Algoritmi in podatkovne strukture – 2

Grafi

najkrajša povezava med dvema vozliščema

Uteženi grafi

Graf je definiran kot:

- Imamo množico vozlišč, ki imajo svoje oznake: $V = \{v_1, v_2, \ldots, v_n\}$.
- Imamo množico povezav $E = \{(v_i, v_j; w_{ij}) \mid i, j = 1, 2, \ldots, n\}$, kjer povezava $(v_i, v_j; w_{ij})$ povezuje vozlišči v_i in v_j ter ima utež $w_{ij} \in \mathcal{R}$.
- Potem je utežen graf definiran kot $G = (V, E)$.

Utež predstavlja lahko $\gg$dolžino$\ll$, $\gg$težo$\ll$, $\gg$propustnost$\ll$, ...

Dopolnitev definicije:

- utež $w_{ij} \in \{0, \ldots, M-1\}$.
- utež $w_{ij} \in \mathcal{R}$ in utež $w_{ij} > 0$.
- graf je usmerjen ali ne.

Najkrajše poti

- Naj bo G povezan usmerjen utežen graf.
- *Pot* v grafu je zaporedje vozlišč $(v_0, v_1, \ldots, v_k)$, tako da je $(v_i, v_{i+i}; w_{i,i+i}) \in G.E$ za $i = 0..k-1$.
- **Pozor:** vozlišča se lahko ponavljajo, zato bi morali namesto *pot* govoriti *sprehod*.
- *Teža* poti $p = (v_0, v_1, \ldots, v_k)$ je

$$w(p) = \sum_{i=0}^{k-i} w_{i,i+i} \quad .$$

- Najkrajša pot od u do v je

$$\delta(u, v) = \begin{cases} \min_p(w(p) \mid p \text{ je pot od } u \text{ do } v)\} & \text{obstaja pot od } u \text{ do } v. \\ \infty & \text{sicer.} \end{cases}$$

- Različici problema: iščemo najkrajše poti (1) od danega vozlišča do vseh drugih vozlišč; (2) za vsak par različnih vozlišč.

Najkrajše poti

Izkoriščamo naslednjo lastnost.

Lema 1. *Naj bo v usmerjenem uteženem grafu* $G = (V, E)$, $p = (v_0, v_1, \ldots, v_k)$ *najkrajša pot od* v_0 *do* v_k *in naj bo* $p_{ij} = (v_i, v_{i+1}, \ldots, v_j)$ ≫*podpot*≪ *poti* p. *Potem je* p_{ij} *najkrajša pot od* v_i *do* v_j.

Ali lahko najkrajša pot med dvema vozliščema vsebuje cikel:

- negativen cikel: **težava!** – napihujemo
- pozitiven cikel: ne.
- cikel dolžine 0: lahko (a jih lahko odstranimo, ker ne vplivajo na dolžino).

Relaksacija – sproščanje

- Za vsako vozlišče v hranimo vrednost $d[v]$, ki je najkrajša do sedaj ocenjena razdalja od izvora s do v.
- vzamemo novo vozlišče u in če se $d[v]$ zaradi poti skozi u zmanjša, to popravimo.

```
Initialize:
  for all v d[v]= ∞;
Relax(u,v):
  if d[v] > d[u]+w(u,v) d[v]= d[u]+w(u,v)
```

Kako izbirati vozlišča u?

Dijsktra

```
Dijkstra(G, s) {
  S= {};
  for all v in G { d[v]= ∞; pq.insert(v); }
  pq.decreaseKey(s, 0);
  while !pq.empty() {
    u= pq.delMin();
    for all x in (u, x)
      if d[x] > d[x]+w(u,x)
        pq.decreaseKey(x, d[u]+w(u,v))
  }
}
```

Manjka knjigivodstvo za rekonstrukcijo poti. Dodajte!

Zahtevnost: $O(n \log n)$

Primerjaj tudi *Bellman-Fordovim* algoritmom.

Najkrajše poti med vsemi vozlišči

Poženemo zgornji algoritem n-krat: $O(n^2 \log n)$

- Definirajmo matriko:

$$w_{i,j} = \begin{cases} 0 & i = j \\ w(v_i, v_j) & \text{if} \exists (v_i, v_j) \\ \infty & \text{sicer} \end{cases}$$

- posamezne uteži so lahko negativne
- iščemo matriko $\delta_{i,j}$ – najcenejših poti iz v_i do v_j
- in matriko $\pi_{i,j}$ – predhodnikov na poti iz v_i do v_j
- lema o podpoti še vedno velja

Teorija

- ker lema velja, lahko pot $v_i \rightsquigarrow v_j$ razdelimo na $v_i \rightsquigarrow v_j \to v_j$, kjer je prva pot dolga m korakov in druga $m-1$ in nato še enega
- definirajnmo $d_{i,j}^{(k)}$ kot najkrajšo pot od v_i do v_j, ki je dolga največ k korakov in je

$$d_{i,j}^{(0)} = \begin{cases} 0 & \text{if} i = j \\ \infty & \text{sicer} \end{cases}$$

- potem velja:

$$d_{i,j}^{(m)} = \min(d_{i,j}^{(m-1)}, \min_{v_k \text{sosed} v_j}(d_{i,k}^{(m-1)} + w(v_k, v_j)))$$

  kar je v resnici relaksacijska enačba.
- ker je dolžina najkrajše poti največ n, $\delta_{i,j} = d_{i,j}^{(n-1)}$
- v vsakem koraku podaljšamo doseg najkrajše poti za 1

Algoritem – globalna relaksacija

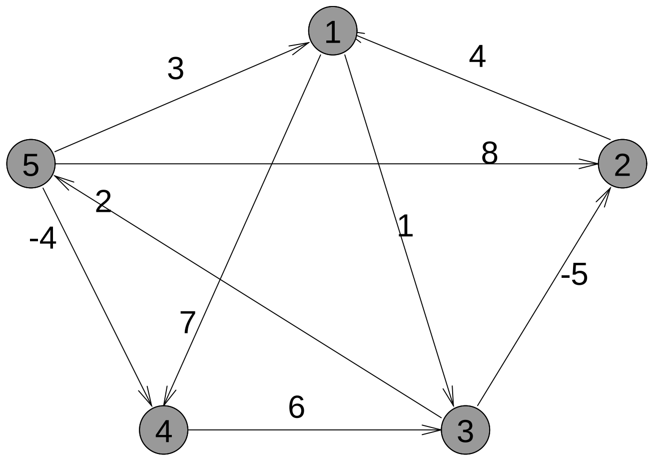
```
matrix ExtendSP(D, W) {
  new D’;            // D’ = d^(m), D = d^(m−1)
  for (i= 1; i <= n; i++)
    for (j= 1; j <= n; j++)
      D’[i,j]= D[i,j];
      for (k= 1; k <= n; k++)
        if (D’[i,j] > D[i,k] + W[k,j])
          D’[i,j]= D[i,k] + W[k,j]
}
```

Zahtevnost: $O(n^3)$.

NAMIG: Če zamenjamo min z $+$ in $+$ s $\times$ je to v resnici množenje matrik D in W.

Najkrajša pot – vsi pari

```
AllPairsShortestPath(G) {
  W= new Weights(G);
  D= new Distances(W);
  for (i= 1; i < n; i++)
    D= ExtendSP(D, W);
}
```



Zahtevnost: $O(n^4)$.

Splošnejši pogled

- naračunavamo: $D^{(0)}$, $D^{(1)} = W$, $D^{(2)} = W^2$, $D^{(3)} = W^3$, ..., $D^{(n-1)} = W^{n-1} = \delta$
- $D^{(k)}$, kjer $k \geq n$ ostaja δ
- se da to pospešiti?

Rešitev

- naračunavajmo: $D^{(1)} = W$, $D^{(2)} = w^2$, $D^{(4)} = D^{(2)} \cdot D^{(2)}$, ..., $D^{(2^k)} = D^{(2^{k-1})} \cdot D^{(2^{k-1})}$
- to počnemo, dokler $2^k \geq n$, torej $k = \lg n$

```
AllPairsShortestPath(G) {
  W= new Weights(G);
  D= new Distances(W);
  for (i= 2; i < 2*n; i*i)
    D= ExtendSP(D, D);
}
```

Zahtevnost: $O(n^3 \log n)$

Primerjaj *Floyd-Warshallow* algoritem.

Zahtevnost

	en vir	**več virov**
Dijkstra	$O(E \log V)$	$O(V E \log V)$
podaljševanje		$O(V^4)$
kvadriranje		$O(V^3 \log V)$

- Kako komentiramo zadnji stolpec?