

ISKANJE V ŠIRINO (BFS)

Definicija: Najkrajša pot med vozliščema v grafu je geodetska pot.

Velja: Ko enkrat določimo vozlišča na razdaljah $0, 1, \dots, d$, so vozlišča na razdalji $d+1$ natanko tista, ki jih še nismo obiskali, in so sosednja tistim na razdalji d .

Algoritem: procedure BFS(G, s):
for each $v \in G$:
 $\text{distance}[v] = \infty$
 $\text{previous}[v] = \text{null}$

 queue Q

$\text{distance}[s] = 0$
 $Q.\text{enqueue}()$

 while not $Q.\text{empty}()$:
 $u = Q.\text{dequeue}()$

 for each $v \in N(u)$:
 if $\text{distance}[v] == \infty$:
 $\text{distance}[v] = \text{distance}[u] + 1$
 $\text{previous}[v] = u$
 $Q.\text{enqueue}()$

Trditev: Za vsako razdaljo $d = 0, 1, 2, \dots$ obstaja trenutek, ko velja:

- 1) Vsa vozlišča na razdalji največ d imajo pravilno določene razdalje.
- 2) Vsa ostala vozlišča imajo razdaljo ∞ .
- 3) Vrsta vsebuje natanko vozlišča na razdalji d .

Dokaz: 2 indukcijo ...

Časovna zahtevnost: $O(|V| + |E|)$

DIJKSTROV ALGORITEM

Algoritem: procedure dijkstra(G, l, s):

for each $v \in G$:
 distance[v] = ∞
 previous[v] = null

distance[s] = 0

priority queue H ($V, \text{distance}$)

while not $H.empty()$:
 $u = H.minimal()$

for each $(u, v) \in E$:

if distance[v] > distance[u] + $l(u, v)$:

distance[v] = distance[u] + $l(u, v)$

previous[v] = u

$H.update(v, \text{distance}[v])$

Implementacije prioritete vrste:

- Seznam
- Dvojiška kopica
- Binomska kopica
- Fibonaccijeva kopica
- Parna kopica
- d-vejna kopica
- :

$$R := G \setminus H$$

Trditev: Obstaja naravno število d , da so vsa vozlišča v R na razdalji največ d od s , medtem ko so vsa vozlišča zunaj R na razdalji vsaj d .

Trditev: Za vsako vozlišče u iz G velja:

$\text{dist}(u)$ je nastavljena na ceno najcenejše poti od s do u , kjer vsa vmesna vozlišča so v R . Če take pot ne obstaja, je ta vrednost ∞ .

$$T(G) = |V| \cdot \text{insert}() + |V| \cdot \text{minimal}() + |E| \cdot \text{update}()$$

(n)

Prioritetna vrsta	insert	minimal	update	skupaj
neurejena tabela	$O(1)$	$O(V)$	$O(1)$	$O(V ^2)$
urejena tabela	$O(V)$	$O(1)$	$O(V)$	$O(V ^2 + E \cdot V)$
dvojiška kopica	$O(\log V)$	$O(\log V)$	$O(\log V)$	$O((V + E) \cdot \log V)$
d-vejna kopica	$O(\log_d V)$	$O(d \cdot \log_d V)$	$O(d \cdot \log_d V)$	$O(E \cdot \log_d V + V \cdot d \cdot \log_d V)$

Optimalno: $d \sim \frac{|E|}{|V|}$

BELLMAN-FORDOV ALGORITEM

Algoritem: Vhod: $G = (V, E)$ vsmerjen utežen graf, $s \in V$ začetna točka
Izhod: $\forall u \in C(s)$: $\text{dist}(u)$ cena najcenejše poti

procedure ShortestPaths(G, l, s):

for each $u \in V$:
 $\text{dist}(u) = \infty$
 $\text{prev}(u) = \text{null}$

$$\text{dist}(s) = 0$$

repeat $|V|-1$ times:
for each $e \in E$:
update(e)

procedure update($(u,v) \in E$):
if $\text{dist}(v) > \text{dist}(u) + l_{u,v}$:
 $\text{dist}(v) = \text{dist}(u) + l_{u,v}$
 $\text{prev}(v) = u$

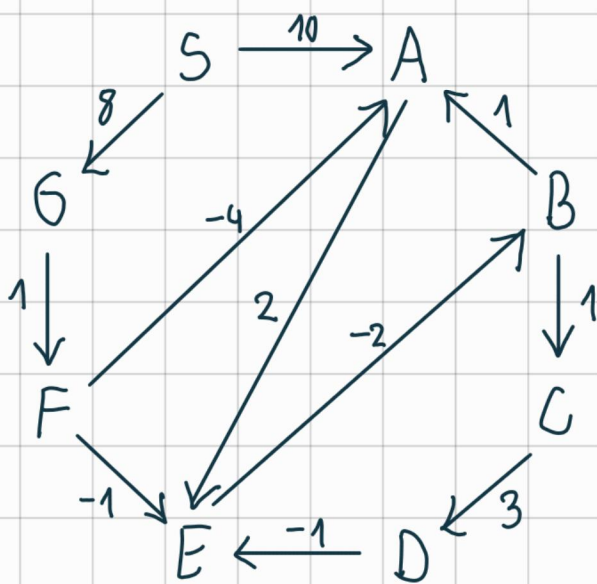
$$d_0(s, u) = \text{dist}(u)$$

$$\text{dist}(v) = \text{dist}(u) + l_{u,v}$$

$$\Rightarrow d_0(s, v) = \text{dist}(v)$$

Trditev: Po i -ti iteraciji repeat zanke je $\text{dist}(v)$ enaka najcenziji poti med vzemi potmi z največ i povezavami od s do v , če taka pot obstaja. Če ta pot ne obstaja, je ta vrednost ∞ .

Primer:



		iteracije								
		0	1	2	3	4	5	6	7	8
Vozlišča	S	0	0	0	0	0	0	0	0	0
	A	∞	10	10	5	5	5	5	5	5
	B	∞	∞	∞	10	6	5	5	5	5
	C	∞	∞	∞	∞	11	7	6	6	6
	D	∞	∞	∞	∞	∞	14	10	9	9
	E	∞	∞	12	8	7	7	7	7	7
	F	∞	∞	9	9	9	9	9	9	9
	G	∞	8	8	8	8	8	8	8	8

Ko enkrat ni sprememb v iteraciji, se ustavimo. Če ni negativnih ciklov, končamo v največ $|V|-1$ korakih.

Če imamo spremembo v $|V|$ -tem koraku, imamo negativen cikel.

Časovna zahtevnost: $O(|V| \cdot |E|)$

NAJCENEJŠE POTI V DAG

Algoritmi: Vhod: $G=(V,E)$ DAG, $s \in V$ začetna točka
Izhod: $\forall u: \text{dist}(u)$ najcenejša pot od s do u

procedure DagShortestPath(G, l, s):

for each $u \in V$:
 $\text{dist}(u) = \infty$
 $\text{prev}(u) = \text{null}$

$\text{dist}(s) = 0$

linearise(G) (topološko uredimo)

for each $u \in V$ in topological order:

for each $(u, v) \in E$:

update(u, v) (update od prej)

Časovna zahtevnost: $O(|V| + |E|)$

Trditev: Naj bo $V_i = \{v_1, \dots, v_i\}$. Po i -ti iteraciji zanke je $\text{dist}(v_j)$ enaka ceni najcenejše poti v grafu $G(V_i \cup \{v_j\})$, če taka pot obstaja, sicer pa ∞ .