Oblikovanje i analiza algoritama

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

10. studenoga, 2023.

Pohlepni pristup - primjer 1

- **Coin Change Problem**: kupcu treba vratiti ostatak koristeći **minimalni broj** novčića. Pretpostavimo da su na raspolaganju novčići vrijednosti 1, 5, 10 ili 25 jedinica.
- **Karakteristični elementi**:
  - **kandidati:** konačni skup novčića čije su vrijednosti 1, 5, 10 ili 25 jedinica; skup sadrži **barem jedan** novčić svake vrste
  - **rješenje:** ukupna vrijednost izabranog skupa novčića je točno jednaka iznosu kojeg treba vratiti kupcu
  - **funkcija izbora:** izabire novčić najveće vrijednosti u skupu preostalih kandidata
  - **dopustiv skup:** ukupna vrijednost izbranog skupa novčića ne prelazi ($\leq$) iznos kojeg treba vratiti
  - **funkcija cilja:** broj novčića u rješenju.

Pohlepni pristup - zadatak 1

- Dokažite da, uz predloženi izbor vrijednosti novčića iz primjera, pohlepni algoritam uvijek nalazi optimalno rješenje, ako rješenje postoji.
- **Skica dokaza:** Neka je $n_1 \; n_5 \; n_{10} \; n_{25}$ rješenje dobiveno opisanim pohlepnim algoritmom, a $m_1 \; m_5 \; m_{10} \; m_{25}$ optimalno rješenje. n_i/m_i označavaju broj kovanica vrijednosti i u dobivenom rješenju.
- Krenuvši s desna (ili s lijeva) tražimo prvo mjesto, par (n_i, m_i) na kojemu se rješenja razlikuju.
- Promatramo redom: $i = 25, \; n_{25} > m_{25} \Rightarrow \ldots$

Pohlepni pristup - primjer 2

Primjer 2. a) Pokažite da ako fali novčić vrijednosti 1, da ne možemo vratiti proizvoljni iznos novca. b) Konstruirajte primjer skupa novčića i iznosa za koje pohlepni algoritam neće pronaći rješenje iako ono postoji. c) Konstruirajte primjer skupa novčića i iznosa za koje pohlepni algoritam ne daje optimalno rješenje.

a) ne možemo vratiti iznos od $x = 17,\ 18,\ 19,\ 21...$ novčića koristeći $C = \{5,\ 10,\ 25\}$.

b) pohlepni algoritam ne nalazi rješenje iako ono postoji za: $x = 12$, $C = \{2,\ 4,\ 9\}$. Rješenje: $4 + 4 + 4$, pohlepno: $9 + 2$.

c) pohlepni algoritam ne daje optimalno rješenje: $x = 12$, $C = \{1,\ 3,\ 4,\ 5\}$. Optimalno rješenje: $5 + 4 + 3$ ili $4 + 4 + 4$, pohlepno: $5 + 5 + 1 + 1$.

Napomena: pri računanju koristiti **cijelobrojno dijeljenje** umjesto oduzimanja.

Coin change problem

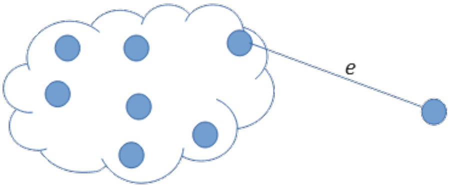
- Neka je k broj različitih vrijednosti. Tada je vremenska složenost algoritma koji rješava problem **Coin change problem** iz $\mathcal{O}(k)$ $(\Theta(k))$.
- Neka je:
  - n iznos kojega treba razmijeniti (vratiti kupcu)
  - c najveća vrijednost novčića koja je $\leq n$.
- Problem možemo riješiti i rekurzivno, na sljedeći način:
  - ako je $n = 0$ algoritam završava
  - u pritovnom rješavamo problem razmjene iznosa $n - c$.
- Razmislite kako riješiti problem koristeći dinamičko programiranje.

Minimalno razapinjuće stablo

- Neka je:
  - $G(V, E)$ povezan, težinski, neusmjereni graf
  - $l_e \geq 0$ za svaki brid $e \in E$
- Problem MST (Minimal Spanning Tree):
  - Treba pronaći podskup T bridova grafa G tako da svi vrhovi ostanu povezani samo bridovima iz T i da je zbroj duljina bridova iz T najmanji moguć.
- **Napomena:** Lako se vidi da je podgraf (V, T) grafa G stablo, tj. povezan graf bez ciklusa. Taj graf zovemo minimalno razapinjuće stablo grafa G.
- **Primjena:**
  - V - gradovi
  - težina brida $\{a, b\}$ - cijena izgradnje ceste između gradova a i b.
  - treba projektirati sistem cesta koji povezuje sve gradove uz najmanju moguću cijenu.

Minimalno razapinjuće stablo

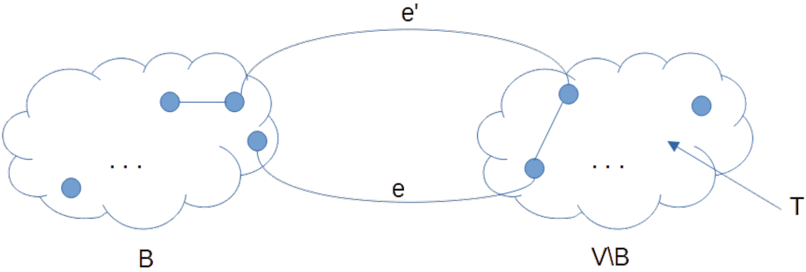
- **Pohlepni algoritmi za MST:**
  - Primov
  - Kruskalov
- Karakteristični elementi problema:
  - **kandidati:** bridovi (na početku $C = E$)
  - **rješenje:** skup bridova koji čini razapinjuće stablo
  - **funkcija izbora:** specificirat ćemo ju kasnije jer se ona razlikuje u dva gore spomenuta algoritma
  - **dopustiv skup:** skup bridova je dopustiv ako ne sadrži ciklus (ne zahtijevamo da bude stablo, tj. da bude povezan; može biti i šuma – svaka komponenta povezanosti je stablo)
  - **funkcija cilja:** suma duljina svih bridova u rješenju.
- Dodatna dva termina potrebna za dokaz korektnosti ovih algoritama:
  - Dopustiv skup bridova je obećavajući ako se može dopuniti do optimalnog rješenja. (Posebno, prazni skup je uvijek obećavajući jer je G povezan).
  - Brid e dira dani skup vrhova ako je točno jedan kraj brida u tom skupu.



- **Lema:** Neka je $G = (V, E)$ povezan, neusmjereni graf sa zadanim duljinama svih bridova. Neka je $B \subset V$ pravi podskup skupa vrhova grafa G. Neka je $T \subseteq E$ obećavajući skup bridova, takav da niti jedan brid iz T ne dira B. Neka je e najkraći brid koji dira B (ili bilo koji takav, ako ima više najkraćih). Tada je i skup $T \cup \{e\}$ obećavajući.

Dokaz leme: Pošto je po pretpostavci T obećavajući skup, po definiciji postoji skup U koji čini minimalno razapinjuće stablo grafa G, takvo da $T \subseteq U$. Imamo dva moguća slučaja:

- brid $e \in U$, tada je po definiciji $T \cup \{e\}$ obećavajući, što dokazuje tvrdnju leme.
- pretpostavimo $e \notin U$. Tada, pošto je U po pretpostavci minimalno razapinjuće stablo takvo da $T \subseteq U$, dodavanjem brida e u U dobivamo jedan ciklus, jer U je stablo (vidi sliku).



Minimalno razapinjuće stablo

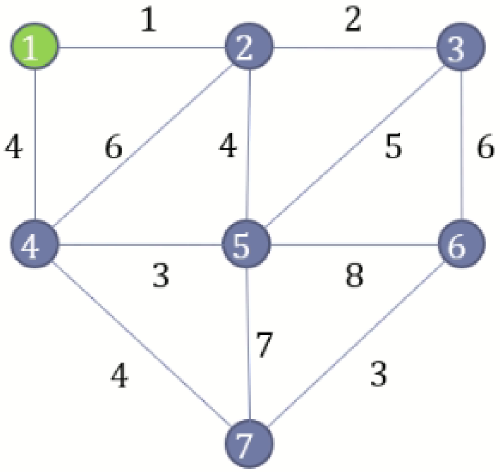
- Ako izbacimo brid e', ciklus nestaje i dobijemo stablo U' koje razapinje G. Pošto e ne prelazi duljinu e', duljina bridova u U' ne prelazi duljinu bridova u U. Stoga, U' je minimalno razapinjuće stablo grafa G i sadrži brid e. Pošto zbog uvjeta leme $e' \notin T$, mora biti $T \subseteq U'$. Dakle, $T \cup \{e\}$ je obećavajući skup bridova.

Primov algoritam

- Razvijen 1930 od strane češkog matematičara Vojtěch Jarník-a. 1957. je ponovo otkriven od strane američkog matematičara i računarskog znanstvenika Roberta C. Prima i 1959. od strane nizozemskog matematičara i računarskog znanstvenika Edsgera W. Dijsktre.
- **Primov algoritam:**
  - $B \leftarrow \{v\}$, v proizvoljan
  - $T \leftarrow \emptyset$
  - Dok $B \neq V$
    - pronađi najkraći mogući brid (u', v') takav da $u' \in V \setminus B$, a $v' \in B$.
    - $B \leftarrow B \cup \{u'\}$
    - $T \leftarrow T \cup \{(u', v')\}$
- **Teorem:** Primov algoritam radi korektno, tj. za povezan, neusmjereni graf G on vraća minimalno razapinjuće stablo T.

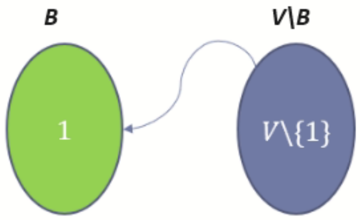
  Dokaz: direktno iz leme, indukcijom po broju vrhova u skupu B.

Primjer: Zadan je graf G kao na slici. Koristimo Primov algoritam za pronalazak minimalnog razapinjućeg stabla.



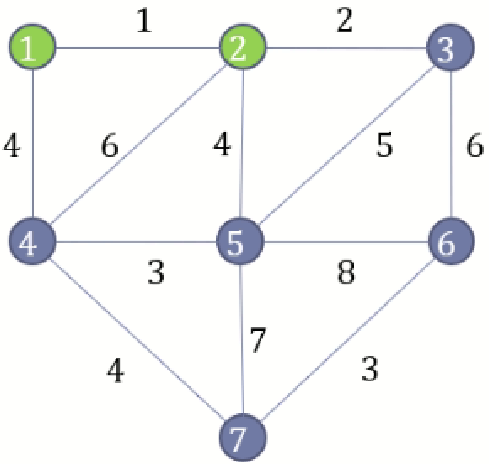
Primov algoritam

Inicijalizacija: $B = \{1\},\ \ T = \varnothing$



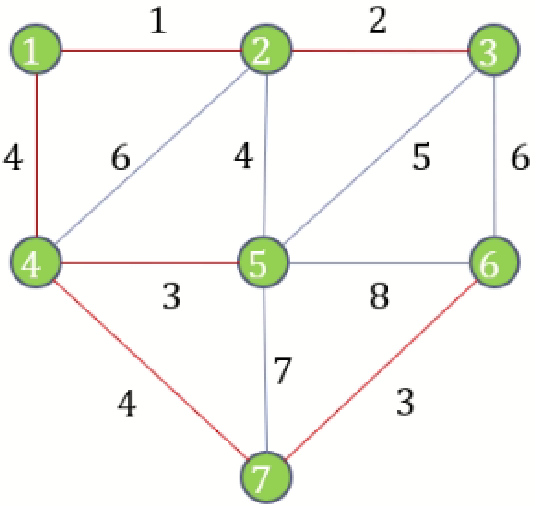
- mogući kandidati su $(1,2),\ (1,4) \Rightarrow B = \{1,2\},\ \ T = \{(1,2)\}$

Primov algoritam



- mogući kandidati su
  $(1,4),\ (2,3),\ (2,4),\ (2,5) \Rightarrow B = \{1,2,3\},\ \ T = \{(1,2),\ (2,3)\}$
- . . .

Primov algoritam



- $B = V, \ \ T = \{(1,2),\ (2,3),\ (1,4),\ (4,5),\ (4,7),\ (6,7)\}$
- Ukupna duljina je $1 + 2 + 4 + 3 + 4 + 3 = 17$.

Primov algoritam

- **Napomena:** U ovoj formi, izabrani brid uvijek prihvaćamo (nema odbacivanja). Kada bi funkcija izbora samo vratila brid najmanje duljine (među preostalim bridovima), imali bi mogućnost odbacivanja.
- Funkcija prihvaća samo one bridove (u, v) za koje je $u \in V \setminus B$ i $v \in B$ (prema Lemi) tako da dobijemo minimalno razapinjuće stablo za $B \cup \{u\}$.
- Graf može imati više minimalnih razapinjućih stabala. U algoritmu se ta mogućnost uočava kada imamo nekoliko bridova iste najmanje duljine $l((u, v))$ koji zadovoljavaju i uvjet $u \in V \setminus B$ i $v \in B$.
- **Složenost:**
    - $\mathcal{O}(|V|^2)$ - koristeći pretraživanje i matricu susjedstva
    - $\mathcal{O}(|E| log_2(|V|))$ - binarna hrpa i matrica susjedstva
    - $\mathcal{O}(|E| + |V| log_2(V))$ - fibonacijeva hrpa i matrica susjedstva

Primov algoritam - pseudokod

```
int prim(int n, int ** L, edge * T, int *B, int * nearest,
    int* mindist){

    int i,j,k;

    B[0] = 1; //izaberemo proizvoljni pocetni vrh
    T = empty_set();

    for(int i=2;i<=n;i++){
        nearest[i]  = 1;
        mindist[i] = L[i][1];
    }

    int min;

    for(int i=1;i<=n-1;i++){
        min = INFINITY;
        for(int j=2;j<=n;j++){
            if(mindist[j]>=0 && mindist[j]<min){
                min = mindist[j];
```

Primov algoritam - pseudokod

```
                k = j;
                }
        }
        T = set_union(T,edge(k,nearest[k]));
        mindist[k] = -1;

        for(j=2;j<=n;j++)
            if(L[k][j]<mindist[j]){
                mindist[j] = L[k][j];
                nearest[j] = k;
                }
    }
}
```

Zadatak: pretpostavka algoritma je da je G povezan graf. Popravite ga da prepoznaje nepovezan graf G. Što bi algoritam mogao vratiti u tom slučaju?