Oblikovanje i analiza algoritama

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

23. studenoga, 2023.

Odabir aktivnosti

Problem odabira aktivnosti (*Activity-Selection Problem*):

- $S = \{a_1, a_2, \ldots, a_n\}$ - skup aktivnosti
- $s_i \geq 0$ - početak i-te aktivnosti
- $s_i < f_i < \infty$ - završetak i-te aktivnosti
- $[s_i, f_i >$ - trajanje aktivnosti a_i
- $[s_i, f_i > \cap [s_k, f_k >= \varnothing$ - aktivnosti a_i i a_k su kompatibilne
- **Zadatak:** pronaći **maksimalni** podskup međusobno kompatibilnih aktivnosti

Odabir aktivnosti

Primjer:

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Podskupovi međusobno kompatibilnih aktivnosti:

- $\{a_3,\ a_7,\ a_{11}\}$
- $\{a_3,\ a_9,\ a_{11}\}$
- $\{a_1,\ a_4,\ a_8,\ a_{11}\}$
- $\{a_2,\ a_4,\ a_8,\ a_{11}\}$
- $\{a_2,\ a_4,\ a_9,\ a_{11}\}$

Rješenje: sortirati aktivnosti uzlazno, prema vremenu završetka f_i.

Odabir aktivnosti

Primjer:

i	9	10	5	3	4	2	6	7	8	0	1
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16
		$3 < 4$		$5 \geq 4$		$5 < 7$		$8 \geq 7$			

Pronađeni podskup međusobno kompatibilnih aktivnosti:

- $\{a_9,\ a_3,\ a_7,\ a_1\}$

Pohlepni pristup će **uvijek** pronaći jedno od optimalnih rješenja.

- Krenuvši od proizvoljne aktivnosti k, pohlepni algoritam redom bira sve **moguće** (validne) aktivnosti s početkom $\geq f_k$. Stoga će postupak pronaći maksimalan podskup takvih aktivnosti.
- Pretpostavimo da postoji maksimalan podskup aktivnosti B koji počinje nekom aktivnosti $j > 1$. Tada skup aktivnosti $A = (B \setminus \{j\}) \cup \{1\}$ (zbog $f_1 \leq f_j$) sadrži jednak broj aktivnosti kao i B (stoga je maksimalan).

Minimalni broj perona

- $S = \{a_1,\ a_2,\ \ldots,\ a_n\}$ - skup autobusa
- $s_i \geq 0$ - dolazak i-tog autobusa na peron
- $s_i < f_i < 24$ - odlazak i-tog autobusa s perona
- $[s_i, f_i >$ - trajanje boravka i-tog autobusa na peronu
- $[s_i, f_i > \cap [s_k, f_k >= \varnothing$ - autobusi a_i i a_k mogu koristiti isti peron.

Zadatak: treba odrediti minimalan broj perona.
Rješenje: sortirati autobuse uzlazno, prema vremenu dolaska s_i.

Najkraći putevi iz jednog izvora

Neka je $G(V, E)$ težinski usmjeren graf, $l_e \geq 0$ za svaki brid $e \in E$, $s \in V$ istaknuti vrh (izvor, polazni vrh, *source*).

Problem SSSP(*Single Source Shortest Paths*):

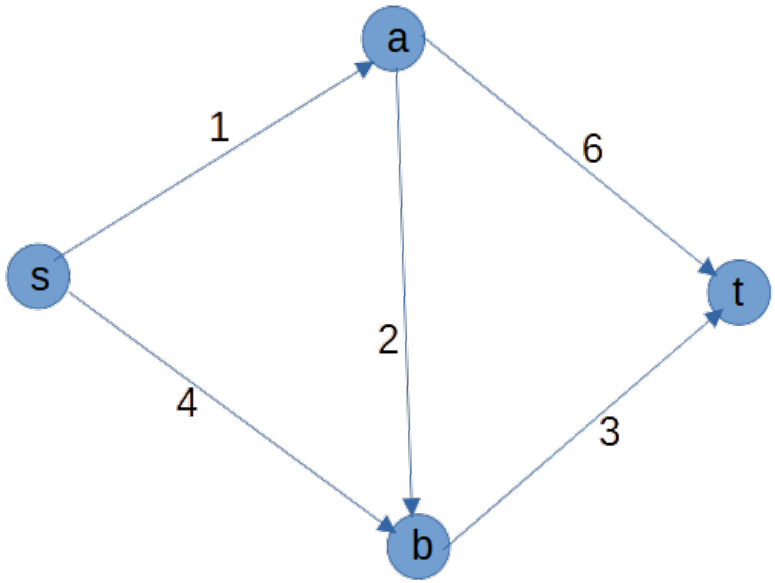
- Treba naći duljine najkraćih puteva (i same puteve) od izvora do svih preostalih vrhova grafa koji su dohvatljivi iz izvora.

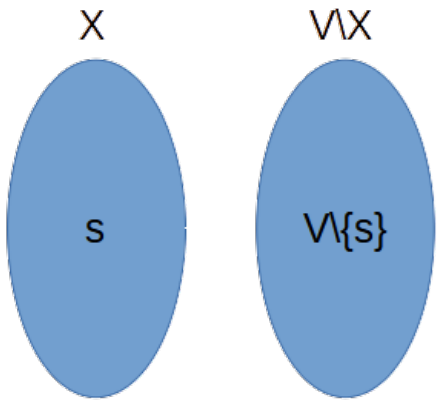
Algoritam vraća:

- $X \subset V$ - skup dohvatljivih vrhova
- $l(s, v)$ - duljinu najkraćeg puta od s do v, za svaki $v \in X$.

Dijkstrin algoritam

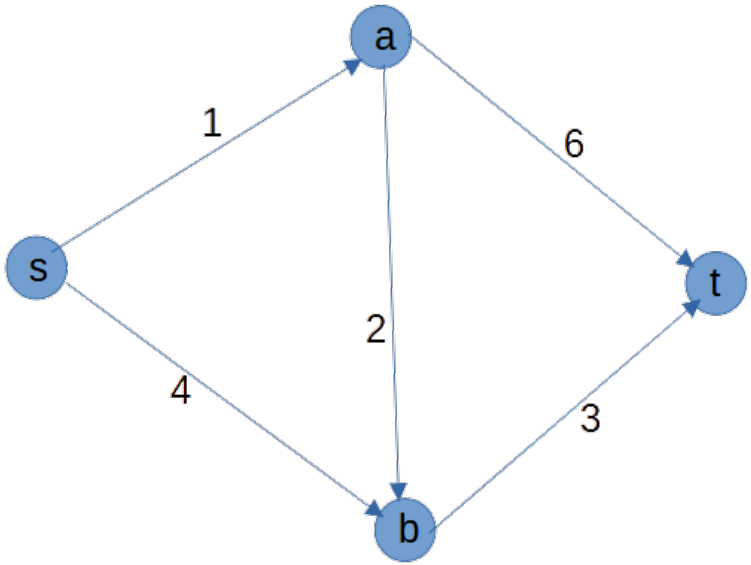
Algoritam za rješavanje problema SSSP.

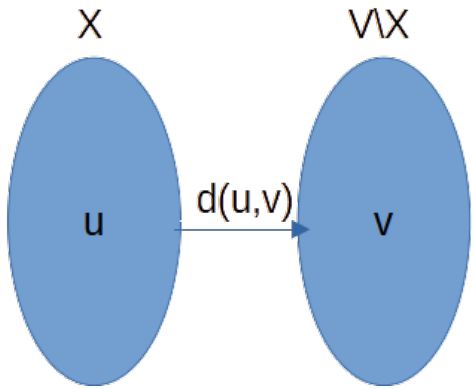




- Početak: $l(s,s) = 0$
- $\min \{l(s,u) + d(u,v)\},\ u \in X,\ v \in V \setminus X$
  - $\min\{1,4\} = 1,\ X = \{s,a\},\ l(s,a) = 1$
  - $\min\{0+4, 1+2, 1+6\} = 3,\ X = \{s,a,b\},\ l(s,b) = 3$
  - $\min\{1+6, 3+3\} = 6,\ X = \{s,a,b,t\} = V,\ l(s,t) = 6$

Dijkstrin algoritam





- $s \to a \to b \to t$

Dijkstrin algoritam - korektnost

Dijkstrin algoritam napreduje po koracima:

- inicijalizacija: $l(s,s) = 0,\ X = \{s\}$
- sve dok nismo našli put:
  - $\pi(v^*) \leftarrow min\{l(s,u) + d(u,v)\},\ u \in X,\ v \in V \setminus X$
  - $X \leftarrow X \cup \{v^*\},\ l(s,v^*) \leftarrow \pi(v^*)$

Definicija: Za neki put od s do $v \in V$ kažemo da je S-put, ako svi vrhovi na tom putu, osim eventualno zadnjeg vrha v pripadaju skupu S.

Dokaz korektnosti:

- Dokaz provodimo matematičkom indukcijom po koracima algoritma.
- Označimo s D polje koje sadrži najkraće putove do svih vrhova iz G. $D[u]$ predstavlja duljinu najkraćeg puta od s do u.
- Označimo s R skup svih vrhova $v \in V$ takvih da postoji put od s do v koji prolazi samo vrhovima koji se nalaze u skupu X.
- **baza indukcije**: inicijalizacija
  - X sadrži samo s, $D[s]$ možemo proizvoljno inicijalizirati (npr. 0).
  - R sadrži vrhove povezane bridom s vrhom s. Pošto od s do $v \in R$ postoji samo jedan X-put (preko brida (s,v)) on je i najkraći.

Dijkstrin algoritam - korektnost

- **pretpostavka indukcije**:
  - Pretpostavimo da na početku novog koraka (u petlji) vrijedi:
    a) $\forall v \in X,\ D[v]$ - duljina najkraćeg puta od s do v.
    b) $\forall v \in R \setminus X,\ D[v]$ - duljina najkraćeg X-puta od s do v.
- **korak indukcije**:
  - Ako je $R = X$:
    - Prema pretpostavci indukcije smo za sve vrhove iz X našli najkraće udaljenosti od s.
    - Ne postoji niti jedan brid koji spaja vrhove iz X s vrhovima iz $V \setminus X$ (nedohvatljivi su). Stoga algoritam korektno staje.
  - Ako je $R \neq X$ onda se izvršava sljedeći korak algoritma koji dodaje vrh v u skup X:
    a) Prošireni X je za vrh v veći od starog, stoga za sve vrhove $u \in X,\ u \neq v$ vrijedi tvrdnja iz prepostavke indukcije.
    - Pokažimo da tvrdnja vrijedi i za v. Prema pretpostavci indukcije $D[v]$ sadrži duljinu najkraćeg X-puta do v. Treba pokazati da je to ujedno i najkraći put od s do v, odnosno da najkraći put ne može prolaziti vrhom izvan X.

Dijkstrin algoritam - korektnost

- Slučaj $R \neq X$ nastavak:
  a) Nastavak:
    - Pretpostavimo suprotno, tj. da najkraći put prolazi nekim vrhom izvan X. Neka je $x \notin X$ prvi vrh na koji naiđemo tim putem od s do v.
    - Tada sigurno $x \in R$ jer postoji X-put od s do x, pa $D[x]$ sadrži najkraću X-udaljenost od s do x. Vrijedi:
    - najkraća udaljenost od s do v (preko x) $\geq$ udaljenost od s do x duž tog puta (bridovi nenegativne duljine) $\geq D[x]$ jer $D[x]$ sadrži najkraću duljinu X-puta, a po pretpostavci je x prvi vrh izvan X na putu $\geq D[v]$ (jer $v \in R \setminus X$, a v se bira tako da $D[v] = min_{w \in R \setminus X} D[w]$).
    - Ukoliko je put kroz x kraći imamo kontradikciju.
    - Ili je put kroz x jednak kao put s cijenom $D[v]$ (tada cijena puta od x do v mora biti 0).
    - Slijedi da je pronađeni X-put do v jedan od najkraćih puteva od s do v.

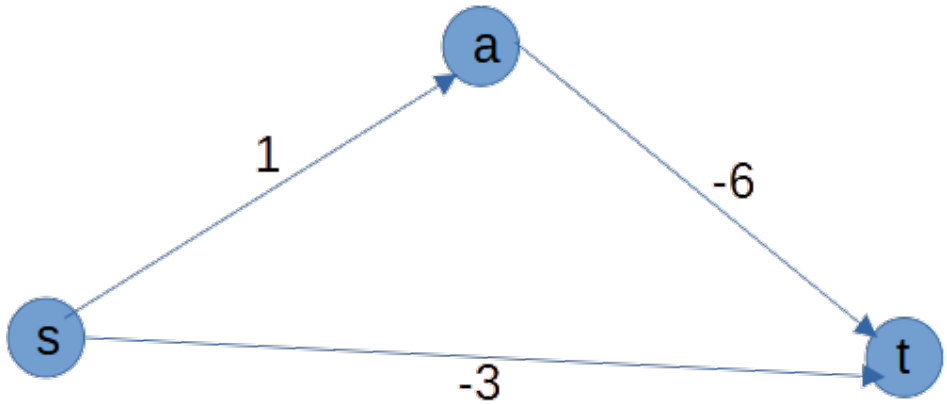
Dijkstrin algoritam - korektnost

- Slučaj $R \neq X$ nastavak:
  b) Promatrajmo vrh $w \in R \setminus X$. Kad vrh v dodamo skupu X imamo dvije mogućnosti za najkraći X-put od izvora do w:
    - nije se promijenio, pa $D[w]$ po pretpostavci već sadrži najkraću X-udaljenost do w.
    - skratio se i prolazi kroz v. Tada v mora biti zadnji vrh iz X na tom putu.
    - Pošto je v zadnji vrh iz X na tom putu, tada je duljina tog puta $D[v] + l(v, w) \leq D[w]$.
    - Gornja razmatranja vrijede za sve vrhove $w \in R \setminus X$.
- u skup $R \setminus X$ dodajemo sve vrhove koji ranije nisu bili dohvatljivi iz X.

Time je dokazano da algoritam vraća najkraći put od vrha s do svakog **dohvatljivog** vrha $v \in V$, odnosno algoritam radi korektno.

Dijkstrin algoritam - negativne težine

Dijsktrin algoritam **ne radi korektno** ukoliko graf sadrži bridove s negativnim težinama!



- $\min\{1, -3\} = -3,\ X = \{s, t\},\ l(s, t) = -3$
- $s \to a \to t,\ l(s, t) = -5$
- Ako težinama bridova dodamo pozitivnu konstantu koja sve težine pretvara u pozitivne, hoće li Dijkstrin algoritam dati korektno rješenje?

Dijkstrin algoritam - kod

```
1  #include <stdio.h>
2  #include <limits.h>
3  #include <stdlib.h>
4
5  int minDistance(int dist[], int observed[], int numNodes){
6    int min = INT_MAX, min_index;
7
8    for (int v = 0; v < numNodes; v++)
9       if (observed[v] == 0 && dist[v] <= min && dist[v]!=-1)
10         min = dist[v], min_index = v;
11   return min_index; }
12
13 void dijkstra(int numNodes, int graph[numNodes][numNodes],
       int src){
14     int *dist = (int*) malloc (numNodes*sizeof(int));
15     int *observed = (int*) malloc (numNodes*sizeof(int));
16
17     for (int i = 0; i < numNodes; i++)
18        dist[i] = -1, observed[i] = 0;
```

Dijkstrin algoritam - kod

```
19      dist[src] = 0;
20
21      for (int i = 0; i < numNodes - 1; i++) {
22          int u = minDistance(dist, observed,numNodes);
23          observed[u] = 1;
24
25          for (int v = 0; v < numNodes; v++)
26
27              if (!observed[v] && graph[u][v]
28                  && dist[u] != -1
29                  && (dist[u] + graph[u][v] < dist[v] || dist[
    v]==-1))
30                  dist[v] = dist[u] + graph[u][v];
31      }
32
33      printDistance(dist, numNodes);
34      free(dist); free(observed);
35  }
```

Dijkstrin algoritam - složenost

- $n = |V|$
- $m = |E|$
- implementacija pomoću polja - $\mathcal{O}(n^2)$
- implementacija pomoću hrpe - $\mathcal{O}(mlog_2 n)$
- implementacija pomoću Fibonnacijeve hrpe - $\mathcal{O}(m + nlog_2 n)$

Efikasna implementacija algoritma za svaki neistraženi $v \notin X$ umjesto:

- $\pi(v) = min_{e=(u,v):u \in X}(d[u] + l_e)$ računa
- $\pi[v] = min\{\pi[v], \pi[u] + l_e\}$

Dijkstrin algoritam - na raskrižju dinamičkog programiranja i pohlepnih pristupa

Veza s dinamičkim programiranjem:

- Uključuje Bellmanov princip optimalnosti kao ključnu komponentu algoritma.
- Može se shvatiti kao metoda dinamičkog programiranja zvana **uzastopna metoda aproksimacije**. Ta metoda iterativno poboljšava vrijednosti unutar matrice prilikom istraživanja prostora rješenja.

Veza s pohlepnim pristupima:

- Održava i iterativno poboljšava skup kandidata.
- Ispitani kandidati se naknadno više ne ispituju.
- Daje optimalno rješenje samo za **posebnu** klasu problema pronalaska najkraćeg puta u grafu (**nenegativne težine**).

Dijkstrin algoritam - na raskrižju dinamičkog programiranja i pohlepnih pristupa

- $A_j = \{v \in V,\ d(v_j, v) < \infty\},\ v_j \in V$ (čvorovi u koje možemo doći iz v_j)
- $B_j = \{v \in V,\ d(v, v_j) < \infty\},\ v_j \in V$ (čvorovi iz kojih možemo doći u v_j)
- $U = V \setminus X$

Funkcija koju optimizira Dijkstra u odnosu na postupak dinamičkog programiranja:

- (dijkstra), $D[i] = min\{D[i], D[j] + d(j, i)\},\ i \in A_j \cap U$
- (dinamičko), $D[i] = min\{D[i], D[j] + d(j, i)\},\ i \in A_j$