogledamo, ali so se s privzemom u razdalje
      if d[v] > d[u] + w(u,v) then     za sosede kaj zmanjšale
         d[v] = d[u] + w(u,v)
         π[v] = u
```

Zgled.

Opomba. Pri prvem min vedno izlenemo začetno tč s.





dobljeno drevo najkrajših poti

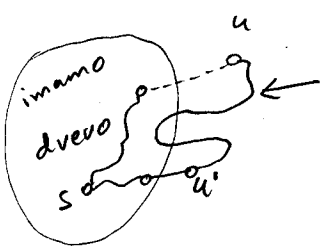
		d					Π				
inicializacija		0	∞	∞	∞	∞	–	–	–	–	–
izlenemo s		0	10	∞	∞	5	–	1	–	–	1
izlenemo tč. 5	①	0	8	14	7	5	–	5	5	5	1
izlenemo tč. 4	②	0	8	13	7	5	–	5	4	5	1
izlenemo tč. 2	③	0	8	9	7	5	–	5	2	5	1
izlenemo tč. 3	④	0	8	9	7	5	–	5	2	5	1
		razdalje					opis drevesa				

Pravilnost algoritma:

Na vsakem koraku imamo zgrajen del drevesa najkrajših poti v pregledanem delu grafa.

Na začetku to velja (le s v drevesu).

Poglejmo razširjanje:



recimo, da je to najkrajša pot

u' prva tč, ki je zunaj pregledanega dela

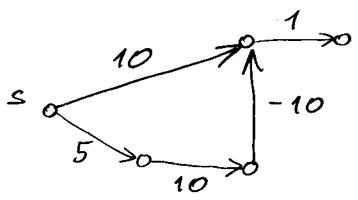
$$dist(s,u) = dist(s,u') + \underbrace{dist(u',u)}_{\geq 0 \text{ ni negativnih uteži}} \geq d[u]$$

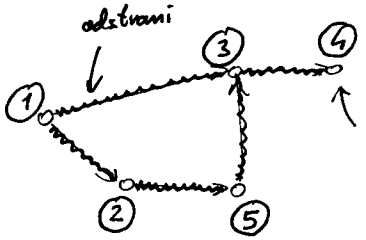
$$dist(s,u') \overset{\text{def. } d}{=} d[u']$$

$$d[u'] \leq d[u]$$

$$\Rightarrow$$ Tudi izbrana pot je ena od najkrajših

Zgled. Dijkstrov postopek ne deluje pravilno na grafih z negativnimi utežmi





drevo je pravo, razdalja v tej točki ni prava

(Seveda lahko dalinmo tudi naravno drevo.)

Iz dokaza se jasno vidi, kaj je narobe.

Implementacija: (če nas zanima le najkrajša pot do nekaj točk, lahko algoritem predčasno končamo)

① Naivna implementacija (slabo)

vrsnaja razlika po točkah

"za Q" { vsakič pregledamo vse povezave med pregledanim in nepregledanim delom in izberemo krajišče, ki je najbližje s } $O(|V| \cdot |E|)$

② Običajna implementacija

za strukturo Q uporabimo običajno tabelo

razlika po tč ... $O(|V|)$ } $O(|V|^2)$ } $O(|V|^2)$

min ... $O(|V|)$

popravki d in π v celoti $O(|E|)$ }

③ Običajna implementacija za redke grafe

za strukturo Q uporabimo kopico (gradnja $O(|V|)$)

razlika po tč ... $O(|V|)$

min. in odstranjevanje ... $O(\log |V|)$

popravki: $O(\log |V|)$ eno popravljanje, $O(|E|)$ popravljanj } $O(|E| \log |V|)$

Opomba. Z bolj zapletenimi strukturami lahko teoretično dosežemo $O(|V| \log |V| + |E|)$.