Primov algoritam

Stefani Duvnjak

27. studenoga 2023.

Problem - Minimalno razapinjuće stablo

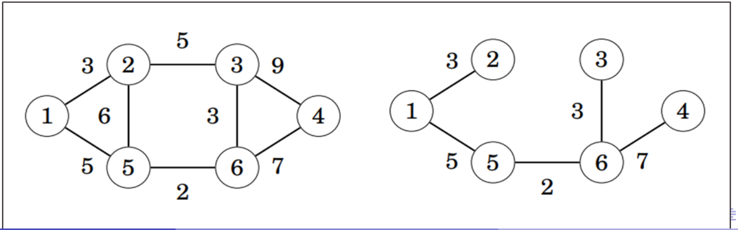
Definicija

Neka je $G = (V, E)$ povezan, neusmjeren graf, gdje V predstavlja skup čvorova(vrhova), a E skup bridova(grana).
Svaki brid $e \in E, e = (u, v)$, ima nenegativnu težinu(duljinu) $w(u, v)$.

- CILJ: pronaći podskup $T \subseteq E$ koji povezuje sve vrhove skupa V samo bridovima iz T uz najmanju težinu $w(T)$

$$\rightarrow w(T) = \sum w(u, v)$$

minimalna.
- Podgraf (V, T) grafa G je **minimalno razapinjuće stablo** grafa G.



Pohlepni algoritmi i minimalno razapinjuće stablo

- rješenje problema koristeći pohlepne algoritme
  - Kruskalov algoritam
  - Primov algoritam
- pojmovi
  - skup C - svi raspoloživi kandidati $\rightarrow C = E$
  - skup S - rješenje $\rightarrow$ skup bridova koji povezuju sve vrhove
  - S dopustiv ako ne sadrži ciklus
  - funkcija cilja - zbroj bridova koji minimiziramo
  - funkcija izbora

```
S = prazan_skup
while ( S ne tvori razapinjuće stablo za G )
    nađi brid e koji je dopustiv
    dodaj e u S
return S
```

Pohlepni algoritmi i minimalno razapinjuće stablo

Definicija

Skup bridova je **obećavajući** *ako se može dopuniti do optimalnog rješenja.*
Ako je G povezan, prazan skup je obećavajući.
Brid e **dira** *skup vrhova W ako je točno jedan kraj brida u tom skupu.*

Lema

Neka je $G = (V, E)$ povezan, neusmjeren graf sa zadanim težinama bridova. Neka je $W \subset V$ pravi podskup skupa vrhova V. Neka je $T \subseteq E$ obećavajući skup bridova, takav da niti jedan brid iz T ne dira W. Neka je e najkraći brid koji dira W (ili bilo koji takav, ako ih je više). Tada je i skup $T \cup \{e\}$ obećavajući.

Primov algoritam

- funkcija izbora - u svakom koraku za trenutni graf (W, T), bira najkraći brid $e = (u, v)$ takav da je $u \in V \setminus W$ i $v \in W$.

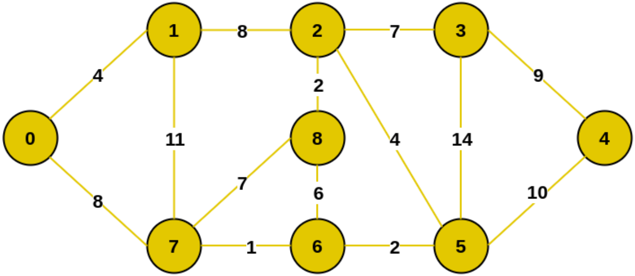
Teorem

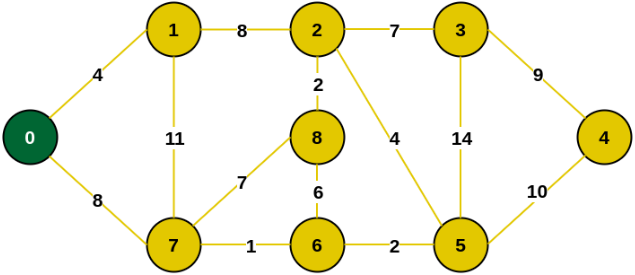
Primov algoritam radi ispravno, tj. za povezan neusmjeren graf G, vraća minimalno razapinjuće stablo T.

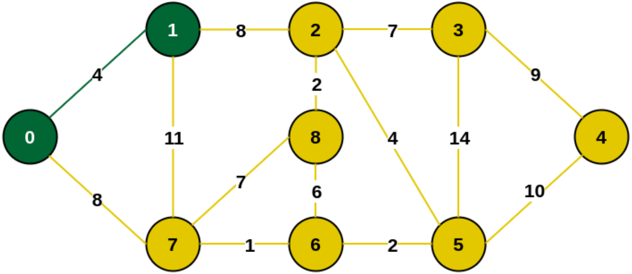
Dokaz:
Slijedi direktno iz Leme, indukcijom po broju vrhova u skupu W.

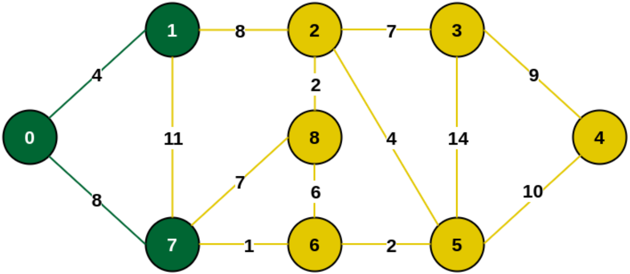
Pseudokod za Primov algoritam

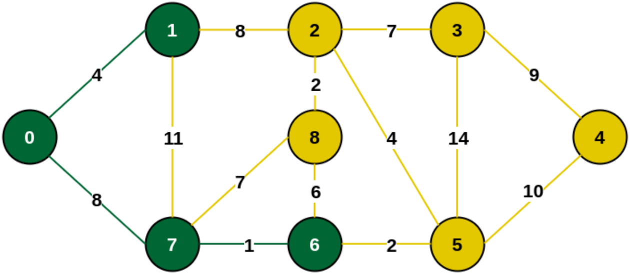
```
procedure Prim(
        G = (V, E) : graf;
        w : E → ℝ₀⁺ : funkcija za težinu;
        var T : skup bridova );
begin
      { Inicijalizacija }
    T ← ∅; { sadržavat će bridove minimalno razapinjućeg stabla }
    W ← bilo koji vrh iz V;
    while W ≠ V do
      begin
        nađi (u, v) najmanje duljine w(u, v), takav da je u ∈ V \ W i
v ∈ W;
        T ← T ∪ {(u, v)};
        W ← W ∪ {u};
      end;
end; {Prim}
```

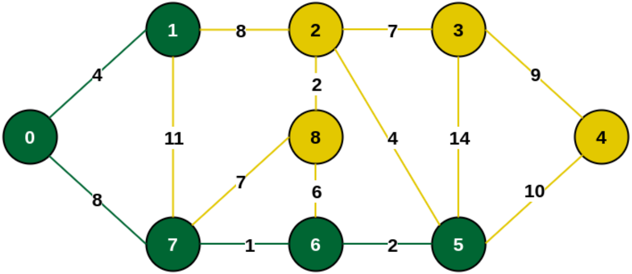

1
8
2
7
3
4
9
2
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

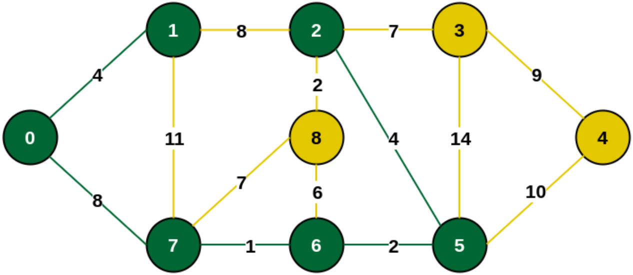

1
8
2
7
3
4
9
2
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

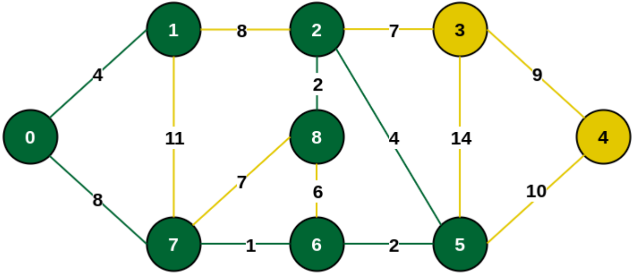

1
8
2
7
3
4
2
9
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

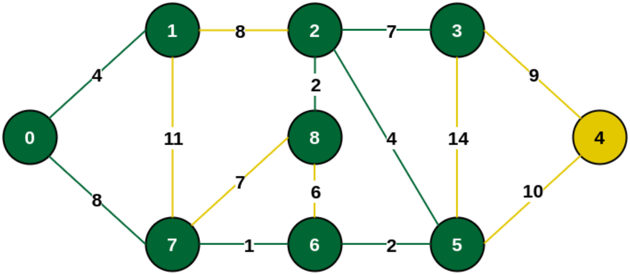

1
8
2
7
3
4
2
9
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

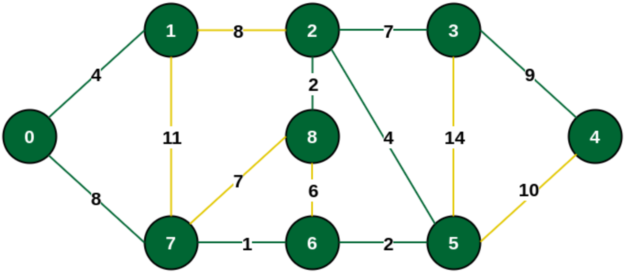

1
8
2
7
3
4
2
9
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

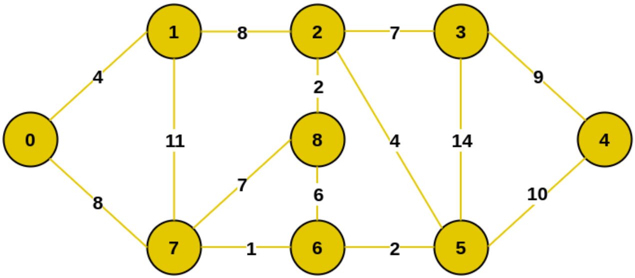

1
8
2
7
3
4
2
9
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5

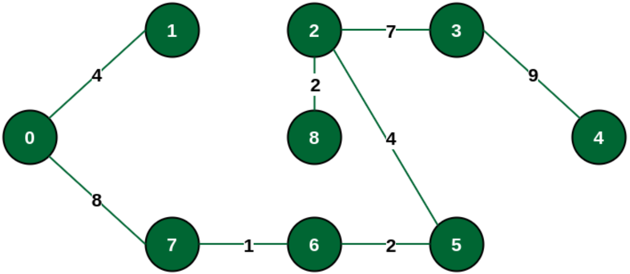

0
1
2
3
4
5
6
7
8
4
8
8
7
11
2
4
14
9
7
6
10
1
2


0
1
2
3
4
5
6
7
8
4
8
8
7
11
2
4
14
9
7
6
10
1
2


0
1
2
3
4
5
6
7
8
4
8
11
8
7
2
4
9
14
7
6
10
1
2


0
1
2
3
4
5
6
7
8
4
8
11
8
7
2
4
9
14
7
6
10
1
2


1
8
2
7
3
4
2
9
0
11
8
4
14
4
7
6
10
8
7
1
6
2
5


1
2
7
3
4
2
9
0
8
4
4
8
7
1
6
2
5

Implementacija

- koristeći pretraživanja i matricu susjedstva
- koristeći hrpu i matricu susjedstva
- koristeći fibonacijevu hrpu i matricu susjedstva

Implementacija koristeći pretraživanja i matricu susjedstva

$$L(u,v) = \begin{cases} w(u,v) & \text{,ako je } (u,v) \in E \\ \infty & \text{,inače} \end{cases}$$

begin

{ Inicijalizacija $W := [1]$, $T = [\,]$. }
$T := \emptyset$; { T akumulira bridove MST. }
for $j := 2$ **to** n **do** { samo vrh 1 je u skupu W. }
begin
$mindist[j] := L[j,1]$;
$nearest[j] := 1$;
end ; { for j }

Implementacija koristeći pretraživanja i matricu susjedstva

```
    { Pohlepna petlja. }
for i := 1 to n − 1 do   { T sadrži n − 1 bridova. }
    begin
    min := ∞ ;
    for j := 2 to n do
        if (mindist[j] ≥ 0) and (mindist[j] < min) then
            begin
            min := mindist[j] ;
            k := j ;
            end ;  { if, for j }

        { k je definiran ako i samo ako je G povezan, inače ne! }
    T := T ∪ { {k, nearest[k]} } ;
    mindist[k] := −1 ;  { dodaj vrh k skupu W. }

        { Popravi polja nearest, mindist za novi k u skupu W. }
    if i < n − 1 then
        for j := 2 to n do
            if L[k, j] < mindist[j] then   { j ∉ W }
                begin
                mindist[j] := L[k, j] ;
                nearest[j] := k ;
                end ;  { if, for j, if }

    end ;  { for i — pohlepna petlja }
```

Vremenska složenost

- Pretpostavka da imamo graf G s n vrhova
- Vanjska(pohlepna) petlja se ponavlja $n-1$ puta $\rightarrow T(n) \in O(n)$
- U toj petlji imamo jos dvije for petlje koje se ponavljaju $n-2$ puta $\rightarrow T(n) = 2(n-2) \rightarrow T(n) \in O(n)$

$$T(n) \in O(n^2)$$

- Ovisi o broju bridova, vrijedi

$$n-1 \leq |E| \leq \frac{n(n-1)}{2}$$

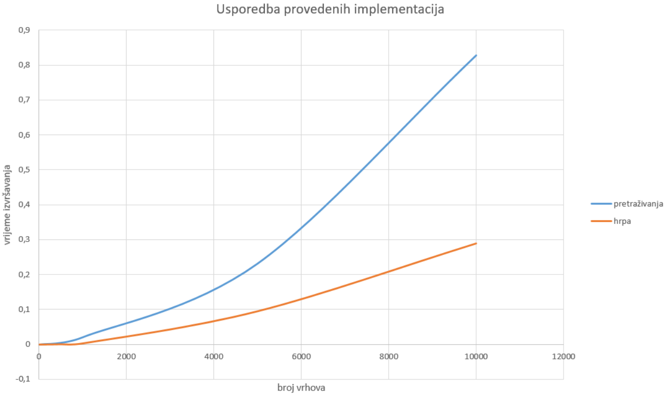
Vremenska složenost

- Pretpotavka da imamo graf G s n vrhova
- U minimalnu hrpu spremamo sve vrhove koji diraju W s najmanjom težinom brida
- while petlja ponavlja se n puta i za svaki prolasku uzimamo korijen hrpe koju "prepravljamo" (EXTRACT-MIN) sa složenosti $O(log_2 n) \to T(n) \in O(n \cdot log_2 n)$
- u svakom prolasku while petlje "prepravljamo" težine bridova u hrpi (DECREASE-KEY) $\to T(n) \in O(\mid E \mid \cdot log_2 n)$
- $\mid E \mid \geq n \to T(n) \in O(\mid E \mid \cdot log_2 n)$

```
PRIM(G, w, r)
Q = ∅
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
    INSERT(Q, u)
DECREASE-KEY(Q, r, 0)        // r.key = 0
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            DECREASE-KEY(Q, v, w(u, v))
```

Empirijska analiza

- Programski kod napisan u C-u
- Nasumični broj bridova u grafu za 10, 100, 500, 1000, 5000 i 10000 vrhova
- Svaki test izvršen 5 puta
- Vrijeme u sekundama



Literatura

- Cormen, T.H., Leiserson C. E., Rivest, R. L., Stein, C. Introduction to algorithms 4th edition. Str. 585.-591, 594.-598.
- Saša Singer, Složenost algoritama. Str. 31. - 43.

Hvala na pažnji