RAČUNALNIŠTVO 2 VAJE

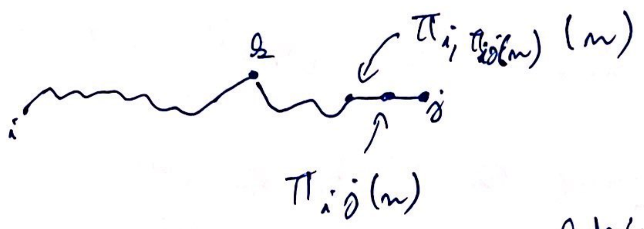
29.3.2023

7. VAJE

NALOGA 1:

F.W.: $D_{i,j}(k) = \min\{\underset{(*)}{D_{i,j}(k-1)},\ \underset{(**)}{D_{i,k}(k-1) + D_{k,j}(k-1)}\}$

$\Pi_{i,j}(k)$ ... (pred)zadnje vozlišče na $i-j$ poti (i do j), kjer smemo vmes uporabljati samo vozlišča $1, \dots, k$



G[i]

začetni je predhodnik

začetni je predhodnik

Začetni/robni pogoji: $\Pi_{i,i}(0) = i$, $\Pi_{i,G[i]}(0) = i$

(*): $\Pi_{i,j}(k) = \Pi_{i,j}(k-1)$

(**): $\Pi_{i,j}(k) = \Pi_{k,j}(k-1)$

~~NAPIŠI DO KONCA~~

REKONSTRUKCIJA POTI:

VHOD: vozlišča i, j, matrika $\Pi(n) = \Pi$

IZHOD: najkrajša pot od i do j

ALGORITEM:

```
p = j
pot = []
while p != i:
    pot.append(p)
    p = Π[i][p]
pot.append(i)
return pot.reverse()
```

ČASOVNA ZAHTEVNOST:

$O(n)$

NALOGA 2:

K rešitvi damo še uteži.

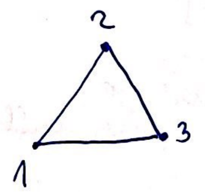
2 možni ideji:

1.) Shranjujemo povezave, ki kažejo v to vozlišče

2.) — " " — ven iz vozlišča

Odvisno se glede na problem, smiselno obravnavamo začetno in končno vozlišče.

NALOGA 3:



$1 \to 3$: 1 3

$\boxed{1\ 2\ 3}$ ← max

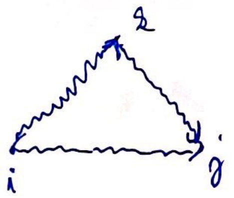
1 − 2 − 3

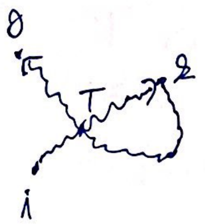
pot(1,2) + pot(2,3)

132 213

Ti dve poti nista "kompatibilni", to pomeni, če te poti združimo se ponovijo vozlišča in povezave.

Nazaj na F-W algoritem:





To se lahko zgodi v primeru, ko imamo negativne cikle.

NALOGA 4:

Sestavimo graf $G(V,E)$.

Vozlišča: EUR, YEN, USD (v splošnem: VALUTE)

Povezave: PRETVORBA

Zanima nas: "Najdaljša pot" v grafu od i do j

↑

vzdolž poti dobimo čim večji produkt uteži na povezavah

KONČNA VERZIJA:

Utež na povezavi i do j nastavimo na $-\log(R_{ij})$

⇓

iščemo najkrajšo pot od i - j v tem grafu

V ta namen želimo uporabiti F-W algoritem (to lahko naredimo, če graf nima negativnih ciklov)

Dokažimo, da jih naš graf res nima.

DOKAZ S PROTISLOVJEM:

Predpostavimo, da imamo negativen cikel od i - i

cena tega cikla: $-\left(\sum_{j=1}^{k} \log\left(R(i_{j-1}, i_j)\right)\right) < 0$

$$-\log\left(\prod_{j=1}^{k} R(i_{j-1}, i_j)\right) < 0 \quad \cdot(-1)$$

$$\log\left(\prod_{j=1}^{k} R(i_{j-1}, i_j)\right) > 0$$

$$\prod_{j=1}^{k} R(i_{j-1}, i_j) > 1$$

$\Rightarrow$ To je arbitraža $\rightarrow\leftarrow$

$\Rightarrow$ Graf nima negativnih ciklov

NALOGA 5:

Vhodni podatki:

- Usmerjeni graf $G = (V, E)$
- začetno vozlišče $s \in V$
- $c_{ij} \geq 0$: uteži so pozitivne

Izhodni podatki:

- Cena najcenejših poti od $s \rightarrow i \ \forall i \in V \rightsquigarrow D$
- Drevo najkrajših poti od $s \rightarrow i \ \forall i \in V \rightsquigarrow P$

$\downarrow$ implementacija

```
def dijkstra(G, s):
    n = len(G)
    D = [inf] * n
    P = [None] * n
    D[s] = 0
    P[s] = s
    obiskani = set()
    q = vrsta(V(G))
    while len(obiskani) != n:
        c = q.popmin()
        obiskani.add(c)
G/n     for sosed, utež in G[c]:
            if sosed not in obiskani:
                if D[c] + utež < D[sosed]:
                    D[sosed] = D[c] + utež
                    P[sosed] = c
    return D, P
```

$O(n)$

pregledamo na različno v q in vrnemo tistega z najmanjšim D[·].
ČZ: $O(n)$ ← v splošnem (za popmin())

$O(n)$

$O(n)$

DIJKSTRA: ČZ: $O(n^2)$

če bi namesto popmin() uporabili prioritetno vrsto ali kopico:

Vrsto bi definirali kot: q = pq((0, s))

razdalja, vozlišče

```
while q:
    c, d = q.popmin()   ← O(log n)
    if c in obiskani:
⑤       continue
```

```
if ...
   continue
obiđeni.add(c)
D[c] = d
for susjed, težina in G[c]:
    if not obiđeni <-> susjed:
        Q.push((d+težina, susjed))
```

do while

$O(|V|+|E|) \cdot O(\log n)$

Č7: $O(|E| \cdot \log |E|)$

dijkstra