Tokovi najmanjeg troška i tokovi maksimalne vrijednosti

Barbiš, Apolinar

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:587549

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**



Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb







Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Apolinar Barbiš

TOKOVI NAJMANJEG TROŠKA I TOKOVI MAKSIMALNE VRIJEDNOSTI

Diplomski rad

Zagreb, srpanj, 2018.

Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Apolinar Barbiš

TOKOVI NAJMANJEG TROŠKA I TOKOVI MAKSIMALNE VRIJEDNOSTI

Diplomski rad

Voditelj rada:
doc. dr. sc. Lavoslav Čaklović

Zagreb, srpanj, 2018.

Ovaj diplomski rad obranjen je dana _____ pred nastavničkim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____ .

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

1 Uvod **2**

2 Definicije **3**

3 Put najmanjeg troška (najkraći put) **5**
3.1 Najkraći putevi iz jednog izvora (vrha) 7
3.1.1 Dijkstrin algoritam 8
3.1.2 Moore - Bellman - Fordov (MBF) algoritam 12
3.2 Najkraći putevi između svaka dva vrha na grafu 22
3.2.1 Floyd - Warshall (FW) algoritam 23

4 Mrežni tokovi **28**
4.1 Tok maksimalne vrijednosti 29
4.1.1 Ford - Fulkerson (FF) algoritam 31
4.1.2 Edmonds - Karp (EK) algoritam 38
4.1.3 Dinic (Dinitz) algoritam 41
4.1.4 Goldberg - Tarjan (GT) algoritam 46
4.2 Tokovi s više ponora i izvora 53

5 Literatura **54**

6 Sažetak **55**

7 Summary **55**

8 Životopis **56**

1 Uvod

U ovome radu obradit ćemo dva veoma poznata problema područja diskretne matematike.

U poglavlju 3 bavit ćemo se traženjem puta najmanjeg troška (najkraćeg puta) između dva vrha na (usmjerenom ili neusmjerenom) grafu. Obradit ćemo 3 algoritma za rješavanje ovog problema: Dijkstrin, Moore - Bellman - Ford te Floyd - Warshall algoritam.

U poglavlju 4 bavit ćemo se traženjem toka maksimalne vrijednosti unutar zadane mreže. Obradit ćemo 4 algoritma za rješavanje ovog problema: Ford - Fulkerson, Edmonds - Karp, Dinic te Goldberg - Tarjan algoritam.

Navedeni problemi zanimljivi su za proučavanje iz razloga što ih primijenjujemo na unaprijed zadane grafove stoga ih je veoma jednostavno vizualizirati. Dodatna motivacija za proučavanje ovih problema jest ta što su veoma primijenjivi u stvarnome životu. Jednu od primjena problema najmanjeg troška prikazat ćemo u Primjeru 1., a za problem toka maksimalne vrijednosti dat ćemo dva primjera, Primjer 3. i Primjer 4.

Prije no što krenemo u samu razradu teme, u poglavlju 2 dajemo sve definicije potrebne za razumijevanje rada.

2 Definicije

Neusmjereni graf G je uređena trojka $G := (V, E, \varphi)$, gdje je $\emptyset \neq V = V(G)$ skup **vrhova**, $E = E(G)$ skup **bridova** disjunktan sa $V(G)$, a φ funkcija incidencije koja svakom bridu $e \in E(G)$ pridružuje skup $\{u, v\}$, gdje su $u, v \in V(G)$, ne nužno različiti. **Usmjereni graf G** je uređena trojka $G := (V, E, \varphi)$, gdje funkcija incidecije φ svakom bridu $e \in E$ pridružuje uređen par (u, v) vrhova iz $V(G)$. Tada kažemo da su u i v krajevi od e i pišemo $e = \{u, v\}$ ($e = (u, v)$). Kažemo da su vrhovi u i v **susjedni** ako postoji brid t.d. su mu u i v krajevi. Dva brida ili više njih s istim (uređenim) parom krajeva zovu se **višestruki** ili **paralelni** bridovi. Brid čiji su krajevi isti zove se **petlja**. Kažemo da je graf G je **prazan** ako je $E(G) = \emptyset$. Graf G je **jednostavan** ako nema petlji ni višestrukih bridova. Odsada nadalje, radi jednostavnosti, kroz cijeli rad koristiti ćemo oznaku $e = (u, v)$ neovisno o tome je li graf usmjeren ili neusmjeren, imajući pritom na umu navedene definicije. Kažemo da je ***H* podgraf** grafa G i pišemo $H \subseteq G$ ako je $V(H) \subseteq V(G)$ i $E(H) \subseteq E(G)$, a $\varphi_H = \varphi_G \mid_{E(H)}$ (tj. φ_H je restrikcija od φ_G na $E(H)$). Kažemo još da ***G* sadrži *H***. Ponekad, radi jednostavnosti, pišemo $H = (V(H), E(H))$, podrazumijevajući pritom navedenu prirodnu restrikciju funkcije φ. Neka je $V' \subseteq V$. Graf $\mathbf{G - V'}$ definiramo sa $G - V' := (V \setminus V', \{e = (u, w) \in E(G) \mid u, v \in V \setminus V'\})$. Ako je $V' = \{v\}$, za neki $v \in V(G)$, umjesto $G - \{v\}$ pišemo i $G - v$. Slično, za $E' \subseteq E$, graf $\mathbf{G - E'}$ definiramo sa $G - E' := (V(G), \{e = (u, w) \in E'\})$. Ako je $E' = \{e\}$, za neki $e \in E(G)$, umjesto $G - \{e\}$ pišemo i $G - e$. Neka je $H \subseteq G$ i neka je $v \in V(G) \backslash V(H)$. Graf $\mathbf{H + v}$ definiramo sa $H + v := (V(H) \cup \{v\}, \{e = (u, v) \in E(G) \mid u, v \in V(H) \cup \{v\})$. Neka je $H \subseteq G$ i neka je $v \in V(H)$. Graf $\mathbf{H - v}$ definiramo sa $H - v := (V(H) \backslash \{v\}, \{e = (u, v) \in E(G) \mid u, v \in V(H) \backslash \{v\})$.

Težinski graf je uređeni par (G, u), gdje je $G = (V, E)$ graf, a $u : E(G) \to \mathbb{R}_+$ funkcija koju zovemo **težinskom funkcijom** grafa G. Ukoliko na grafu G nije zadana težinska funkcija, uzimamo $u(e) = 1$, za sve $e \in E(G)$. Za $H \subseteq G$ definiramo $c(E(H)) := \sum_{e \in E(H)} c(v)$. **Šetnja** W u grafu G je niz $W = v_0 e_1 v_1 e_2 \dots e_k v_k$ gdje su $v_i \in V(G)$, za $0 \leq i \leq k$ te $e_i = (v_{i-1}, v_i) \in E(G)$, za $1 \leq i \leq k$. Vrijedi $W \subseteq G$. Šetnju W zovemo (v_0, v_k) - šetnja. Kažemo da vrh v_{i-1} **prethodi** vrhu v_i u W, za $1 \leq i \leq k$. Vrhove $v \in V(W) \backslash \{v_0, v_k\}$ nazivamo **unutarnjim** vrhovima šetnje W, a vrhove v_0 i v_k **vanjskim** vrhovima. **Duljina (težina)** šetnje W definirana je sa $\sum_{e \in E(W)} u(e)$. Šetnja W je **zatvorena** ako je $v_o = v_k$. Za $v_i, v_j \in V(W)$ definiramo $W_{[v_i, v_j]} \subseteq W$ sa $W_{[v_i, v_j]} := v_i e_{i+1} v_{i+1} \dots e_j v_j$. Ako su svi bridovi šetnje W međusobno različiti, onda W zovemo **staza**, a ako su na stazi i svi vrhovi međusobno različiti, zovemo ju **put**. Zatvoreni put zove se **ciklus**. Kažemo da je graf G **cikličan** ako postoji ciklus C t.d. $C \subseteq G$. U suprotnom kažemo da je **acikličan**. **Hamiltonov put** na grafu G je razapinjući put grafa G, tj. put koji sadrži sve vrhove grafa, a **Hamiltonov ciklus** je razapinjući ciklus grafa. Kažemo da je graf **Hamiltonov** ako sadrži Hamiltonov ciklus.

Kažemo da je graf G **planaran** ako se može nacrtati u ravnini tako da mu se bridovi sijeku samo u vrhovima. Kažemo da su vrhovi $u, v \in V(G)$ **povezani** ako postoji (u, v) ili (v, u)

- put. Graf je **povezan** ako su svaka dva njegova vrha povezana nekim putom. Kažemo da je vrh v **dostižan** iz vrha u ako postoji (u, v) - put. **Udaljenost** vrha v od vrha u u grafu G je težina najkraćeg (u, v) - puta u G i označavamo ju sa $dist_G(u, v)$. Ponekad umjesto $dist_G(u, v)$ pišemo samo $dist(u, v)$. Po definiciji uzimamo $dist_G(v, v) = 0, \forall v \in V(G)$. Ako vrhovi $u, v \in V(G)$ nisu povezani u grafu G onda kažemo da je udaljenost između njih beskonačna, i pišemo $dist_G(u, v) = dist_G(v, u) = \infty$. **Udaljenost vrha u od skupa V_1**, $u \in V(G), V_1 \subseteq V(G)$, definiramo sa:

$$dist_G(u, V_1) = \min_{v \in V_1} dist_G(u, v).$$

Neka su $X, Y \subseteq V(G)$. Skup $\boldsymbol{E(X, Y)}$ definiramo sa $E(X, Y) := \{(x, y) \in E(G) \mid x \in X \backslash Y, y \in Y \backslash X\}$. **Skup susjeda skupa X** definiramo sa $\Gamma(X) := \{v \in V(G) \backslash X \mid E(X, \{v\}) \neq \emptyset\}$. Skupove $\delta^+(X)$ i $\delta^-(X)$ definiramo sa $\delta^+(X) := E(X, V(G) \backslash X)$ te $\delta^-(X) := \delta^+(V(G) \backslash X)$. Ako je $X = \{v\}$, za neki $v \in V(G)$, umjesto $\delta^+(\{v\})$ odn. $\delta^-(\{v\})$ pišemo i $\delta^+(v)$ odn. $\delta^-(v)$.

Neka je A algoritam s ulaznim skupom podataka X i neka je $f : \mathbb{N} \to \mathbb{R}_+$ funkcija. Kažemo da je **složenost** algoritma A $\boldsymbol{O(f)}$ ako postoji $\alpha > 0$ t.d. A završava nakon najviše $\alpha \cdot f(x)$ izvršenih operacija, za svaki $x \in X$. Kažemo još da je A **rješiv** u $O(f)$ vremenu.

3 Put najmanjeg troška (najkraći put)

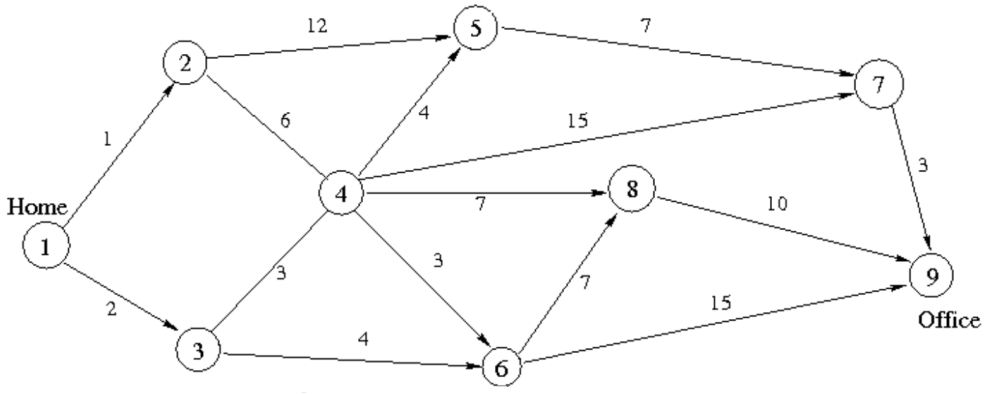
Jedan od najpoznatijih problema u diskretnoj matematici jest kako pronaći put najmanjeg troška između dva vrha na usmjerenome grafu. Problem je sljedeći:

· neka je G usmjereni graf, $c : E(G) \to \mathbb{R}$ težinska funkcija te neka su $s, t \in V(G)$,

· treba pronaći $(s-t)$ - put P_{st}, koji će imati minimalnu težinu $c(E(P_{st})) = \sum_{e \in E(P_{st})} c(e)$.

Vrh koji prethodi vrhu t u P_{st} označavamo sa p_{st}. Ako put P_{st} ne postoji, odnosno ako t nije dostižan iz s, problem je trivijalno rješen te pišemo $l_{st} = \infty$, a za p_{st} kažemo da je nedefiniran. Kroz cijeli rad koristit će se jednakosti $n = |V(G)|$ i $m = |E(G)|$. Ovaj problem veoma je primjenjiv u stvarnosti.

Primjer 1. Ivan živi u Zagrebu. Dobio je novi posao te sutra kreće raditi po prvi puta. Ivan na posao ide automobilom. Kako bi postrošio što manje goriva, želi pronaći najkraći put od svog doma (Home) do posla (Office). Da bi ga pronašao, Ivan je istražio sve ulice koje vode do željene destinacije, te ih nacrtao na grafu (Slika 2). Neke ulice su jednosmjerne, a neke dvosmjerne. Jednosmjernim je ulicama Ivan dodao strelicu na grafu.



Slika 2: Primjer 1.

Intuitivno, težina ulice u ovome primjeru njena je duljina. Kako je duljina ulice uvijek pozitivna, problem se rješava primjenom Dijkstrinog[1] algoritma kojeg ćemo u poglavlju 3.1.1 detaljno obraditi.

[1] Edsger Wybe Dijkstra, nizozemski računalni znanstvenik.

Primjer 2. Neka su težine svih bridova u grafu jednake -1. Tražimo sve (s,t) - puteve minimalne duljine, $s,t \in V(G)$. Jasno je kako je rješenje ovoga zadatka najduži (s,t) - put koji postoji na grafu. Ako postoji Hamiltonov (s,t) - put, on će biti rješenje našega problema. Svaki takav put biti će težine $1 - |V(G)|$.

Međutim, traženje Hamiltonovih puteva u usmjerenome grafu može biti veoma težak problem. Općenito, problem može biti veoma teško rješiv dopustimo li da težine bridova budu proizvoljni brojevi u $\mathbb{R}$. Problem postaje mnogo jednostavniji ako ograničimo naš problem na nenegativne težine bridova, ili barem isključimo cikluse negativnih težina.

Definicija 1. Neka je G (usmjereni ili neusmjereni) graf sa funkcijom težine $c : E(G) \to \mathbb{R}$. Kažemo da je c **konzervativna** ako G ne sadrži ciklus negativne težine. Kažemo da je ciklus negativne težine ako je suma težina njegovih bridova negativni realni broj.

U poglavlju 3.1 razradit ćemo dva algoritma za rješavanje problema puta najmanjeg troška. Prvi će dozvoljavati isključivo nenegativne funkcije težine, dok će drugi razmatrati grafove sa konzervativnom funckijom težine (dopuštene su negativne težine pojedinih bridova). Za zadani $s \in V(G)$, algoritmi računaju sve (s,v) - najkraće puteve, za svaki $v \in V(G)$. U poglavlju 3.2 razmatrat ćemo kako pronaći najkraće puteve između svaka dva vrha na grafu.

Traženje najkraćeg puta na neusmjerenim grafovima teži je problem, osim u slučaju ako je funkcija težine nenegativna. Svaki neusmjereni graf može se poistovjetiti sa usmjerenim. Neusmjereni brid nenegativne težine može se zamijeniti dvama usmjerenim bridovima jednake težine i suprotne orijentacije, čime smo problem prenijeli na usmjerene grafove. Međutim, ako težine na neusmjerenom grafu nisu nenegativne, opisani postupak dovodi do nastanka ciklusa (duljine 2) negativne duljine. Kako na netom opisani način svaki neusmjereni graf možemo svesti na usmjereni, u ovome radu razmatrati ćemo samo usmjerene grafove.

Odsada nadalje, bez smanjenja općenitosti možemo pretpostaviti kako su grafovi koje ćemo razmatrati povezani i jednostavni. U suprotnome, između paralelnih bridova uvijek bi koristili onaj s najmanjom težinom, što nebi promijenilo problem kojim se ovdje bavimo. Povezanost intuitivno također ne utječe na naš problem: u slučaju da je graf nepovezan te tražimo najkraće puteve iz vrha v, za svaki vrh w na grafu koji ne pripada istoj komponenti povezanosti kao i vrh v logično zaključujemo da najkraći (v,w) - put ne postoji.

3.1 Najkraći putevi iz jednog izvora (vrha)

Napomena 1. Svi algoritmi najkraćih puteva koje ćemo razraditi bazirani su na Bellmanovom[2] načelu optimalnosti, koje je zapravo srž dinamičkog programiranja.

Propozicija 1. Neka je G usmjereni graf sa konzervativnom funkcijom težine $c : E(G) \to \mathbb{R}$, neka je $k \in \mathbb{N}$ i neka su $s, v, w \in V(G)$. Neka je P najkraći (s, w) - put s najviše k bridova te neka je $e = (v, w)$ posljednji brid u P. Tada je $P_{[s,v]}$ najkraći (s, v) - put s najviše $(k - 1)$ bridova.

Dokaz. Dokaz se izvodi obratom po kontrapoziciji. Dakle, pretpostavimo suprotno. Neka je $Q(s, v)$ - put kraći od $P_{[s,v]}$ te neka je $|E(Q)| \leq k-1$. Tada vrijedi: $c(E(Q))+c(e) < c(E(P))$. Rastavimo nastavak dokaza razmatrajući sljedeća dva slučaja odvojeno:

1) Q ne sadrži vrh w.
   Tada je $Q + e$ (s, w) - put kraći od P, što je u kontradikciji sa pretpostavkom zadatka.
2) Q sadrži vrh w. Sada imamo:

$$\begin{aligned} c(E(Q_{[s,w]})) &= c(E(Q)) + c(e) - c(E(Q_{[w,v]} \cup e)) \\ &< c(E(P)) - c(E(Q_{[w,v]} \cup e)) \leq c(E(P)). \end{aligned}$$

Zadnja nejednakost slijedi iz zbog toga što je $Q_{[w,v]} + e$ ciklus, pa kako je c konzervativna, vrijedi: $c(E(Q_{[w,v]} + e)) \geq 0$. Dakle, dobili smo da je $Q_{[s,w]}$ (s, w) - put kraći od P, što je u kontradikciji sa pretpostavkom zadatka.

■

Propozicija 1. vrijedi i za sve neusmjerene grafove sa nenegativnim funkcijama težine, kao i za sve aciklične usmjerene grafove sa proizvoljnim težinama, rezultirajući sljedećom rekurzivnom formulom:

$$\begin{gathered} \text{dist}_G((s, s)) = 0, \quad \forall s \in V(G); \\ \text{dist}_G((s, w)) = \min\{\text{dist}_G((s, v)) + c((v, w)) \mid (v, w) \in E(G)\}, \quad \forall w \in V(G)\backslash\{s\}. \end{gathered}$$

Dobivena formula predstavlja rješenje problema najkraćeg puta za aciklične grafove. Kada bi postojao algoritam koji bi za proizvoljne $s, t \in V(G)$ mogao pronaći najkraći (s, t) - put P, koristeći Propoziciju 1., iz toga rezultata lako bismo iščitali najkraće (s, v) - puteve, za sve $v \in P$. Međutim, kako je nemoguće unaprijed znati koji vrhovi pripadaju putu P, prirodno je računati najkraće (s, v) - puteve za sve $v \in V(G)$.

Kao što smo ranije napomenuli, prvo ćemo razmatrati grafove sa nenegativnim težinama bridova, tj. težinskim funkcijama oblika $c : E(G) \to (\mathbb{R}_+ \cup \{0\})$. Ovaj problem rješavamo Dijkstrinim algoritmom.

[2]Richard Ernest Bellman. 1953. godine postavio je temelje dinamičkog programiranja.

3.1.1 Dijkstrin algoritam

Za zadani $s \in V(G)$, Dijkstrin algoritam daje najkraće (s, v) - puteve, kao i njihove težine, za sve $v \in V(G)$. Graf G koji promatramo jest usmjeren, a njegova težinska funkcija nenegativna. Sa $l(v)$ označavat ćemo duljinu najkraćega (s, v) - puta, a sa $p(v)$ vrh koji prethodi vrhu v u tome putu. Ako v nije dostižan iz s, pišemo $l(v) = \infty$, a za $p(v)$ kažemo da je nedefiniran. Može se pokazati da je Dijksrin algoritam rješiv u $O(m + n \cdot logn)$ vremenu. Algoritam se provodi kroz sljedećih pet koraka:

① Stavljamo:

- $l(v) := \begin{cases} 0, & \text{za } v = s, \\ c((s,v)), & \text{za } v \in V(G) \text{ t.d. } (s,v) \in E(G), \\ \infty, & \text{inače.} \end{cases}$
- $p(v) := s, \quad \text{za} \quad v \in V(G) \quad \text{t.d.} \quad (s,v) \in E(G),$
- $R := \{s\}.$

② Uzimamo $v \in V(G)\backslash R$ t.d. vrijedi:

$$l(v) = \min_{w \in V(G)\backslash R} l(w).$$

③ Stavljamo $R := R \cup \{v\}$.

④ Za svaki $w \in V(G)\backslash R$ t.d. $(v, w) \in E(G)$ i za kojega vrijedi $l(w) > l(v)+c((v, w))$, stavljamo:

- $l(w) := l(v) + c((v, w)),$
- $p(w) := v.$

⑤ Ako je $R \neq V(G)$, vraćamo se na ②
{u suprotnom, prekidamo algoritam}.

Teorem 1. (Dijkstra [1959]) Dijkstrin algoritam daje ispravne rezultate, tj. nakon sprovedenog algoritma vrijedi $l(v) = \text{dist}_{(G,c)}(s, v)$, za svaki $v \in V(G)$.

Dokaz. Neka je $v \in V(G)\backslash\{s\}$ vrh koji je u koraku ③ dodan u R. Iz koraka ④ vidimo da se za taj vrh duljina $l(v)$ do kraja algoritma ne mijenja. Dokaz ćemo dakle sprovesti indukcijom po vrhovima koji u koraku ③ ulaze u skup R (algoritam traje dok se ne ispuni jednakost $R = V(G)$ pa znamo da ćemo tvrdnju provesti kroz sve vrhove grafa).

Za $v = s$, tvrdnja trivijalno slijedi: $l(s) = 0 = \text{dist}_{(G,c)}(s, s)$. Neka je sada $v \in V(G)\backslash\{s\}$ vrh koji je u koraku ③ dodan u R, i neka je pretpostavka indukcije ispunjena za sve vrhove koji su prije v uvršteni u R.

Pretpostavimo suprotno, tj. da postoji (s, v) - put P u G koji je kraći od $l(v)$. Neka je $y \in V(G)$ prvi vrh na P koji pripada skupu $(V(G)\backslash R) \cup \{v\}$ i neka je $x \in R$ vrh koji prethodi vrhu y u P (kako je $s \in R$ prvi vrh puta P, znamo da takav x postoji). Sada imamo:

$$\begin{aligned} l(y) &\leq l(x) + c((x, y)) \\ &= \text{dist}_{(G,c)}(s, x) + c((x, y)) \leq c(E(P_{[}s, y])) \leq c(E(P)) \\ &< l(v). \end{aligned}$$

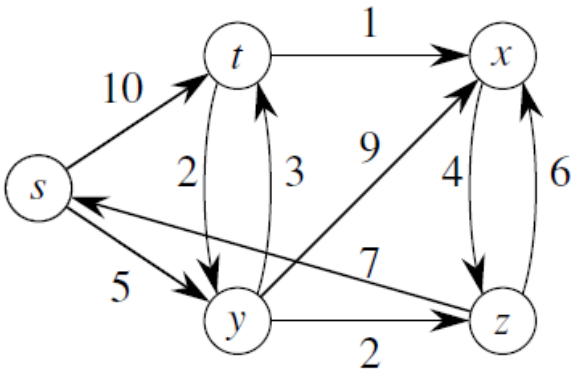
Prva nejednakost slijedi iz $x \in R$, $y \in [(V(G)\backslash R) \cup \{v\}]$, te koraka ④. Jednakost slijedi iz $x \in R$, a posljednja nejednakost slijedi iz pretpostavke da je P kraći od $l(v)$. Dakle, dobili smo $l(y) < l(v)$. Ako je put P bio takav da smo imali $y = v$, jasno je da je tvrdnja kontradiktorna. U suprotnom, ako smo imali $y \neq v$, dobivamo kontradikciju sa izborom vrha v u koraku ②, jer zbog dobivene nejednakosti u koraku ② sigurno nebi izabrali vrh v.

Dakle, obratom po kontrapoziciji zaključujemo da ne postoji (s, v) - put P u G koji je kraći od $l(v)$. Indukcija je gotova.

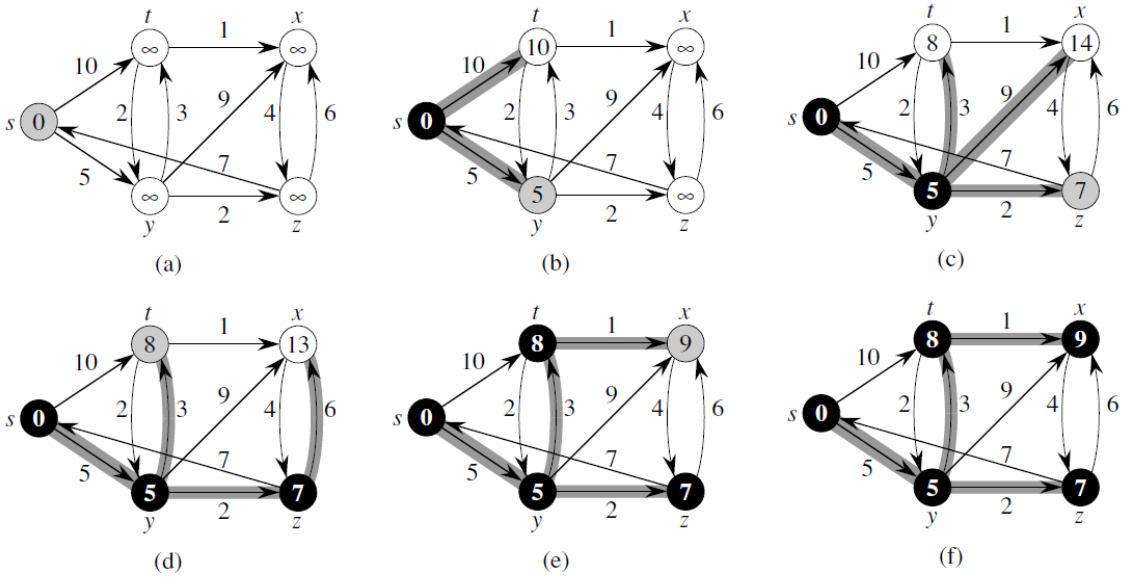
■

Primjer Dijkstrinog algoritma

Neka je zadan Graf G sa slike 3.



Slika 3: Graf G. [2]



Slika 4: Dijkstrin algoritam, 1. način. [2]

Promotrimo sliku 4.

Broj napisan pokraj svakog $v \in V(G)$ označava vrijednost $l(v)$ u gledanom trenutku. Na početku stavljamo $l(s) = 0$. Kako su y i t vrhovi t.d. $(s, y), (s, t) \in E(G)$, u (b) stavljamo $l(y) = 5$, $l(t) = 10$, sukladno težinama pripadajućih bridova. Stavljamo $R = \{s\}$.

Kako je $5 = l(y) = min_{w \in V(G) \setminus R} l(w)$, stavljamo $R = \{s, y\}$. Sva tri vrha skupa $V(G) \setminus R$, t, x i z su takva da vrijedi $(y, t), (y, x), (y, z) \in E(G)$. Kako je $10 = l(t) > l(y) + c((y, t)) = 5 + 3 = 8$, u (c) stavljamo $l(t) = 8$. Kako je $\infty = l(x) > l(y) + c((y, x)) = 5 + 9 = 14$, u (c) stavljamo $l(x) = 14$. Kako je $\infty = l(z) > l(y) + c((y, z)) = 5 + 2 = 7$, u (c) stavljamo $l(z) = 7$. Kako vrijedi $R \subsetneq V(G)$, vraćamo se na korak ②.

Kako je $7 = l(z) = min_{w \in V(G) \setminus R} l(w)$, stavljamo $R = \{s, y, z\}$. Vrh x zadovoljava $(z, x) \in E(G)$, pa zbog $14 = l(x) > l(z) + c((z, x)) = 7 + 6 = 13$, u (d) stavljamo $l(x) = 13$. Kako vrijedi $R \subsetneq V(G)$, vraćamo se na korak ②.

Kako je $8 = l(t) = min_{w \in V(G) \setminus R} l(w)$, stavljamo $R = \{s, y, z, t\}$. Vrh x zadovoljava $(t, x) \in E(G)$, pa zbog $13 = l(x) > l(t) + c((t, x)) = 8 + 1 = 9$, u (e) stavljamo $l(x) = 9$. Kako vrijedi $R \subsetneq V(G)$, vraćamo se na korak ②.

Kako je vrh x jedini preostali vrh u skupu $V(G) \setminus R$, u koraku ② biramo njega, te stavljamo $R = \{s, y, z, t, x\}$. Kako je sada $R = V(G)$, korak ④ više ne dolazi u obzir, a iz istog razloga više se ne vraćamo na korak ②. Algoritam je gotov.

Riješimo sada isti primjer na drugi način, pomoću tablice. Osim što se iz tablice za dani $v \in V(G)$ direktno isčitava duljina najkraćeg (s, v) - puta P_v, iz nje se također lako isčitava i sam P_v, zbog čega je ovaj način najviše korišten u praksi.

v	s	t	x	y	z
s	$\underline{0_s}$	10_s	∞	5_s	∞
y		8_y	14_y	$\underline{5_s}$	7_y
z		8_y	13_z		$\underline{7_y}$
t		$\underline{8_y}$	9_t		
x			$\underline{9_t}$		

Slika 5: Dijkstrin algoritam, 2. način.

Promotrimo sliku 5.

Tablicu popunjavamo po redovima. U prvi red tablice najprije upisujemo slovo v, a nakon njega sve vrhove grafa G. U prvi stupac upisujemo redom vrhove $v \in V(G)$ koji su, zadovoljivši korak ②, ušli u skup R.

Posljednja vrijednost ispisana u stupcu 'v' označava vrijednost $l(v)$ u trenutnoj fazi algoritma. Indeks te vrijednosti (ako ona nije ∞) označava trenutni $p(v)$. Nakon što $v \in V(G)$ uđe u skup R, u istome redu podcrtavamo (i više ne mijenjamo) trenutnu vrijednost $l(v)$ i trenutni $p(v)$, jer iz samog algoritma jasno vidimo da se te vrijednosti više neće mijenjati.

Prvi po algoritmu u skupu R našao se izvorni vrh s. Nakon što smo ga upisali u tablicu, popunjavamo ostatak drugoga reda. U stupac 's' upisujemo $\underline{0_s}$. Ako je $v \in V(G)\backslash\{s\}$ t.d. je $(s, v) \in E(G)$, u stupac 'v' u drugome redu stavljamo $c((s, v))_s$. U suprotnome, pišemo ∞. Primijetimo kako smo na ovaj način napravili korak ①.

Kako vrijedi $R \subsetneq V(G)$, krećemo popunjavati treći red tablice. Prema koraku ②, odabiremo vrh $v \in V(G)$ koji ima minimalni $l(v)$ među svim vrijednostima iz drugoga reda koje nisu podcrtane. Iz drugoga reda isčitavamo da je $l(y) = 5$ minimum tog skupa.

Na početku trećega reda upisujemo y te popunjavamo ostatak reda. Gdjegod je to moguće, smanjujemo trenutne vrijednosti $l(v)$ provodeći korak ④. Nakon što smo ispunili red, kako još uvijek vrijedi $R \subsetneq V(G)$, ponavljamo postupak i odabiremo vrh $v \in V(G)$ koji ima minimalni $l(v)$. Iz trećega reda isčitavamo da je $l(z) = 7$ traženi minimum.

Na početku četvrtoga reda upisujemo z te popunjavamo ostatak reda. Gdjegod je to moguće, smanjujemo trenutne vrijednosti $l(v)$ provodeći korak ④. Nakon što smo ispunili red, kako još uvijek vrijedi $R \subsetneq V(G)$, ponavljamo postupak i odabiremo vrh $v \in V(G)$ koji ima minimalni $l(v)$. Iz četvrtoga reda isčitavamo da je $l(t) = 8$ traženi minimum. Postupak

ponavljamo dok se u prvome stupcu ne pojave svi vrhovi grafa. Nakon što smo popunili zadnji red, tablica je gotova. Jedino što nam preostaje jest pokazati na koji način iz tablice za proizvoljni $v \in V(G)$ isčitavamo najkraći put P_v i njegovu težinu.

Uzmimo za primjer neka je $v = x$. Tražimo red u kojem smo vrh x uvrstili u skup R. To je šesti red tablice. Sada iz šestoga reda i stupca 'x' direktno isčitavamo težinu puta P_x. Dakle, $c(P_x) = 9$. Iz šestoga reda isčitavamo $p(x) = t$, pa se vraćamo na peti red gdje je t bio uvršten u skup R. Iz petoga reda isčitavamo $p(t) = y$, pa se vraćamo na drugi red gdje je y bio uvršten u skup R. Iz drugoga reda isčitavamo $p(y) = s$, čime smo došli do izvornog vrha. Dakle traženi P_x isčitavamo unatrag: $P_x = sytx$.

Razmotrimo sada grafove čiji bridovi nemaju nužno nenegativne težine, ali kojima je težinska funkcija konzervativna. Algoritam traženja najkraćih puteva u takvim grafovima zove se Moore-Bellman-Fordov (MBF)[3] algoritam.

3.1.2 Moore - Bellman - Fordov (MBF) algoritam

Za zadani $s \in V(G)$, MBF algoritam daje najkraće (s, v) - puteve, kao i njihove težine, za sve $v \in V(G)$. Graf G koji promatramo jest usmjeren, a njegova težinska funkcija konzervativna. Neka je $|(V(G)| = n \in \mathbb{N}$. Sa $l(v)$ označavat ćemo duljinu najkraćega (s, v) - puta, a sa $p(v)$ vrh koji prethodi vrhu v u tome putu. Ako v nije dostižan iz s, pišemo $l(v) = \infty$, a za $p(v)$ kažemo da je nedefiniran. Algoritam se provodi kroz sljedeća dva koraka:

① Stavljamo:

- $l(v) := \begin{cases} 0, & \text{za } v = s, \\ \infty, & \text{inače.} \end{cases}$

② Za $i = 1, 2, ..., n-1$:
{ za svaki $(v, w) \in E(G)$:
{ ako je $l(w) > l(v) + c((v, w))$, stavljamo:
$\{l(w) := l(v) + c((v, w))$,
$p(w) := v.$ $\}\}\}$.

Napomena 2. Za svaki $i \in \{1, 2, ..., n-1\}$ koji izvrtimo u algoritmu kažemo da smo napravili jednu iteraciju. Kasnije ćemo vidjeti da algoritam ponekad možemo ubrzati na način da ga jednostavno zaustavimo nakon iteracije u kojoj ne dolazi do nikakve promjene u vrijednostima $l(v)$. Dakle, ukupno radimo najviše $n - 1$ iteracija.

Teorem 2. (Moore [1959], Bellman [1958], Ford [1956]) MBF algoritam daje ispravne rezultate, tj. nakon sprovedenog algoritma vrijedi $l(v) = \text{dist}_{(G,c)}(s, v)$, za svaki $v \in V(G)$.

[3] Edward Forrest Moore i Lester Randolph Ford Jr, američki matematičari.

Dokaz. Neka je $w \in V(G)$ proizvoljan. Dokažimo najprije sljedeću tvrdnju: nakon $k \in \{1, 2, ..., n-1\}$ iteracija algoritma, duljina najkraćeg (s, w) - puta s najviše k bridova jest veća ili jednaka od vrijednosti $l(w)$ u toj fazi algoritma. Dokaz tvrdnje radimo indukcijom po k.

Neka je $k = 1$. Ovdje razlikujemo dva slučaja. U prvom slučaju razmatramo $w \in V(G)$ za koje vrijedi $(s, w) \notin E(G)$. Tokom izvođenja prve iteracije, zbog koraka ① znamo da će za svaki $(v, w) \in E(G)$ uvjet $l(w) > l(v) + c((v, w))$ biti ispunjen ako i samo ako jest $v = s$ (u protivnom, desna strana nejednakosti iznosila bi ∞). Kako imamo $(s, w) \notin E(G)$, lako je zaključiti da će nakon prve iteracije vrijednost $l(w)$ ostati ∞. Dakle, kako (s, w) - put duljine 1 ne postoji, njegovu udaljenost uzimamo kao ∞, što je jednako vrijednosti $l(w) = \infty$ u toj fazi algoritma.

U drugome slučaju razmatramo $w \in V(G)$ za koje vrijedi $(s, w) \in E(G)$. Zbog koraka ① algoritma, lako je vidjeti da će uvjet $l(w) > l(s)+c((s, w))$ biti ispunjen, iz čega zaključujemo da ćemo nakon prve iteracije imati $l(w) = l(s) + c((s, w)) = 0 + c((s, w)) = c((s, w))$. Kako je $P = sw$ jedini mogući (s, w) - put duljine 1, te kako je $c(E(P)) = c((s, w)) = l(w)$, slučaj je dokazan.

Pretpostavimo sada da tvrdnja vrijedi nakon $k - 1$ napravljenih iteracija. Preostaje nam dokazati korak indukcije. Neka je P najkraći (s, w) - put s najviše k bridova i neka je (x, w) posljednji brid u P. Tada je, prema Propoziciji 1., $P_[s, x]$ najkraći (s, x) - put s najviše $k - 1$ bridova pa iz pretpostavke indukcije zaključujemo da nakon $k - 1$ iteracija vrijedi $l(x) \leq c(E(P_[s, x]))$.

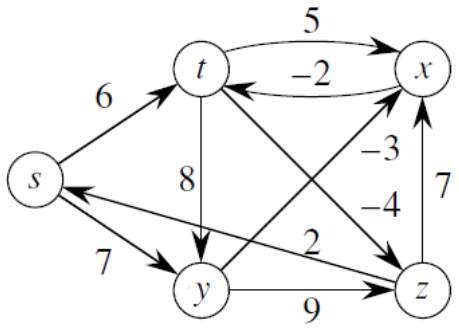
Nadalje, kako je $(x, w) \in E(G)$, te kako su u k - toj iteraciji ponovno obrađeni svi bridovi, dobivamo $l(w) \leq l(x) + c((x, w)) \leq c(E(P_[s, x])) + c((x, w)) = c(E(P))$, čime je korak indukcije dokazan. Kako u grafu s n vrhova ne postoji put s više od $n-1$ bridova, zaključujemo da je dovoljno napraviti $n - 1$ iteracija.

■

Iz samog algoritma jasno je vidljivo kako ne postoji nikakv uvjet na redoslijed obrađivanja bridova. Dakle, sam redoslijed može se razlikovati među pojedinim iteracijama. Međutim, moramo pripaziti da se u svakoj iteraciji obradi svaki brid grafa. Da bi spriječili eventualan propust, obično se prije izvođenja prve iteracije ispišu svi bridovi grafa u proizvoljnom slijedu, te se posljedično, u svakoj iteraciji, oni obrađuju po istome slijedu.

Primjer MBF algoritma

Neka je zadan graf G sa slike 6. Prije no što krenemo izvoditi algoritam, provjeravamo postoji li u grafu ciklus negativne težine. Za dani G, iz slike isčitavamo kako takav ciklus ne postoji, stoga možemo krenuti u sprovedbu algoritma.



Slika 6: Graf G. [2]

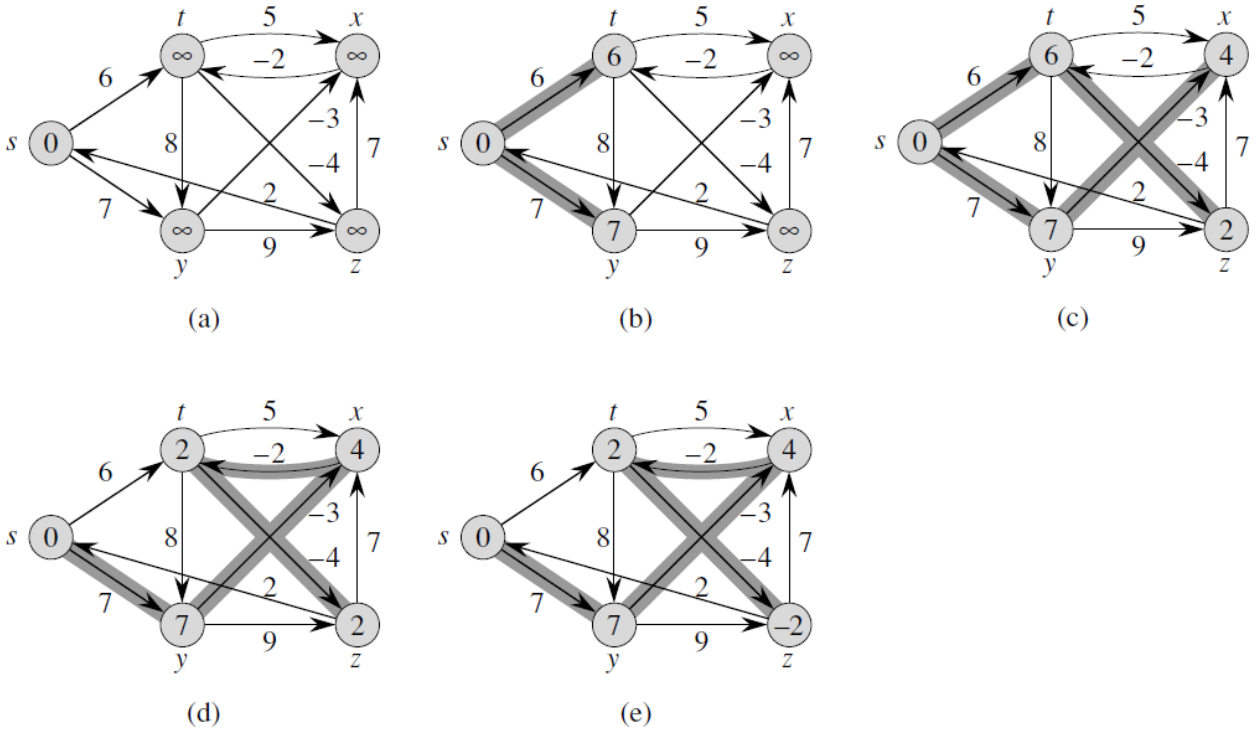
Promotrimo sliku 7.

Broj napisan pokraj svakog $v \in V(G)$ označava vrijednost $l(v)$ u gledanom trenutku. Na početku (slika (a)) stavljamo $l(s) = 0$, $l(v) = \infty$, za sve $v \in V(G)\backslash\{s\}$. Ispišimo sve bridove grafa u proizvoljnom ali fiksnom poretku:

$$(t,x),\ (t,y),\ (t,z),\ (x,t),\ (y,x),\ (y,z),\ (z,x),\ (z,s),\ (s,t),\ (s,y).$$

Stavljamo $i = 1$ i obrađujemo vrijednosti $l(v)$, za sve $v \in V(G)$:

- $l(x) = \infty \not> \infty + 5 = l(t) + c((t,x))$;
- $l(y) = \infty \not> \infty + 8 = l(t) + c((t,y))$;
- $l(z) = \infty \not> \infty - 4 = l(t) + c((t,z))$;
- $l(t) = \infty \not> \infty - 2 = l(x) + c((x,t))$;
- $l(x) = \infty \not> \infty - 3 = l(y) + c((y,x))$;
- $l(z) = \infty \not> \infty + 9 = l(y) + c((y,z))$;
- $l(x) = \infty \not> \infty + 7 = l(z) + c((z,x))$;
- $l(s) = 0 \not> \infty + 2 = l(z) + c((z,s))$;
- $l(t) = \infty > 0 + 6 = l(s) + c((s,t)), \quad \Rightarrow \quad l(t) := 6,\ p(t) := s$;
- $l(y) = \infty > 0 + 7 = l(s) + c((s,y)), \quad \Rightarrow \quad l(y) := 7,\ p(y) := s$.



Slika 7: MBF algoritam. [2]

Dakle, na slici (b) stavljamo nove vrijednosti u vrhovima t i y. Stavljamo $i = 2$ i obrađujemo vrijednosti $l(v)$, za sve $v \in V(G)$:

- $l(x) = \infty > 6 + 5 = l(t) + c((t,x)), \quad \Rightarrow \quad l(x) := 11, \;\; p(x) := t.$
- $l(y) = 7 \not> 6 + 8 = l(t) + c((t,y));$
- $l(z) = \infty > 6 - 4 = l(t) + c((t,z)), \quad \Rightarrow \quad l(z) := 2, \quad p(z) := t.$
- $l(t) = 6 \not> 11 - 2 = l(x) + c((x,t));$
- $l(x) = 11 > 7 - 3 = l(y) + c((y,x)), \quad \Rightarrow \quad l(x) := 4, \quad p(x) := y.$
- $l(z) = 2 \not> 7 + 9 = l(y) + c((y,z));$
- $l(x) = 4 \not> 2 + 7 = l(z) + c((z,x));$
- $l(s) = 0 \not> 2 + 2 = l(z) + c((z,s));$
- $l(t) = 6 \not> 0 + 6 = l(s) + c((s,t));$
- $l(y) = 7 \not> 0 + 7 = l(s) + c((s,y)).$

Dakle, na slici (c) stavljamo nove vrijednosti u vrhovima x i z. Postupak ponavljamo dok ne izvrtimo svih $|V(G)| - 1 = 4$ iteracija.

Sjetimo se kako smo u napomeni 2. rekli kako algoritam ponekad možemo ubrzati na način da ga jednostavno zaustavimo nakon iteracije u kojoj ne dolazi da promjene u vrijednostima $l(v)$. Razlog je i više nego očit. Zašto bi u sljedećim iteracijama ponavljali postupak za koji smo u danoj iteraciji vidjeli da neće donijeti nikakve promjene? S pravom smijemo stati te očitati rješenje problema.

Razmislimo li malo bolje, početni poredak bridova uvelike pridonosi brzini izvođenja algoritma. Dakle, uspijemo li poredati bridove na poželjan način, uvelike ćemo si olakšati posao. Razmotrimo sada na koji način bi 'pametno' poredali slijed bridova. Pokažimo to na istome primjeru grafa G sa slike 6.

Kako bismo algoritam krenuli efikasno (odmah mijenjajući vrijednosti $l(v)$), logično je da ćemo slijed započeti bridovima za koje je izraz $l(w) > l(v) + c((v,w))$ na početku algoritma ispunjen, a to su bridovi kojima je početni vrh s. Dakle, slijed počinjemo bridovima $(s,t), (s,y)$. Slijedeći razmišljanje kako su vrijednosti $l(t)$ i $l(y)$ sada konačne, iz istog razloga nastavljamo bridovima kojima su početni vrhovi t ili y, pa imamo slijed: (s,t), (s,y), (t,y), (t,z), (t,x), (y,z), (y,x).

Zastanimo ovdje na trenutak. Prirodno bi bilo zapitati se je li redoslijed između bridova kojima su početni vrhovi t ili y bitan? Odgovor je potvrdan. Naime, kako u G postoji brid (t,y), mudro je njega staviti na prvo mjesto. Razlog je jednostavan. Tokom obrade tog brida, postoji mogućnost da se vrijednost $l(y)$ smanji, pa će se povećati i vjerojatnosti da se tokom obrade bridova (y,z) i (y,x) smanje vrijednosti $l(z)$ i $l(x)$. Postupak nastavljamo istom logikom razmišljanja te dobivamo slijed:

$$(s,t),\ (s,y),\ (t,y),\ (t,z),\ (t,x),\ (y,z),\ (y,x),\ (z,x),\ (x,t),\ (z,s).$$

Stavljamo $i = 1$ i obrađujemo vrijednosti $l(v)$, za sve $v \in V(G)$:

- $l(t) = \infty > 0 + 6 = l(s) + c((s,t)), \quad \Rightarrow \quad l(t) := 6, \quad p(t) := s;$
- $l(y) = \infty > 0 + 7 = l(s) + c((s,y)), \quad \Rightarrow \quad l(y) := 7, \quad p(y) := s;$
- $l(y) = 7 \not> 6 + 8 = l(t) + c((t,y));$
- $l(z) = \infty > 6 - 4 = l(t) + c((t,z)), \quad \Rightarrow \quad l(z) := 2, \quad p(z) := t;$
- $l(x) = \infty > 6 + 5 = l(t) + c((t,x)), \quad \Rightarrow \quad l(x) := 11, \quad p(x) := t;$
- $l(z) = 2 \not> 7 + 9 = l(y) + c((y,z));$
- $l(x) = 11 > 7 - 3 = l(y) + c((y,x)), \quad \Rightarrow \quad l(x) := 4, \quad p(x) := y;$
- $l(x) = 4 \not> 2 + 7 = l(z) + c((z,x));$
- $l(t) = 6 > 4 - 2 = l(x) + c((x,t)), \quad \Rightarrow \quad l(t) := 2, \quad p(t) := x;$
- $l(s) = 0 \not> 2 + 2 = l(z) + c((z,s)).$

Pamtimo dobivene vrijednosti $l(v)$ i nastavljamo dalje. Stavljamo $i = 2$ i obrađujemo vrijednosti $l(v)$, za sve $v \in V(G)$:

- $l(t) = 2 \not> 0 + 6 = l(s) + c((s,t))$;
- $l(y) = 7 \not> 0 + 7 = l(s) + c((s,y))$;
- $l(y) = 7 \not> 2 + 8 = l(t) + c((t,y))$;
- $l(z) = 2 > 2 - 4 = l(t) + c((t,z)), \quad \Rightarrow \quad l(z) := -2, \;\; p(z) := t$;
- $l(x) = 4 \not> 6 + 5 = l(t) + c((t,x))$;
- $l(z) = -2 \not> 7 + 9 = l(y) + c((y,z))$;
- $l(x) = 4 \not> 7 - 3 = l(y) + c((y,x))$;
- $l(x) = 4 \not> 2 + 7 = l(z) + c((z,x))$;
- $l(t) = 2 \not> 4 - 2 = l(x) + c((x,t))$;
- $l(s) = 0 \not> -2 + 2 = l(z) + c((z,s))$.

Postupak nastavljamo te već u sljedećoj iteraciji, $i = 3$, ne dobivamo promjene. Dakle, bilo je potrebno izvrtiti algoritam samo 3 puta i dobiti rješenje. Općenito, pametnim biranjem redoslijeda bridova često se dogodi da već druga iteracija ne izbaci nikakvu promjenu. U našem slučaju, radi velikom broja bridova u grafu, trebali smo napraviti i treću iteraciju.

Primijetimo kako je graf G u ovome primjeru bio relativno jednostavan te se iz same slike lako isčitava kako ciklus negativne težine ne postoji. Međutim, problem nastaje kako se broj vrhova i bridova grafa povećava. Kao zaključak poglavlja 2.1 biti će obrađen Moore-Bellman-Fordov (MBF) algoritam za traženje ciklusa negativne težine.

Prikazani algoritam može se dodatno popratiti popunjavanjem tablice. Tablica je ovdje pogodna iz istog razloga kao i kod Dijkstrinog algoritma; osim što iz nje za dani $v \in V(G)$ direktno isčitavamo duljinu najkraćeg (s,v) - puta P_v, također lako isčitavamo i sam P_v, zbog čega se tablica veoma često koristi u praksi. Napraviti ćemo dvije tablice, prva će odgovarati prvome slijedu bridova kojeg smo imali, a druga drugome - onome u kojem smo bridove poredali na "pametan" način.

Promotrimo sliku 8.

Tablicu popunjavamo po redovima. U prvi red tablice najprije upisujemo slovo i, a nakon njega sve vrhove grafa G. U prvi stupac upisujemo redom brojeve $0, 1, 2, ...$ koji označuju redni broj iteracije algoritma u kojoj se trenutno nalazimo. Posljednja vrijednost ispisana u stupcu 'v', $v \in V(G)$, označava vrijednost $l(v)$ u trenutnoj fazi algoritma. Indeks te vrijednosti (ako ona nije ∞) označava trenutni $p(v)$.

i	s	t	x	y	z
0	$\underline{0_s}$	∞	∞	∞	∞
1		6_s	∞	7_s	∞
2		6_s	4_y	7_s	2_t
3		2_x	4_y	7_s	2_t
4		2_x	4_y	7_s	-2_t
5		2_x	4_y	7_s	-2_t

Slika 8: Pomoćna tablica za MBF algoritam rješen korištenjem prvog slijeda bridova.

Na početku drugoga reda upisujemo broj 0 koji označava korak ① (dakle, to nije prva iteracija). Nakon što smo ga upisali u tablicu, popunjavamo ostatak drugoga reda. U stupac 's' upisujemo $\underline{0_s}$, a kako iz samog algoritma znamo da se vrijednosti $l(s)$ i $p(s)$ više neće mijenjati, taj stupac u sljedećim iteracijama više ne trebamo razmatrati.

Na početku trećega reda upisujemo broj 1 koji označava prvu iteraciju algoritma, te zatim popunjavamo ostatak trećega reda. Ako je $v \in V(G)\backslash\{s\}$ t.d. se vrijednost $l(v)$ u obradi prve iteracije promijenila, u stupac 'v' u trećemu redu upisujemo novedobivene vrijednosti $l(v)$ i $p(v)$. U suprotnome, prepisujemo prethodne vrijednosti.

Postupak ponavljamo dok se ne pojavi iteracija 'i_*' u kojoj neće biti promjene u odnosu na prethodnu iteraciju '$i_* - 1$'. Preostaje još pokazati na koji način iz tablice za proizvoljni $v \in V(G)$ isčitavamo najkraći put P_v i njegovu težinu.

Uzmimo za primjer neka je $v = z$. Poslužimo se slikom 8. Iz stupca 'z' te iz posljednjeg reda tablice isčitavamo $p(z) = t$. Sada iz stupca 't' te iz posljednjeg reda tablice isčitavamo $p(t) = x$. Iz stupca 'x' te iz posljednjeg reda tablice isčitavamo $p(x) = y$. Iz stupca 'y' te iz posljednjeg reda tablice isčitavamo $p(y) = s$, čime smo došli do izvornog vrha. Dakle traženi P_z isčitavamo unatrag: $P_z = syxtz$. Njegovu težinu isčitavamo iz zadnjeg retka stupca 'z', $c(E(P_z)) = -2$.

i	s	t	x	y	z
0	$\underline{0_s}$	∞	∞	∞	∞
1		2_x	4_y	7_s	2_t
2		2_x	4_y	7_s	-2_t
3		2_x	4_y	7_s	-2_t

Slika 9: Pomoćna tablica za MBF algoritam rješen korištenjem "pametnog" slijeda bridova.

Promotrimo sliku 9.

Slika prikazuje opisani postupak sproveden na "pametno" odabranome slijedu bridova grafa G. Kao i iz samog algoritma, iz tablice čitamo da su ovdje potrebne bile samo 3 iteracije algoritma.

Iz samoga algoritma jasno je vidljivo kako je njegova složenost $O(nm)$, gdje su $n = |V(G)|$, $m = |E(G)|$. MBF algoritam najbrži je poznati algoritam za traženje puta najmanjeg troška na grafovima sa konzervativnim funkcijama težine. U slučajevima kada je graf G planaran, postoji algoritam složenosti $O(nlog^3n)$.

Za grafove koji sadrže ciklus negativne težine, još uvijek nije pronađen algoritam polinomne složenosti. Problem se krije u tome što Propozicija 1. općenito ne vrijedi za grafove čije težinske funkcije nisu konzervativne.

Traženje ciklusa negativne težine

Definicija 2. Neka je G usmjeren graf, neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija i neka je $\pi : V(G) \to \mathbb{R}$ funkcija. Tada za svaki $(x, y) \in E(G)$ definiramo **smanjeni trošak brida (x, y) obzirom na π** na sljedeći način: $c_\pi((x, y)) := c((x, y)) + \pi(x) - \pi(y)$. Ako je $c_\pi(e) \geq 0$ za sve $e \in E(G)$, funkciju π zovemo **ostvarljiv potencijal**.

Lema 1. Neka je G usmjeren graf i neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija. U proizvoljnoj fazi MBF algoritma definiramo $F := \{(x, y) \in E(G) \mid x = p(y)\}$. Tada za sve $(v, w) \in F$ vrijedi $l(w) \geq l(v) + c(v, w)$.

Dokaz. Pri uvrštavanju brida (v, w) u skup F, vrijednosti $l(w)$ i $p(w)$ bile su redefinirane sa $l(w) := l(v) + c(v, w)$, $p(w) := v$. U kasnijem sprovođenju MBF algoritma, vrh v mogao je promijeniti svog prethodnika, $p(v)$, čime bi došlo do smanjenja vrijednosti $l(v)$, a posljedično i do nejednakosti $l(w) > l(v) + c(v, w)$, koja je uključena u tvrdnju leme. Međutim, ono što mi zapravo trebamo razmotriti jest smanjenje vrijednosti $l(w)$, čime tvrdnja leme više nebi vrijedila. Kako znamo da smanjenje spomenute vrijednosti kao nužnu posljedicu ima promjenu prethodnika $p(w)$, tada brid (v, w) više neće biti element skupa F, čime je lema dokazana.

■

Lema 2. Neka je G usmjeren graf i neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija. U proizvoljnoj fazi MBF algoritma definiramo $F := \{(x, y) \in E(G) \mid x = p(y)\}$. Ako F sadrži ciklus C, tada je $c(C) \leq 0$.

Dokaz. Neka je C ciklus s bridovima iz F, te pretpostavimo kako je C bio završen (zatvoren) dodavanjem brida $e = (x, y)$. Prema definiciji skupa F, tada imamo $l(y) = l(x) + c((x, y))$. Međutim, iz Leme 1. znamo kako za svaki $(v, w) \in E(C)\backslash\{(x, y)\}$ vrijedi $l(w) \geq l(v) + c(v, w)$. Za $z \in E(G)$ uvedimo oznaku $z = (z_1, z_2)$. Sada imamo:

$$\begin{aligned} c(C) &= \sum_{z \in E(C)} c(z) \\ &= \sum_{z \in E(C)\backslash\{(x,y)\}} c(z) + c((x, y)) \\ &\leq \sum_{z \in E(C)\backslash\{(x,y)\}} (l(z_2) - l(z_1)) + c((x, y)) \\ &= l(x) - l(y) + c((x, y)) \\ &= 0. \end{aligned}$$

■

Teorem 3. Neka je G usmjeren graf i neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija. Tada postoji ostvarljiv potencijal π na (G, c) ako i samo ako je funkcija c konzervativna.

Dokaz. Neka je π ostvarljiv potencijal. Tada za svaki ciklus C na grafu G vrijedi:

$$0 \leq \sum_{e \in E(C)} c_\pi(e) = \sum_{e=(x,y) \in E(C)} (c(e) + \pi(x) - \pi(y)) = \sum_{e \in E(C)} c(e) = c(C).$$

Dokažimo sada suptrotni smjer. Neka je funkcija c konzervativna. Sprovedbom MBF algoritma na grafu G, za svaki $v \in V(G)$ dobivamo broj $l(v)$. Znamo da tada za svaki $(v, w) \in E(G)$ vrijedi $l(w) \leq l(v) + c((v, w))$ (u suprotnom, MBF algoritam nebi radio ispravno), iz čega direktno slijedi kako je l ostvarljiv potencijal.

■

Korolar 1. Neka je G usmjeren graf i neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija. Tada u vremenu $O(nm)$ možemo pronaći ostvarljiv pontencijal ili ciklus negativne težine.

Dokaz. Složenost algoritma je očita. Za dani graf G sprovodimo MBF algoritam te dobivamo vrijednosti $l(v)$, ze sve $v \in V(G)$. Ako je dobivena funkcija $l : V(G) \to \mathbb{R}$ ostvarljiv potencijal, dokaz je gotov. U suprotnome, neka je $(v, w) \in E(G)$ takav da nakon sprovednog algoritma vrijedi $l(w) > l(v) + c((v, w))$. Tvrdimo da niz $w, v, p(v), p(p(v)), ...$ sadrži ciklus negativne težine.

Iz samog algoritma vidimo kako je vrijednost $l(v)$ bila promijenjena u posljednoj iteraciji algoritma (u suprotnom, radi obrade brida (v, w) u zadnjoj iteraciji, nejednakost $l(w) > l(v) + c((v, w))$ nebi vrijedila). Nadalje, zaključujemo da je prije promjene vrijednosti $l(v)$

vrijedilo $l(v) > l(p(v)) + c((p(v), v))$, iz čega zaključujemo kako je vrijednost $l(p(v))$ bila promijenjena u jednoj od dvaju posljednjih iteracija (ovisno o tome je li brid $(p(v), v)$ u slijedu bridova bio obrađivan prije ili poslije brida (v, w)). Iz istih razloga zaključujemo kako je vrijednost $p(p(p(v)))$ bila promijenjena u jednoj od zadnjih triju iteracija, itd.

Kako znamo da se vrijednost $l(s)$ ne mijenja tokom algoritma, zaključujemo da se vrh s ne nalazi među prvih $n = |V(G)|$ elemenata niza $w, v, p(v), p(p(v)), ...$ (algoritam ponavlja iteraciju $n-1$ puta). Dakle, jedan od vrhova u danome nizu sigurno se pojavljuje dvaput, iz čega zaključujemo da iz danog niza možemo formirati ciklus. Tada iz Leme 2. te iz nejednakosti $l(w) > l(v) + c((v, w))$ direktno zaključujemo kako on ima negativnu težinu, čime je dokaz korolara gotov.

■

Rezimirajmo dakle način na koji nalazimo ciklus negativne težine ukoliko nakon sprevedenog MBF algoritma dobivena funkcija $l : V(G) \to \mathbb{R}$ nije ostvarljiv potencijal. Kako smo netom ranije nad grafom izvodili MBF algoritam, možemo uzeti isti slijed bridova kojeg smo koristili i u samome algoritmu. Uzimamo, dakle, bridove iz danoga slijeda redom sve dok ne pronađemo brid $e = (v, w)$ za kojeg vrijedi li $l(w) > l(v) + c((v, w))$. Tada ispisujemo niz vrhova $w, v, p(v), p(p(v)), ...$, a zaustavljamo se kada se jedan od vrhova pojavi po drugi put. Taj će slijed vrhova biti traženi ciklus negativne težine.

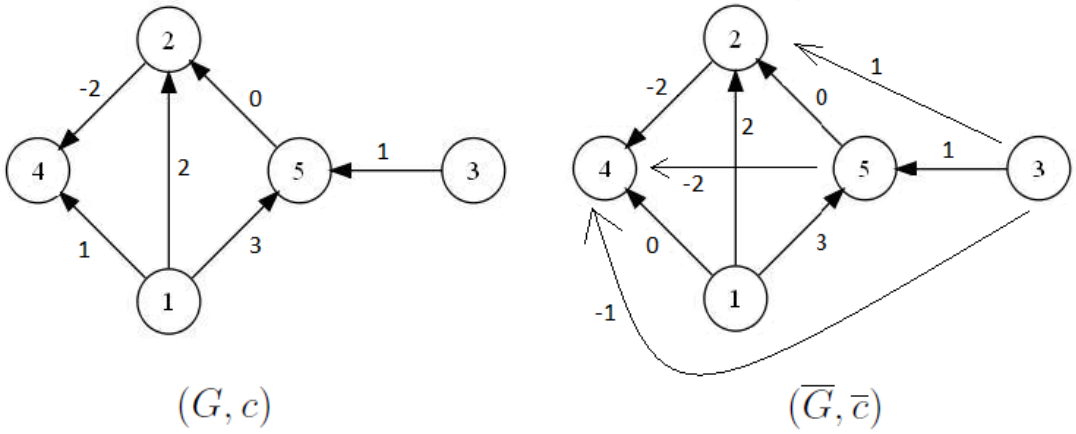
3.2 Najkraći putevi između svaka dva vrha na grafu

Neka je G usmjeren graf i neka je njegova težinska funkcija $c : E(G) \to \mathbb{R}$ konzervativna. Za svaki uređen par (s,t), $s,t \in V(G)$, $s \neq t$, želimo pronaći broj l_{st} i $p_{st} \in V(G)$ takve da je l_{st} duljina najkraćeg (s,t) - puta P_{st}, a (p_{st},t) posljednji brid u P_{st}. Ako za neki uređen par (s,t) put P_{st} ne postoji, odnosno ako t nije dostižan iz s, problem je trivijalno rješen i pišemo $l_{st} = \infty$, a za p_{st} kažemo da je nedefiniran.

Intuitivno, jedna od opcija jest izvršiti MBF algoritam n puta (po jednom za svaki novi izbor vrha s). Time dobivamo algoritam složenosti $O(n^2m)$. Može se pokazati da je problem najkraćeg puta između svaka dva vrha na grafu rješiv u $O(mn + n^2 logn)$ vremenu. Dobiveno rješenje ovoga problema omogućuje nam definirati zatvorenje grafa G.

Definicija 3. Neka je G (usmjeren ili neusmjeren) graf i neka je $c : E(G) \to \mathbb{R}$ njegova težinska funkcija. **Zatvorenje uređenog para (G, c)** je uređen par $(\overline{G}, \overline{c})$, gdje je $\overline{G}$ jednostavan graf sa skupom vrhova $V(\overline{G}) = V(G)$ takav da između vrhova $x, y \in V(\overline{G})$, $x \neq y$ postoji brid $e = \{x, y\}$ (ili $e = (x, y)$ ako je G usmjeren) težine $\overline{c}(e) = \text{dist}_{(G,c)}(x, y)$ ako i samo ako je y dostižan iz x u G.

Zatvorenje danog grafa G rješivo je u $O(mn + n^2 logn)$ vremenu (dovoljno je pronaći najkraće puteve između svaka dva vrha na grafu). Pritom, kako se svaki neusmjereni graf može zamjeniti usmjerenim poistovjećivanjem svakog neusmjerenog brida dvama usmjerenima jednake težine i suprotne orijentacije, nije potrebno zahtijevati da graf G mora biti usmjeren.



Slika 10: Zatvorenje grafa (G, c).

3.2.1 Floyd - Warshall (FW) algoritam

Za zadani usmjereni graf G sa skupom vrhova $V(G) = \{1, 2, ..., n\}$ i konzervativnom funkcijom težine $c : E(G) \to \mathbb{R}$, FW algoritam daje matrice $D = (l_{ij})_{1 \leq i,j \leq n}$ i $\Pi = (p_{ij})_{1 \leq i,j \leq n}$, gdje je l_{ij} duljina najkraćeg (i, j) - puta P_{ij}, a (p_{ij}, j) posljednji brid puta P_{ij}. Ako za neki uređen par (s, t) put P_{st} ne postoji, odnosno ako t nije dostižan iz s, pišemo $l_{st} = \infty$, a za p_{st} kažemo da je nedefiniran. FW algoritam jest najpoznatiji algoritam za rješavanje problema najkraćeg puta između svaka dva vrha na grafu. Algoritam se provodi kroz sljedeća dva koraka:

① Stavljamo:

- $l_{ij} := \begin{cases} 0, & \text{za } j = i, \\ c((i,j)), & \text{za } (i,j) \in E(G), \\ \infty, & \text{inače.} \end{cases}$
- $p_{ij} := i, \quad \text{za sve} (i,j) \in E(G).$

② Za $j = 1, 2, ..., n$:
{ za $i = 1, 2, ..., n$:
{ ako je $i \neq j$, tada:
{ za $k = 1, 2, ..., n$:
{ ako je $k \neq j$, tada:
{ ako je $l_{ik} > l_{ij} + l_{jk}$, stavljamo:
$\{l_{ik} := l_{ij} + l_{jk}$,
$p_{ik} := p_{jk}$. }}}}}}.

Napomena 3. Petlju 'j' u FW algoritmu zovemo vanjskom petljom algoritma. Kao što je slučaj i kod MBF algoritma, FW algoritam može se koristiti pri ispitivanju egzistencije ciklusa negativne težine u grafu.

Teorem 4. (Floyd [1962], Warshall [1962]) FW algoritam daje ispravne rezultate, tj. nakon sprovedenog algoritma vrijedi $l_{ij} = \text{dist}_{(G,c)}(i, j)$, za sve $i, j \in V(G)$. Složenost algoritma jest $O(n^3)$.

Dokaz. Složenost algoritma je očita.
Tvrdnja: Nakon što je algoritam završio vanjsku petlju za $j = 1, 2, ..., j_0$, vrijednost l_{ik} prikazuje duljinu najkraćeg (i, k) - puta koji smije sadržavati unutarnje vrhove isključivo iz skupa $\{1, 2, ..., j_0\}$. Tvrdnju dokazujemo indukcijom po $j_0 = 1, 2, ..., n$. Primijetimo kako će tvrdnja u slučaju $j_0 = n$ dokazivati ispravnost algoritma.

Neka je $j_0 = 1$. Primijetimo za početak kako za svaki p_{ij} koji je definiran u koraku ① vrijedi $p_{ij} = i$. Nakon što obradimo vanjsku petlju algoritma za $j = 1$, iz $p_{ik} := p_{jk} = j = 1$ vidimo kako jedino vrh 1 ima priliku ući te skratiti duljinu trenutno najkraćeg puta P_{ik}, čime smo dokazali bazu indukcije.

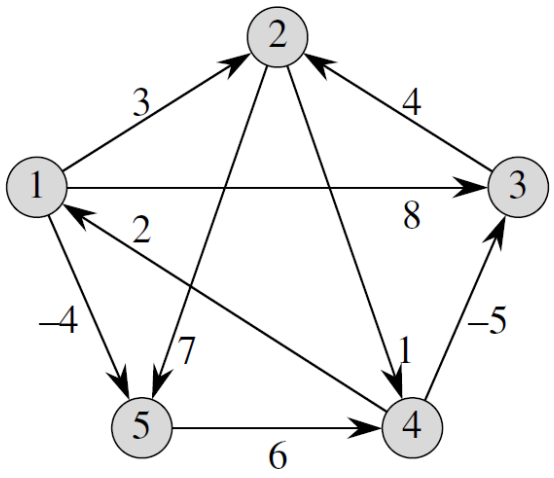
Pretpostavimo kako tvrdnja vrijedi za $j_0 \in \{1, 2, ..., j\}$. Tokom obrade vanjske petlje algoritma za $j_0 = j + 1$, vrijednost l_{ik} biti će zamijenjena vrijednošću $l_{i,j+1} + l_{j+1,k}$ ako vrijedi $l_{ik} > l_{i,j+1} + l_{j+1,k}$. Kako su vrijednosti $l_{i,j+1}$ i $l_{j+1,k}$ dobivene u jednoj od prethodnih obrada vanjske petlje algoritma, prema pretpostavci indukcije njihovi odgovarajući putevi $P_{i,j+1}$ i $P_{j+1,k}$ sadrže unutarnje vrhove isključivo iz skupa $\{1, 2, ..., j\}$. Dakle, novodobiveni put P_{ik} sadrži unutarnje vrhove isključivo iz skupa $\{1, 2, ..., j + 1\}$. Jedino što preostaje pokazati jest da je P_{ik} najkraći takav (i, k) - put.

Kako je pri samom formiranju novoga P_{ik} puta tokom obrade $j+1$ vanjske petlje algoritma prethodna vrijednost l_{ik} bila smanjena, jedini razlog zbog kojeg novodobiveni P_{ik} - put R nebi bio najkraći takav jest ako odgovarajući $(i, j + 1)$ - put P i odgovarajući $(j + 1, k)$ - put Q sadrže zajednički unutarnji vrh. U tome slučaju, put $R = P + Q$ mogli bi skratiti na način da izbacimo odgovarajući ciklus. Međutim, micanjem toga ciklusa izbacili bi i vrh $j + 1$, pa bi zbog početne nejednakosti $l_{ik} > l_{i,j+1} + l_{j+1,k}$ put R (koji sada sadrži unutarnje vrhove iz skupa $\{1, 2, ..., j\}$) bio kraći (i, k) - put od onoga koji je kao takav bio definiran u j -toj obradi vanjske petlje algoritma, što je u kontradikciji sa pretpostavkom indukcije.

■

Primjer FW algoritma

Neka je zadan graf G sa slike 11. Prije no što krenemo izvoditi algoritam, provjeravamo postoji li u grafu ciklus negativne težine. Sprovedbom MBF algoritma za traženje ciklusa negativne duljine lako se provjeri da isti ne postoji.



Slika 11: Graf G. [2]

Promotrimo sliku 12.

Početne matrice $D^{(0)}$ i $\Pi^{(0)}$ dobili smo sprovedbom koraka ①. Svaku sljedeću matricu $D^{(i)}$ i $\Pi^{(i)}$, $i = 1, 2, 3, 4, 5$ dobili smo obradom odgovarajuće vanjske petlje $j = 1, 2, 3, 4, 5$.

Stavljamo $j = 1$ te obrađujemo unutarnje petlje i i k. Slučajeve u kojima je $i = j$ ili $k = j$ netrebamo obrađivati jer iz samog algoritma vidimo kako neće donijeti nikakve promjene. Slučajeve $k = i$ također nemoramo obrađivati jer se vrijednost l_{ik} sigurno neće mijenjati tokom obrade takvog slučaja. Ta tvrdnja slijedi iz početne jednakosti $l_{ii} = 0$, za svaki i, pa kada bi postojao P_{ii} - put težine manje od 0, to bi bio ciklus negativne težine.

$$\underline{\mathbf{j = 1},\ \mathbf{i = 2}:}$$

- $l_{23} = \infty \not> \infty + 8 = l_{21} + l_{13}$;
- $l_{24} = 1 \not> \infty + \infty = l_{21} + l_{14}$;
- $l_{25} = 7 \not> \infty - 4 = l_{21} + l_{15}$.

$$\underline{\mathbf{j = 1},\ \mathbf{i = 3}:}$$

- $l_{32} = 4 \not> \infty + 3 = l_{31} + l_{12}$;
- $l_{34} = \infty \not> \infty + \infty = l_{31} + l_{14}$;
- $l_{35} = \infty \not> \infty - 4 = l_{31} + l_{15}$.

$$\underline{\mathbf{j = 1},\ \mathbf{i = 4}:}$$

- $l_{42} = \infty > 2 + 3 = l_{41} + l_{12}, \quad \Rightarrow \quad l_{42} := 5, \quad p_{42} := 1$;
- $l_{43} = -5 \not> 2 + 8 = l_{41} + l_{13}$;
- $l_{45} = \infty > 2 - 4 = l_{41} + l_{15}, \quad \Rightarrow \quad l_{45} := -2, \quad p_{42} := 1$.

$$\underline{\mathbf{j = 1},\ \mathbf{i = 5}:}$$

- $l_{52} = \infty \not> \infty + 8 = l_{51} + l_{12}$;
- $l_{53} = 1 \not> \infty + \infty = l_{51} + l_{13}$;
- $l_{54} = 7 \not> \infty - 4 = l_{51} + l_{14}$.

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Slika 12: FW algoritam. [2]

Ispunjavamo matrice $D^{(1)}$ i $\Pi^{(1)}$ upravo dobivenim podatcima. Stavljamo $j = 2$ te obrađujemo unutarnje petlje i i k.

$$\underline{\mathbf{j = 2},\ \mathbf{i = 1}:}$$

- $l_{13} = \ \ 8 \ \not> \ 3 + \infty = l_{12} + l_{23}$;
- $l_{14} = \ \infty \not> \ 3 + \ 1 \ = l_{12} + l_{24}, \quad \Rightarrow \quad l_{14} := 4, \ \ p_{14} := 2$;
- $l_{15} = -4 \not> \ 3 + \ 7 \ = l_{12} + l_{25}$.

$$\underline{\mathbf{j = 2},\ \mathbf{i = 3}:}$$

- $l_{31} = \ \infty \not> \ 4 + \infty = l_{32} + l_{21}$;
- $l_{34} = \ \infty \not> \ 4 + \ 1 \ = l_{32} + l_{24}, \quad \Rightarrow \quad l_{34} := 5, \quad p_{34} := 2$;
- $l_{35} = \ \infty \not> \ 4 + \ 7 \ = l_{32} + l_{25}, \quad \Rightarrow \quad l_{35} := 11, \ \ p_{35} := 2$.

$$\underline{\mathbf{j = 2},\ \mathbf{i = 4}:}$$

- $l_{41} = \ \ 2 \ \not> \ 5 + \infty = l_{42} + l_{21}$;
- $l_{43} = -5 \not> \ 5 + \infty = l_{42} + l_{23}$;
- $l_{45} = -2 \not> \ 5 + \ 7 \ = l_{42} + l_{25}$.

$$\underline{\mathbf{j = 2},\ \mathbf{i = 5}:}$$

- $l_{51} = \ \infty \not> \ \infty + \infty = l_{52} + l_{21}$;
- $l_{53} = \ \infty \not> \ \infty + \infty = l_{52} + l_{23}$;
- $l_{54} = \ \ 6 \ \not> \ \infty + \ 1 \ = l_{52} + l_{24}$.

Ispunjavamo matrice $D^{(2)}$ i $\Pi^{(2)}$ upravo dobivenim podatcima. Postupak nastavljamo sve dok ne ispunimo matrice $D^{(|V(G)|)} = D^{(5)}$ i $\Pi^{(|V(G)|)} = \Pi^{(5)}$.

Preostaje još pokazati na koji način za zadane $i, j \in V(G)$ isčitamo najkraći (i, j) - put P_{ij} i njegovu težinu. Uzmimo za primjer neka su $i = 5$, $j = 3$. Iz matrice $D^{(5)}$ direktno isčitavamo $c(E(P_{53})) = 1$. Iz matrice $\Pi^{(5)}$ isčitavamo $p_{53} = 4$, pa zaključujemo kako je $(4, 3)$ posljednji brid u P_{53}. Sada tražimo posljednji brid puta P_{54}, pa iz matrice $\Pi^{(5)}$ isčitavamo $p_{53} = 5$. Kako je 5 početni vrh puta P_{53} ovdje se zaustavljamo. Traženi P_{53} je tada: $P_{53} = 543$.

4 Mrežni tokovi

Neka je G usmjeren graf i neka je $u : E(G) \to \mathbb{R}_+$ njegova nenegativna težinska funkcija. Neka su $s, t \in V(G)$ dva vrha grafa G, pri čemu vrh s nazivamo **izvor**, a vrh t **ponor**. Uređena četvorka (G, u, s, t) naziva se **mreža**.

Definicija 4. Neka je G usmjeren graf i neka je $u : E(G) \to \mathbb{R}_+$ njegova težinska funkcija. **Tok** je funkcija $f : E(G) \to \mathbb{R}_+$ takva da za svaki $e \in E(G)$ vrijedi $f(e) \leq u(e)$. Kažemo da **f čuva tečnost u vrhu v** ako vrijedi :

$$\text{ex}_f(v) := \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) = 0.$$

Definicija 5. Tok koji čuva tečnost u svakom vrhu grafa naziva se **cirkulacija**. $\boldsymbol{(s,t)}$ **- tok** je tok f za kojeg vrijedi $\text{ex}_f(s) < 0$ i $\text{ex}_f(v) = 0$, za sve $v \in V(G)\backslash\{s,t\}$. **Vrijednost $\boldsymbol{(s,t)}$ - toka f** definirana je sa $|f| := -\text{ex}_f(s)$.

Primijetimo kako se pri traženju (s,t) - toka maksimalne vrijednosti u mreži (G,u,s,t) bez smanjenja općenitosti možemo ograničiti na isključivo jednostavne grafove, jer se u suprotnom težine paralelnih bridova mogu zbrojiti ne mijenjajući problem kojim se ovdje bavimo. Ovaj problem veoma je primjenjiv u stvarnosti.

Primjer 3. Neka je zadano $n \in \mathbb{N}$ poslova koje trebamo obaviti. Neka su poslovi označeni brojevima $1, 2, ..., n$. Na raspolaganju imamo $m \in \mathbb{N}$ zaposlenika koje moramo rasporediti po poslovima. Neka je M skup zaposlenika. Svaki posao i zahtijeva $t_i \in \mathbb{R}_+$ uloženog vremena da bi bio završen i za svaki posao i postoji neprazan skup $S_i \subseteq M$ zaposlenika sposobnih za obavljanje i - tog posla. Zaposlenici su međusobno sposobni raditi isti posao istovremeno. Neka je zadano vrijeme $T \in \mathbb{R}_+$ u kojem svi poslovi moraju biti obavljeni. Dakle, cilj je pronaći brojeve $x_{ij} \in \mathbb{R}_+$, $i = 1, 2, ..., n$, $j = 1, 2, ..., m$ (gdje x_{ij} označava vrijeme koje je j - ti zaposlenik uložio u obavljanje i - tog posla) takve da vrijedi:

$$\sum_{j \in S_i} x_{ij} = t_i, \text{ za sve } i \in \{1, 2, ..., n\}, \quad \sum_{\substack{i \\ j \in S_i}} x_{ij} \leq T, \text{ za sve } j \in \{1, 2, ..., m\}.$$

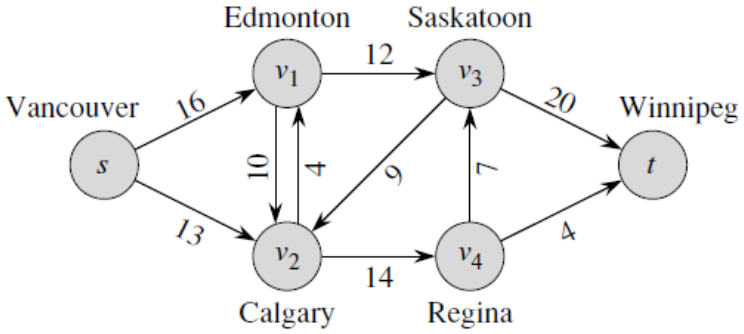
Ovaj problem može se riješiti linearnim programiranjem, no mi ćemo ga ovdje postaviti u kontekstu mrežnog toka. Kreirajmo graf G crajući n vrhova $v_1, v_2, ..., v_n$, gdje v_i predstavlja posao i te m vrhova $w_1, w_2, ..., w_m$, gdje w_j predstavlja zaposlenika j. Dodajemo bridove (v_i, w_j) kada je ispunjeno $j \in S_i$. Na grafu još dodajemo vrhove s i t te bridove (s, v_i) za sve i i (wj, t) za sve j. Definiramo funkciju težine $c : E(G) \to \mathbb{R}+$ sa $u((s, v_i)) := t_i$ i $u(e) = T$ za sve ostale bridove. Tada je svaki (s,t) - tok vrijednosti $\sum_{i=1}^{n} t_i$ u (G,u) (ako takav postoji) rješenje našega problema.

4.1 Tok maksimalne vrijednosti

Neka je (G, u, s, t) mreža. U ovome poglavlju bavit ćemo se traženjem (s, t) - toka maksimalne vrijednosti. Neka je $|E(G)| = m$. Neka je traženi tok oblika $(x_1, x_2, ..., x_m)$. Problem tada možemo zapisati u obliku:

$$\begin{aligned}
&\sum_{e\in\delta^+(s)} x_e - \sum_{e\in\delta^-(s)} x_e && \rightarrow max \\
&\sum_{e\in\delta^-(s)} x_e = \sum_{e\in\delta^+(s)} x_e && (v \in V(G)\backslash\{s,t\}) \\
&x_e \leq u(e) && (e \in E(G)) \\
&x_e \geq 0 && (e \in E(G))
\end{aligned}$$

Primjer 4. Grad Winnipeg kreće u izgradnju novih stabmenih zgrada. Za izgradnju im je potreban pijesak. Vancouver je najbliži grad iz kojeg se pijesak izvozi, stoga Winnipeg naručuje iz istoga. Pijesak se prevozi kamionima. U Winnipegu znaju da će im trebati više pijeska nego što Vancouver može dovesti u jednoj ruti sa svim svojim rasoloživim kamionima, stoga u prvoj rundi dovođenja pijeska naručuju maksimalnu količinu koju grad Vancouver može dovesti. Na slici 13 dana je shema problema. Grad Vancouver ima na raspolaganju $|E(G)| = 10$ raspoloživih kamiona (svaki kamion vozi drugu dionicu). Nijedan kamion ne vozi direktno iz Vancouvera u Winnipeg, već se pijesak prenosi iz kamiona u kamion dok ne stigne na odredište. Kamion koji vozi iz Vancouvera u Edmonton ima kapacitet od 16 tona pijeska, iz Edmontona u Saskatoon 12 tona, itd. Sav pijesak koji krene iz Vancouvera mora stići u Winnipeg. Vancouver, dakle, maksimizira svoj izvoz u Winnipeg. Ovaj primjer kasnije ćemo riješiti primjenom Ford - Fulkerson algoritma kojeg ćemo u poglavlju 4.1.1 detaljno obraditi.



Slika 13: primjer 4. [2]

Definicija 6. **(s,t) - rez u G** je skup $X \subseteq V(G)$ takav da vrijedi $s \in X$ i $t \in V(G)\backslash X$. **Veličina (s,t) - reza** definirana je sa $\sum_{e\in\delta^+(X)} u(e)$. **Minimalni (s,t) - rez** je (s,t) - rez minimalne veličine (obzirom na u) u G.

Lema 3. Za svaki (s,t) - rez A i za svaki (s,t) - tok f vrijedi:

(a) $|f| = \sum_{e\in\delta^+(A)} f(e) - \sum_{e\in\delta^-(A)} f(e)$,

(b) $|f| \leq \sum_{e\in\delta^+(A)} u(e)$.

Dokaz.

$$\begin{aligned} |f| &= \sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) \\ &= \sum_{v\in A} \Big(\sum_{e\in\delta^+(v)} f(e) - \sum_{e\in\delta^-(v)} f(e) \Big) \\ &= \sum_{e\in\delta^+(A)} f(e) - \sum_{e\in\delta^-(A)} f(e). \end{aligned}$$

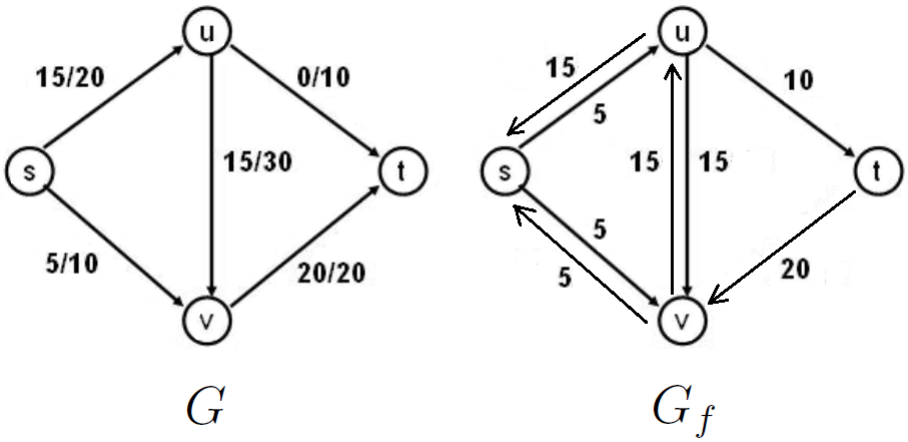
Tvrdnja (b) slijedi iz (a) korištenjem $0 \leq f(e) \leq u(e)$ za sve $e \in E(G)$.

■

Tvrdnja (b) govori kako maksimalna vrijednost (s,t) - toka nemože biti veća od veličine minimalnog (s,t) - reza. U razvoju poglavlja dokazat ćemo kako u toj tvrdnji vrijedi jednakost.

Definicija 7. Za usmjeren graf G definiramo $\overleftrightarrow{\mathbf{G}} := (V(G), E(G) \cup \{\overleftarrow{e} \mid e \in E(G)\})$, gdje je $\overleftarrow{e}$ definiran sa $\overleftarrow{e} := (w,v)$ za $e = (v,w) \in E(G)$. Brid $\overleftarrow{e} := (w,v)$ zovemo **suprotni brid** od e. Primijetimo kako ćemo za $e = (v,w)$ i $e' = (w,v)$ u $\overleftrightarrow{G}$ imati paralelne bridove $\overleftarrow{e}$ i e'. Neka je $u : E(G) \to \mathbb{R}_+$ težinska funkcija grafa G. Tada su težine novodobivenih bridova u $\overleftrightarrow{G}$ definirane sa $u(\overleftarrow{e}) := u(e)$.

Definicija 8. Neka je G usmjeren graf, neka je $u : E(G) \to \mathbb{R}_+$ njegova težinska funkcija i neka je f tok u (G,u). **Rezidualna težina** je funkcija $u_f(e) : E(\overleftrightarrow{G}) \to \mathbb{R}_+$ definirana sa $u_f(e) := u(e) - f(e)$ za sve $e \in E(G)$ i $u_f(\overleftarrow{e}) := f(e)$ inače. **Rezidualni graf $\mathbf{G_f}$** je graf $(V(G), \{e \in E(\overleftrightarrow{G}) \mid u_f(e) > 0\})$. Težinska funkcija rezidualnog grafa G_f jest rezidualna težina u_f.



Slika 14: Primjer rezidualnog grafa G_f.

Napomena 4. Izraz x/y na svakome bridu $e \in E(G)$ označava vrijednost toka $f(e)$ i težinu $u(e)$. Izraz $0/y$ ponekad ćemo kraće zapisivati sa y.

Definicija 9. Neka je f tok i neka je P put (ili ciklus) u G_f. **Povećati f duž P za γ** znači za svaki $e \in E(P)$: ako je $e \in E(G)$ staviti $f(e) := f(e) + \gamma$, ako je $e = \overleftarrow{e_0}$ za neki $e_0 \in E(G)$ staviti $f(e_0) := f(e_0) - \gamma$. Za mrežu (G, u, s, t) i za (s, t) - tok f definiramo **f - rastući put** kao (s, t) - put rezidualnog grafa G_f.

4.1.1 Ford - Fulkerson[4] (FF) algoritam

Za zadanu mrežu (G, u, s, t) gdje je težinska funkcija oblika $u : E(G) \to \mathbb{Z}+$, FF algoritam daje (s, t) - tok maksimalne vrijednosti. Algoritam se provodi kroz sljedeća tri koraka:

① Stvaljamo $f(e) := 0$, za sve $e \in E(G)$,

② Tražimo f - rastući put P
{ako takav ne postoji, prekidamo algoritam},

③ Stavljamo $\gamma := \min_{e \in E(P)} u_f(e)$. Povećamo f duž P za γ i vraćamo se na ②.

FF] algoritam uvijek ima trivijalno rješenje $f \equiv 0$. Kako su težine bridova brojevi isključivo u $\mathbb{Z}_+$, lako se vidi da pri svakoj iteraciji koraka ③ vrijedi $\gamma \in \mathbb{Z}_+$. Zaključujemo kako algoritam uvijek završava nakon konačno mnogo iteracija. Algoritam općenito nije ispravan za težinske funkcije s vrijednostima izvan tog skupa.

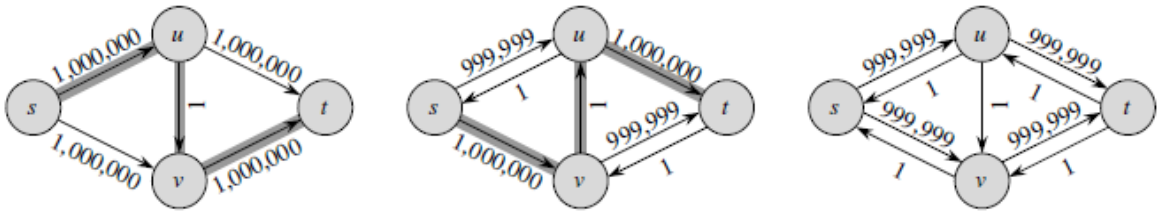
[4]Lester Randolph Ford Jr. i Delbert Ray Fulkerson, američki matematičari.

Bridovi u kojima je u koraku ③ postignut minimum zovu se bridovi uskog prolaza. Način na koji je γ izabran osigurava da f zadrži svojstva toka (vrijednosti toka u bridovima ne mogu nadmašiti težine istih). Kako je P (s,t) - put, lako se vidi da je tečnost očuvana u svim vrhovima osim u s i t.

Traženje rastućeg puta u koraku ② nije uvjetovano ničime, stoga je potrebno pronaći bilo koji (s,t) - put u G_f. Primijetimo kako algoritam daje točno jedan tok kao traženo rješenje. U slučaju kada postoji više (s,t) - tokova maksimalne vrijednosti, rješenje će ovisiti o redoslijedu biranja rastućih puteva. Iako sa izborom rastućeg puta ne možemo pogriješiti, ponekad je korisno razmisliti koji put odabrati. Naravno, riječ je o brzini izvođenja algoritma koju možemo uvelike ubrzati biramo li rastuće puteve na pametan način.

Promotrimo sliku 15.

Izbor rastućeg puta pri svakom izvođenju koraka ② temelji se na odabiru između rastućeg puta duljine 3 ($suvt$ ili $svut$) ili rastućeg puta duljine 2 (sut ili svt). Kada bismo naizmjenično svaki put izabrali put duljine 3, korak ② morali bi ponoviti $2,000,000$ puta. Međutim, lako se vidi da je problem rješiv u samo dvije iteracije algoritma birajući rastuće puteve duljine 2.



Slika 15: Primjer lošeg biranja rastućih puteva. [2]

Teorem 5. FF algoritam daje ispravne rezultate, odnosno nakon što algoritam stane, dobiveni f ima maksimalnu vrijednost.

Dokaz. Sljedeća tvrdnja implicira tvrdnju teorema stoga je dovoljno nju dokazati.
Tvrdnja: (s,t) - tok f ima maksimalnu vrijednost ako i samo ako ne postoji f - rastući put.

Neka je f (s,t) - tok maksimalne vrijednosti. Pretpostavimo suptotno, da postoji f - rastući put P. Tada u koraku ③ dobivamo tok veće vrijednosti od f, što je u kontradikciji sa maksimalnočću toka f.

Pokažimo sada obrat. Kako ne postoji f - rastući put, zaključujemo da t nije dostižan iz s u G_f. Neka je $R \subseteq V(G)$ skup vrhova dostižnih iz s u G_f. Tada za sve $e \in \delta_G^+(R)$ vrijedi $f(e) = u(e)$, jer bi u suprotnom postojao $v_0 \in V(G) \backslash R$ dostižan iz s. Istim razmišljanjem

dolazimo do jednakosti $f(e) = 0$ za sve $e \in \delta_G^-(R)$. Iz Leme 3.a) dobivamo:

$$|f| = \sum_{e \in \delta_G^+(R)} u(e).$$

pa prema Lemi 3.b) zaključujemo kako je f tok maksimalne vrijednosti.

■

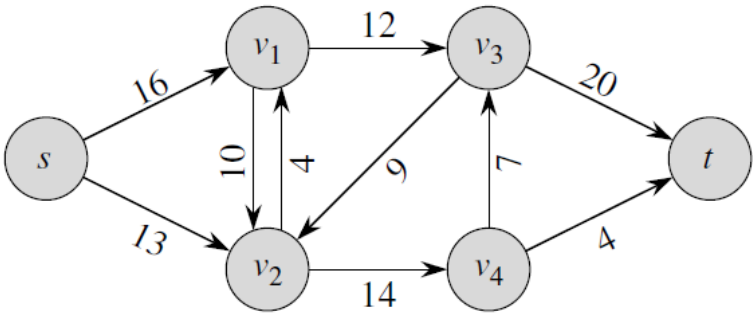
Kako je mreža (G, u, s, t) proizvoljna, zaključujemo da za svaki (s, t) - tok P maksimalne vrijednosti postoji (s, t) - rez čija je veličina jednaka vrijednosti od P (napravi se konstrukcija (s, t) - reza R na isti način kao i u Teoremu 5.). Ova tvrdnja zajedno sa Lemom 3.b) daje najvažniji teorem teorije mrežnih tokova, Max - tok - min - rez teorem.

Teorem 6. Max - tok - min - rez teorem $\big($Ford i Fulkerson [1956]$\big)$ U svakoj mreži maksimalna vrijednost (s, t) - toka jednaka je minimalnoj veličini (s, t) - reza.

■

Primjer FF algoritma (Primjer 4.)

Neka je zadan graf G sa slike 16. Kako funkcija u poprima vrijednosti isključivo u $\mathbb{Z}_+$ smijemo koristiti FF algoritam.



Slika 16: Graf G. [2]

Promotrimo sliku 17.

Na slici imamo pet redova grafova: (a), (b), (c), (d) i (e). Grafovi koji se nalaze u lijevome stupcu odgovaraju G_f grafovima u trenutnoj fazi algoritma (s trenutnim vrijednostima toka f). Grafovi iz desnog stupca odgovaraju grafu G u trenutnoj fazi algoritma.

U redu (a) dan je graf G_f za početni tok dobiven nakon koraka ①. Za takav f iz definicije grafa G_f jasno je da vrijedi $G_f = G$. Na grafu G_f pronašli smo (s,t) - put $P_1 = sv_1v_3v_2v_4t$. Dobivamo $\gamma := \min_{e \in E(P_1)} u_f(e) = u_f((v_4,t)) = 4$. Stavljamo $f(e) := f(e) + 4$ za sve $e \in E(P_1)$ i dobivamo graf G u redu (a).

Iz dobivenog grafa G dobivamo graf G_f u redu (b). Na grafu G_f pronašli smo (s,t) - put $P_2 = sv_1v_2v_4v_3t$. Dobivamo $\gamma := \min_{e \in E(P_2)} u_f(e) = u_f((v_4,v_3)) = 7$. Stavljamo $f(e) := f(e) + 7$ za sve $e \in E(P_2)$ i dobivamo graf G u redu (b).

Iz dobivenog grafa G dobivamo graf G_f u redu (c). Primijetimo kako smo težine paralelnih bridova (v_2,v_1) na grafu G_f radi jednostavnosti zbrojili. Na grafu G_f pronašli smo (s,t) - put $P_3 = sv_2v_1v_3t$. Dobivamo $\gamma := \min_{e \in E(P_3)} u_f(e) = u_f((v_1,v_3)) = 8$. Stavljamo $f(e) := f(e) + 8$ za sve $e \in E(P_3)$ i dobivamo graf G u redu (c).

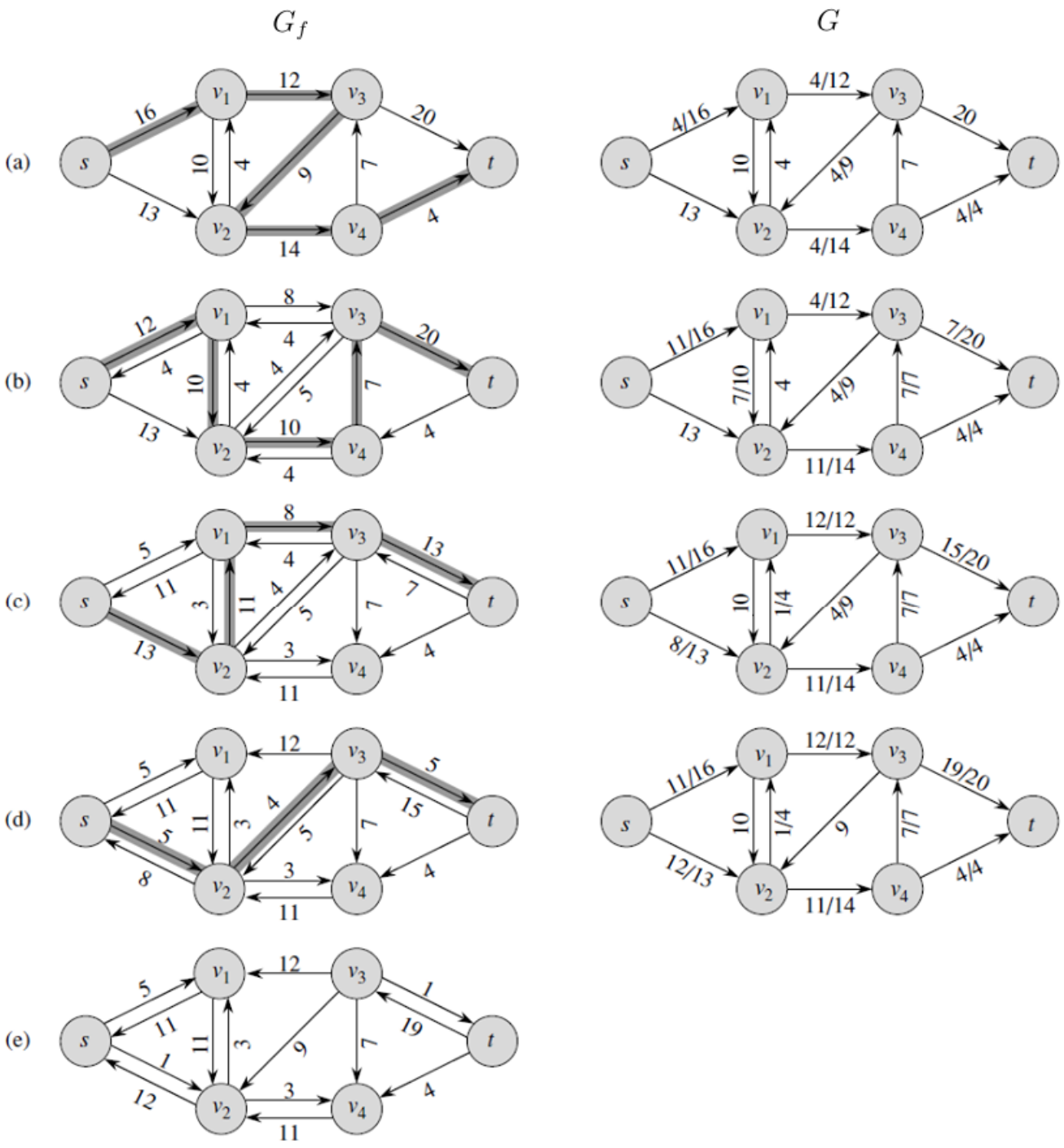
Iz dobivenog grafa G dobivamo graf G_f u redu (d). Na grafu G_f pronašli smo (s,t) - put $P_4 = sv_2v_3t$. Dobivamo $\gamma := \min_{e \in E(P_4)} u_f(e) = u_f((v_2,v_3)) = 4$. Stavljamo $f(e) := f(e) + 4$ za sve $e \in E(P_4)\backslash\{(v_2,v_3)\}$. Kako brid $(v_2,v_3) \notin E(G)$ već je u G_f dobiven kao suprotni brid od (v_3,v_2), stavljamo $f((v_3,v_2)) = f((v_3,v_2)) - 4$ i dobivamo graf G u redu (d).

Iz dobivenog grafa G dobivamo graf G_f u redu (e). Kako na grafu G_f ne postoji (s,t) - put, zaustavljamo algoritam. Dobiveni tok f iz slike grafa G u redu (d) jest tok maksimalne vrijednosti. Iz slike čitamo:

$$|f| = \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) = (11 + 12) - 0 = 23.$$

Slijedeći intuiciju nakon primjera na slici 15. u kojem je pametnije bilo birati rastuće puteve s manjim brojem bridova, dolazimo do Edmonds - Karp (EK)[5] algoritma za traženje (s,t) - toka maksimalne vrijednosti. EK algoritam zahtijeva poznavanje Breadth - first search (BFS) algoritma, stoga prvo upoznajmo isti.

[5] Jack R. Edmonds i Richard Manning Karp, američki informatičari.



Slika 17: FF algoritam. [2]

Breadth - first search (BFS) algoritam

Za zadani (usmjereni ili neusmjereni) graf G i za vrh $s \in V(G)$, BFS algoritam daje najkraće (s, v) - puteve za sve $v \in V(G)$ koji su dostižni iz s. Algoritam se provodi kroz sljedeća četiri koraka:

① Stavljamo:

- $l(v) := \begin{cases} 0, & \text{za } v = s, \\ \infty, & \text{inače.} \end{cases}$
- $R := \{s\}$ i $Q := \{s\}$.

② Ako je $Q = \emptyset$ prekidamo algoritam
{u suprotnom biramo $v \in Q$ koji je u skup Q ušao prije svih ostalih vrhova koji se trenutno nalaze u Q},

③ Biramo $w \in V(G) \backslash R$ za koji vrijedi $e = (v, w) \in E(G)$
{ako takav w ne postoji stavljamo $Q := Q \backslash \{v\}$ i vraćamo se na ②},

④ Stavljamo $R := R \cup \{w\}$, $Q := Q \cup \{w\}$, $l(w) := l(v) + 1$ i vraćamo se na ②.

Napomena 5. Kako na grafu nemamo definiranu težinsku funkciju, broj $dist_G(s, v)$ odgovara broju bridova najkraćeg (s, v) - puta u G.

Teorem 7. BFS algoritam daje ispravne rezultate, tj. nakon sprovedenog algoritma vrijedi $l(v) = \text{dist}_G(s, v)$, za sve $v \in V(G)$.

Dokaz. Pretpostavimo suprotno, neka je $w \in V(G)$ takav da nakon sprovedenog algoritma vrijedi $l(w) \neq \text{dist}_G(s, w)$. Iz algoritma je jasno kako slučaj $l(w) < \text{dist}_G(s, w)$ nije moguć. Dakle, pretpostavimo $l(w) > \text{dist}_G(s, w)$ i neka za vrh w vrijedi

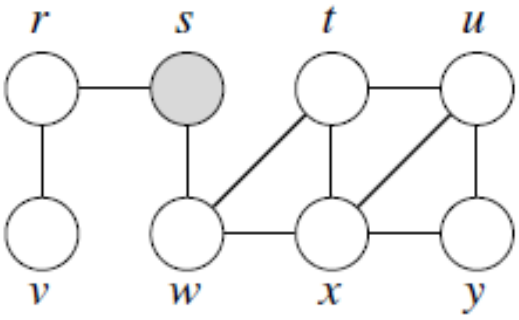
$$\text{dist}_G(s, w) = \min \{\text{dist}_G(s, x) \mid x \in V(G) \text{ t.d. } l(x) > \text{dist}_G(s, x)\}.$$

Neka je P najkraći (s, w) - put u G i neka je $e = (v, w)$ posljednji brid u P. Tada vrijedi $l(v) = \text{dist}_G(s, v)$. Kako je $l(w) > \text{dist}_G(s, w) = \text{dist}_G(s, v) + 1 = l(v) + 1$, iz ③ i ④ zaključujemo $e \notin E(G)$, što je u kontradikciji sa egzistencijom puta P.

■

Primjer BFS algoritma

Neka je zadan neusmjereni graf G sa slike 18.

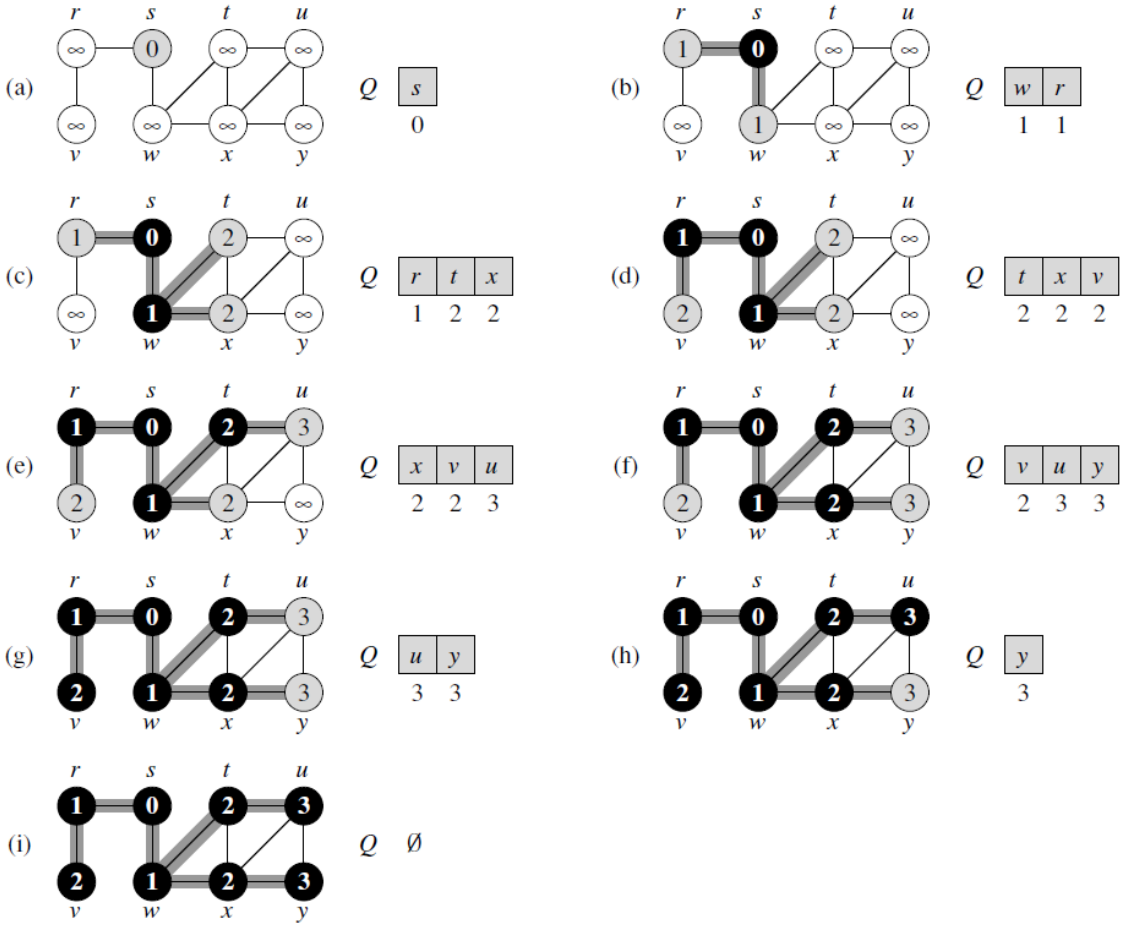


Slika 18: Graf G. [2]

Promotrimo sliku 19.

U podslici (a) izvršen je korak ①. Stavljamo $R := \{s\}$ i $Q := \{s\}$. Biramo $v \in Q$ koji je u skup Q ušao prije svih ostalih vrhova. Kako je $Q = \{s\}$, stavljamo $v = s$. Kako su w i r susjedi od s, u (b) stavljamo $R := \{s, w, r\}$, $Q := \{w, r\}$, $l(w) = l(r) := 1$ i vraćamo se na ②. Biramo $v \in Q$ koji je u skup Q ušao prije svih ostalih vrhova, pa stavljamo $v = w$. Kako su t i x susjedi od w, u (c) stavljamo $R := \{s, w, r, t, x\}$, $Q := \{r, t, x\}$, $l(t) = l(x) := 2$ i vraćamo se na ②.

Stavljamo $v = r$. Kako je v susjed od r, u (d) stavljamo $R := \{s, w, r, t, x, v\}$, $Q := \{t, x, v\}$, $l(v) := 2$ i vraćamo se na ②. Stavljamo $v = t$. Kako je u jedini susjedi od t koji još uvjek nije u skupu R, u (e) stavljamo $R := \{s, w, r, t, x, v, u\}$, $Q := \{x, v, u\}$, $l(u) := 3$ i vraćamo se na ②. Kako je y jedini susjed od x koji još uvjek nije u skupu R, u (f) stavljamo $R := \{s, w, r, t, x, v, u, y\}$, $Q := \{v, u, y\}$, $l(y) := 3$ i vraćamo se na ②. Kako vrh v nema nijednog susjeda koji već nije u skupu R, u (g) stavljamo $Q := \{u, y\}$ i vraćamo se na ②. Kako vrh u nema nijednog susjeda koji već nije u skupu R, u (h) stavljamo $Q := \{y\}$ i vraćamo se na ②. Kako vrh y nema nijednog susjeda koji već nije u skupu R, u (i) stavljamo $Q := \emptyset$ i vraćamo se na ②. Kako je $Q = \emptyset$ prekidamo algoritam.



Slika 19: BFS algoritam. [2]

4.1.2 Edmonds - Karp (EK) algoritam

Za zadanu mrežu (G, u, s, t) gdje je težinska funkcija oblika $u : E(G) \to \mathbb{Z}+$, EK algoritam daje (s, t) - tok maksimalne vrijednosti. Algoritam se provodi kroz sljedeća tri koraka:

① Stavljamo $f(e) := 0$, za sve $e \in E(G)$,

② Tražimo najkraći f - rastući put P
{ako takav ne postoji, prekidamo algoritam},

③ Stavljamo $\gamma := \min_{e \in E(P)} u_f(e)$. Povećamo f duž P za γ i vraćamo se na ②.

Riječ 'najkraći' u koraku ② odnosi se na traženje (s,t) - puta u G_f s minimalnim brojem bridova koristeći BFS algoritam. Dakle, ako algoritmom dobijemo $l(t) = m \in \mathbb{N}$, tražimo postoji li (s,t) - put P s m bridova. Ako takav ne postoji, tražimo postoji li put s $(m+1)$-nim bridom, itd.

Edmonds i Karp objavili su algoritam 1972. godine. EK algoritam prvi je pronađeni algoritam polinomne složenosti za pronalazak toka maksimalne vrijednosti. Njegova složenost jest $O(m^2n)$. U nastavku dajemo Propoziciju 2. koju ćemo koristiti u dokazu složenosti algoritma. Dokaz propozicije tehnički je veoma zahtjevan stoga ju dajemo bez dokaza.

Propozicija 2. Neka je $f_1, f_2, ...$ niz tokova takav da je f_{i+1} dobiven iz f_i povećanjem duž P_i, gdje je P_i najkraći f_i - rastući put. Tada je:

(a) $|E(P_k)| \leq |E(P_{k+1})|$ za svaki k,

(b) $|E(P_k)| + 2 \leq |E(P_l)|$ za sve $k < l$ takve da $P_k \cup P_l$ sadrži par suprotnih bridova.

■

Teorem 8. (Edmonds i Karp [1972]) EK algoritam završava nakon najviše $\frac{mn}{2}$ iteracija, gdje su $m = |E(G)|$, $n = |V(G)|$.

Dokaz. Neka je $P_1, P_2, ...$ niz rastućih puteva iz koraka ②. Znamo da u svakom rastućem putu postoji barem jedan brid uskog prolaza. Neka je $P_{i_1}, P_{i_2}, ...$ podniz niza rastućih puteva u kojima je $e \in E(G)$ brid uskog prolaza. Prema svojstvu brida uskog prolaza znamo da između svakog para P_{i_j} i $P_{i_{j+1}}$ postoji rastući put P_k $(i_j < k < i_{j+1})$ koji sadrži brid $\overleftarrow{e}$. Prema Propoziciji 2.(b) imamo $|E(P_{i_j})| + 4 \leq |E(P_k)| + 2 \leq |E(P_{i_{j+1}})|$, za svaki j. Iz $|E(P_{i_j})| + 4 \leq |E(P_{i_{j+1}})|$ i $|E(P_{i_{j+1}})| \leq n - 1$ zaključujemo da postoji najviše $\frac{n}{4}$ rastućih puteva u kojima je e brid uskog prolaza. Kako u svakom rastućem putu postoji barem jedan brid uskog prolaza iz $\overleftrightarrow{G}$, zaključujemo da postoji najviše $|E(\overleftrightarrow{G})| \cdot \frac{n}{4} \leq \frac{mn}{2}$ rastućih puteva.

■

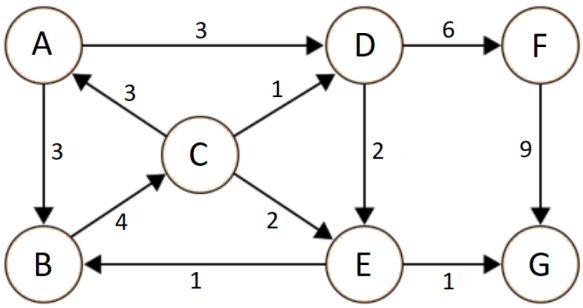
Korolar 2. Složenost EK algoritma jest $O(m^2n)$.

Dokaz. U Teoremu 8. dokazali smo da tokom izvođenja EK algoritma postoji najviše $\frac{mn}{2}$ iteracija. Izbor svakog rastućeg puta zahtijeva izvođenje BFS algoritma čija je složenost $O(m)$ (BFS algoritam izvodi se putem bridova grafa), čime je tvrdnja dokazana.

■

Primjer EK algoritma

Neka je zadan graf G sa slike 20. Neka je vrh A izvor, te neka je vrh G ponor.



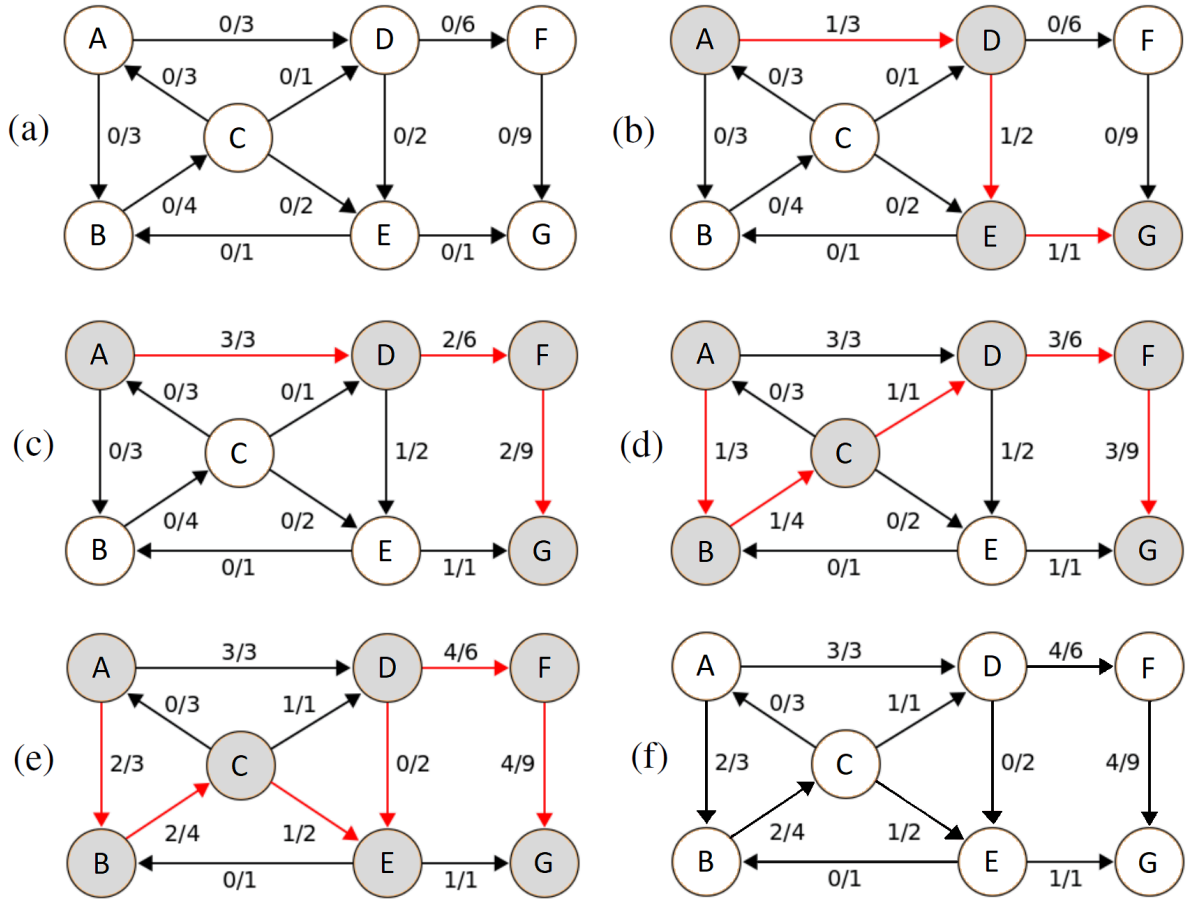
Slika 20: Graf G. [3]

Promotrimo sliku 21.

Primjenom BFS algoritma na rezidualnom grafu grafa (a) dobili smo da je najkraći (A, G) - put duljine 3 brida. Na grafu pronađemo (A, G) - put $P_1 = ADEG$ duljine 3. Dobivamo $\gamma := \min_{e \in E(P_1)} u_f(e) = u_f((E, G)) = 1$. Na slici (b) povećali smo f duž P_1 za $\gamma = 1$. Primjenom BFS algoritma na rezidualnom grafu grafa (b) dobili smo da je najkraći (A, G) - put duljine 3 brida. Na grafu pronađemo (A, G) - put $P_2 = ADFG$ duljine 3. Dobivamo $\gamma := \min_{e \in E(P_2)} u_f(e) = u_f((A, D)) = 2$. Na slici (c) povećali smo f duž P_2 za $\gamma = 2$.

Primjenom BFS algoritma na rezidualnom grafu grafa (c) dobili smo da je najkraći (A, G) - put duljine 5 brida. Na grafu pronađemo (A, G) - put $P_3 = ABCDFG$ duljine 5. Dobivamo $\gamma := \min_{e \in E(P_3)} u_f(e) = u_f((C, D)) = 1$. Na slici (d) povećali smo f duž P_3 za $\gamma = 1$. Primjenom BFS algoritma na rezidualnom grafu grafa (d) dobili smo da je najkraći (A, G) - put duljine 6 brida. Na grafu pronađemo (A, G) - put $P_4 = ABCEDFG$ duljine 6. Dobivamo $\gamma := \min_{e \in E(P_4)} u_f(e) = u_f((E, D)) = 1$. Na slici (e) povećali smo f duž P_4 za $\gamma = 1$. Kako na rezidualnom grafu grafa na slici (e) ne postoji (A, G) - put, prekidamo algoritam. Tok prikazan na slici (f) traženi je (A, G) - tok maksimalne vrijednosti.

Napomena 6. Primijetimo kako je niz duljina dobivenih rastućih puteva nepadajući.



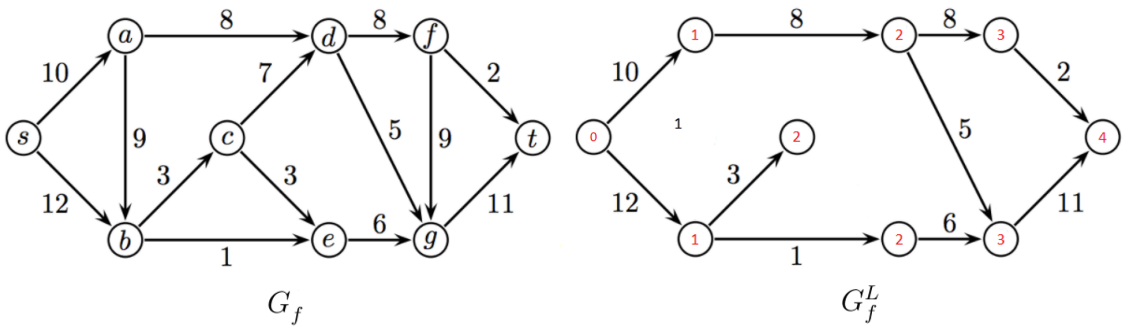
Slika 21: EK algoritam. [3]

4.1.3 Dinic[6] (Dinitz) algoritam

Definicija 10. Neka je (G, u, s, t) mreža i neka je f (s,t) - tok. **Nivo graf** $\mathbf{G_f^L}$[7] grafa G_f definiramo kao graf $(V(G), \{e = (x,y) \in E(G_f) \mid \text{dist}_{G_f}(s,x) + 1 = \text{dist}_{G_f}(s,y)\})$.

Primijetimo kako je nivo graf uvijek acikličan. Nivo graf može se konstruirati u $O(m)$ vremenu koristeći BFS algoritam. Iz Propozicije 2.(a) vidimo da je niz duljina rastućih puteva dobivenih primjenom EK algotitma nepadajući. Niz rastućih puteva iste duljine zovemo **fazom algotitma**.

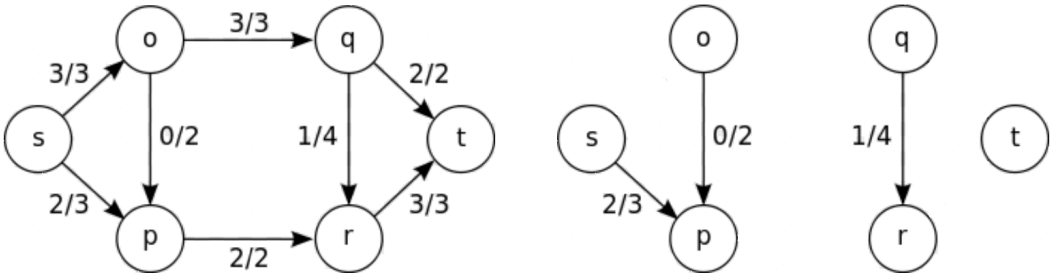
[6] Yefim Dinitz, židovski informatičar.
[7] Oznaka G_f^L dolazi iz engleskog naziva nivo grafa - "level graph".



Slika 22: Primjer nivo grafa G_f^L.

Neka je f tok na početku proizvoljne faze algoritma. Tada iz Propozicije 2.(b) vidimo da se svi rastući putevi ove faze algoritma nužno nalaze na nivo grafu G_f^L jer bi u suprotnom duljine rastućih puteva unutar promatrane faze algoritma bile različite.

Definicija 11. Neka je (G, u, s, t) mreža. Za (s, t) - tok f kažemo da je **blokirajući** ako u grafu $(V(G), \{e \in E(G) \mid f(e) < u(e)\})$ ne postoji (s, t) - put.



Slika 23: Primjer blokirajućeg toka f.

Za zadanu mrežu (G, u, s, t) gdje je težinska funkcija oblika $u : E(G) \to \mathbb{Z}_+$, Dinic algoritam daje (s, t) - tok maksimalne vrijednosti. Algoritam se provodi kroz sljedeća tri koraka:

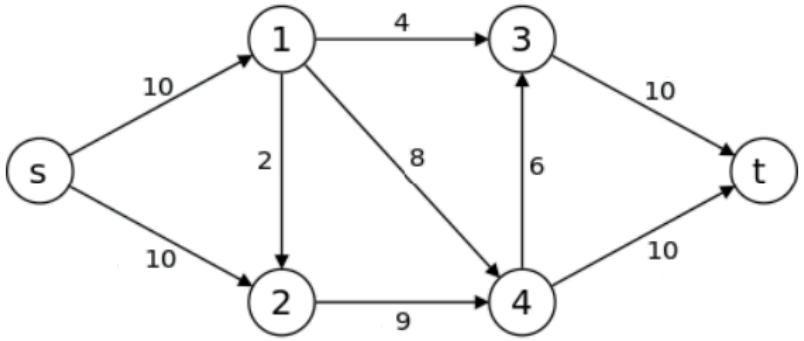
① Stavljamo $f(e) := 0$ za sve $e \in E(G)$,

② Tražimo blokirajući (s, t) - tok f' u G_f^L
{ako je $|f'| = 0$, prekidamo algoritam},

③ Povećavamo f za odgovarajuće vrijednosti pronađenog toka f' i vraćamo se na ②.

Primjer Dinic algoritma

Neka je zadan graf G sa slike 24.



Slika 24: Graf G. [4]

Promotrimo sliku 25. i sliku 26.

Na grafu G(a) stavili smo $f(e) := 0$ za sve $e \in E(G)$ čime je korak ① gotov. Iz grafa G(a) dobivamo redom grafove G_f(a) te G_f^L(a). Tražimo blokirajući (s,t) - tok f' u G_f^L(a).

Na slici G_f^L(a) vidimo rješenje traženog toka. Postupak traženja blokirajućeg toka ponekad je veoma težak. Postoje algoritmi pomoću kojih se isti može pronaći. Međutim, ti algoritmi tehnički su veoma zahtjevni stoga ih u ovome radu nećemo obrađivati. Ukoliko graf nije suviše kompleksan, postoji mnogo principa pomoću kojih se blokirajući tok može tražiti te pronaći. Objasnimo sada princip koji smo u ovome radu primijenili te pronašli traženi tok.

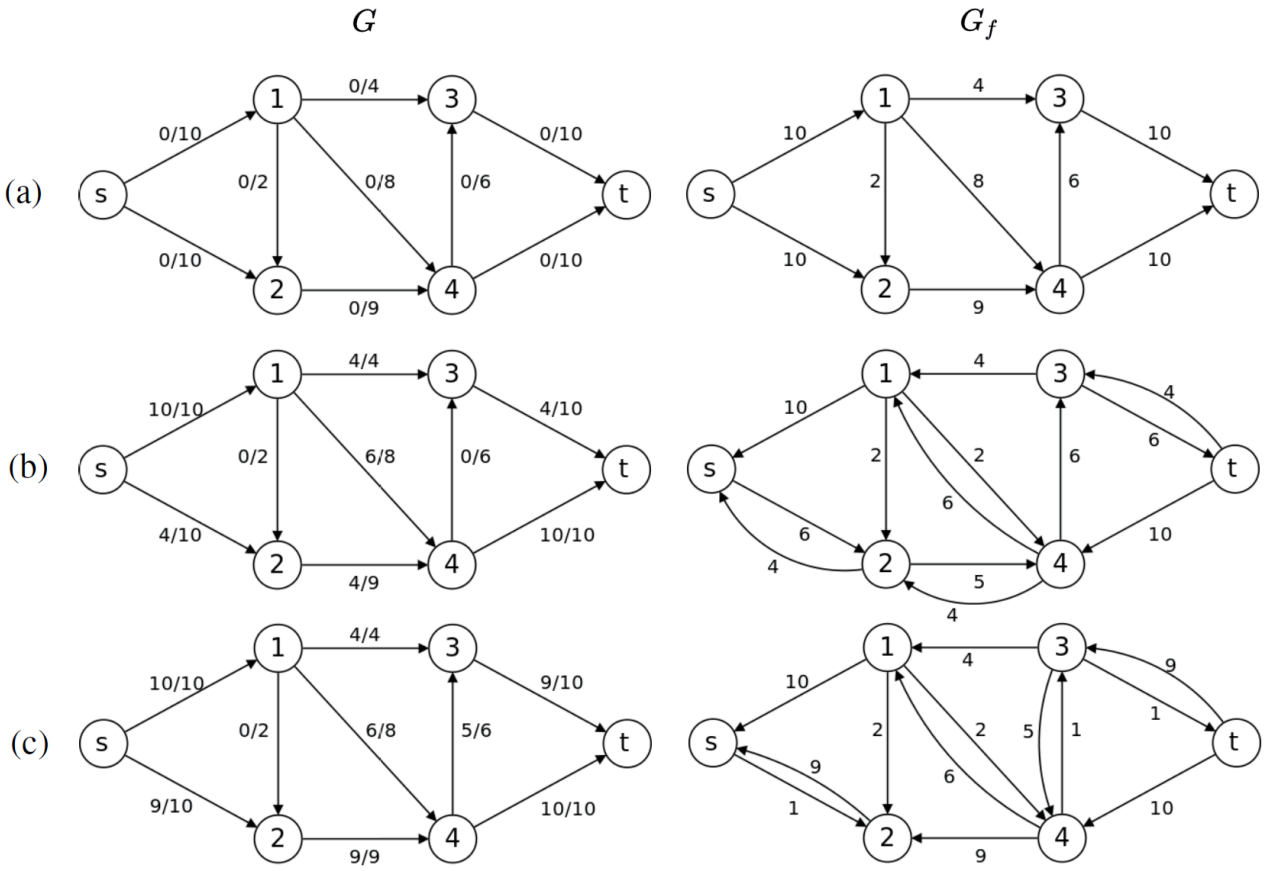
Iz definicije blokirajućeg toka vidimo kako u grafu $G^* = (V(G), \{e \in E(G) \mid f(e) < u(e)\})$ nestaju bridovi $e \in E(G)$ za koje vrijedi $f(e) = u(e)$. Kako želimo da vrh t prestane biti dostižan iz vrha s, stavljamo $f((4,t)) = 10$ čime smo brid $(4,t)$ uklonili iz grafa G^*. Važno je primijetiti kako težine $u((1,4)) = 8$ i $u((2,4)) = 9$ zbog nejednakosti $8 + 9 = 17 > 10$ podržavaju postupak kojim želimo ukloniti brid $(4,t)$. Analognim razmišljanjem brzo uočavamo kako nejednakosti $u((s,1)) = 10 > 8 = u((1,4))$ i $u((s,2)) = 10 > 9 = u((2,4))$ također podržavaju postupak koji izvršavamo. Preostaje odlučiti na koji način raspodjeliti iznos od 10 jedinica između $f((1,4))$ i $f((2,4))$.

Kako je $u((3,t)) = 10$ i $u((1,3)) = 4$, zaključujemo kako brid $(3,t)$ nije moguće izbrisati iz grafa G^*. Međutim, kako je $u((1,3)) = 4$ i $u((s,1)) = 10$, zaključujemo kako je brid $(1,3)$ moguće izbrisati iz grafa G^*. Važno je primijetiti kako će vrh t prestati biti dostižan iz vrha s ukoliko uspijemo izbrisati bridove $(1,3)$ i $(4,t)$ iz grafa G^*. Dakle, stavljamo $f((s,1)) = f((1,3)) = f((3,t)) = 4$. Kako je sada $u((s,1)) - f((s,1)) = 10 - 4 = 6$,

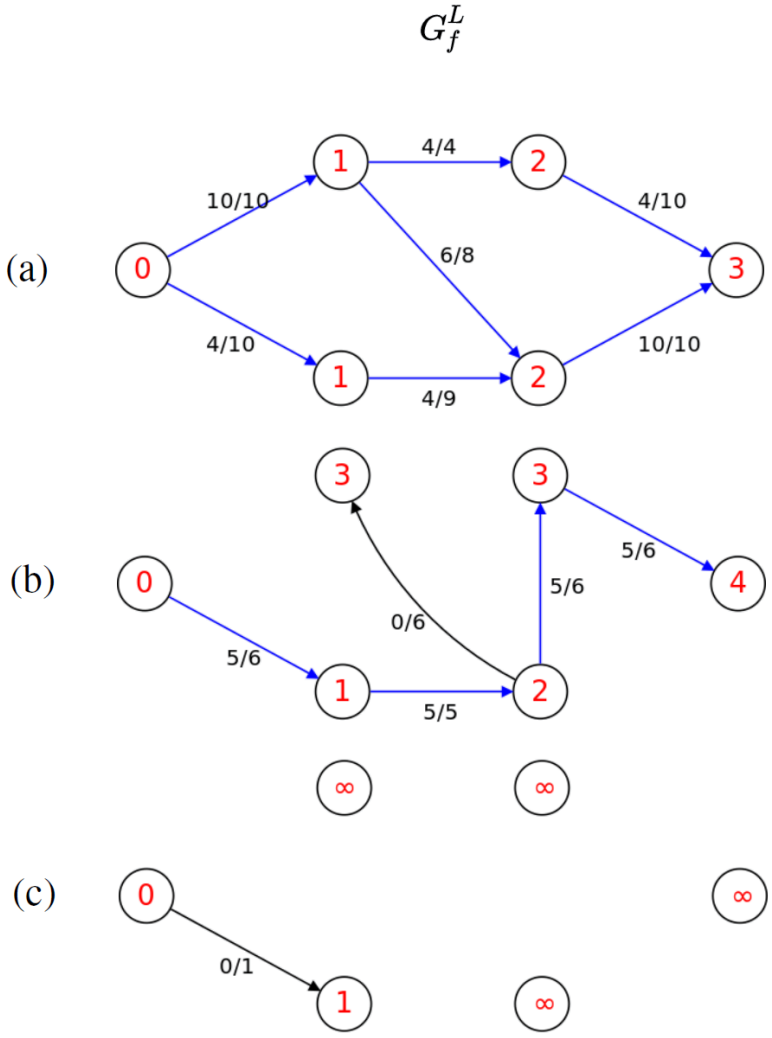
stavljamo $f((s,1)) = 4 + 6 = 10$ i $f((1,4)) = 6$. Preostaje još staviti $f((s,2)) = 10 - 6 = 4$ i $f((2,4)) = 10 - 6 = 4$, čime smo dobili traženi blokirajući tok f'.

Kako je $|f'| = 14 > 0$, stavljamo $f(e) := f(e) + f'(e)$ za sve $e \in E(G)$ i dobivamo graf G(b). Iz grafa G(b) dobivamo redom grafove G_f(b) te G_f^L(b). Tražimo blokirajući (s,t) - tok f' u G_f^L(b). Kako je sada vrh t dostižan iz vrha s preko jedinstvenog puta $P = s243t$, dovoljno je pronaći $u^* = \min_{e \in P} u(e) = u((2,4)) = 5$ te povećati f' duž P za $u^* = 5$.

Kako je $|f'| = 5 > 0$, stavljamo $f(e) := f(e) + f'(e)$ za sve $e \in E(G)$ i dobivamo graf G(c). Iz grafa G(c) dobivamo redom grafove G_f(c) te G_f^L(c). Kako vrh t nije dostižan iz vrha s u G_f^L(c), zaključujemo kako je $|f'| = 0$ i prekidamo algoritam. Traženo rješenje jest graf G(c).



Slika 25: Dinic algoritam (1/2). [4]



Slika 26: Dinic algoritam (2/2). [4]

4.1.4 Goldberg - Tarjan[8] (GT) algoritam

Za zadanu mrežu (G, u, s, t) gdje je težinska funkcija oblika $u : E(G) \to \mathbb{Z}_+$, GT algoritam daje (s, t) - tok maksimalne vrijednosti. Algoritam koristi **Push - Relabel operacije** koje ćemo također objasniti u ovome poglavlju. Prema tvrdnji dokazanoj u Teoremu 5., (s, t) - tok f ima maksimalnu vrijednost ako i samo ako vrijede sljedeća dva uvjeta:

① $\text{ex}_f(v) = 0$ za sve $v \in V(G)\backslash\{s, t\}$,

② Ne postoji f - rastući put.

U prethodno obrađenim FF i EK algoritmima, uvjet ① bio je zadovoljen neprestance, neovisno o fazi algoritma u kojoj smo se nalazili. Algoritmi su završavali u trenutku kada bi uvjet ② bio zadovoljen. GT algoritam funkcionira na obrnutom principu. Uvjet ② zadovoljen je neprestance, a algoritam završava u trenutku kada uvjet ① postane zadovoljen. Kako funkcija f tokom obrade algoritma neće zadovoljavati svojstvo toka, uvodimo najprije pojam (s, t) - protoka.

Definicija 12. Neka je (G, u, s, t) mreža. **(s, t) - protok** jest funkcija $f : E(G) \to \mathbb{R}_+$ koja zadovoljava $f(e) \leq u(e)$ za sve $e \in E(G)$ i $\text{ex}_f(v) \geq 0$ za sve $v \in V(G)\backslash\{s\}$. Za vrh $v \in V(G)\backslash\{s, t\}$ kažemo da je **aktivan** ako vrijedi $\text{ex}_f(v) > 0$.

Definicija 13. Neka je (G, u, s, t) mreža i neka je f (s, t) - protok. **Označavajuća udaljenost** jest funkcija $\psi : V(G) \to \mathbb{Z}_+$ za koju vrijedi $\psi(t) = 0$, $\psi(s) = n$ i $\psi(v) \leq \psi(w) + 1$ za sve $(v, w) \in E(G)$, gdje je $n = |V(G)|$. Za brid $e = (v, w) \in E(\overleftrightarrow{G})$ kažemo da je **dopustiv** ako je $e \in E(G_f)$ i ako vrijedi $\psi(v) = \psi(w) + 1$.

Primijetimo kako je svaki (s, t) - protok ujedno i (s, t) - tok ako i samo ako ne postoji nijedan aktivan vrh. Ako je ψ označavajuća udaljenost i ako je vrh t dostižan iz vrha v u G_f, lako se vidi da je vrijednost $\psi(v)$ donja granica broja bridova najkraćeg (v, t) - puta u G_f. GT algoritam se provodi kroz sljedeća tri koraka:

① Stavljamo:

- $f(e) := \begin{cases} u(e), & \text{za sve } \ e \in \delta^+(s), \\ 0, & \text{inače.} \end{cases}$

② Stavljamo:

- $\psi(v) := \begin{cases} n, & \text{za } \ v = s, \\ 0, & \text{inače.} \end{cases}$

[8] Andrew Vladislav Goldberg i Robert Endre Tarjan, američki informatičari.

③ Ako postoji aktivan vrh, tada:
{pronađemo aktivan vrh v,
ako ne postoji dopustiv brid $e \in \delta^+_{G_f}(v)$, tada Relabel(v),
inače pronađemo dupustiv brid $e \in \delta^+_{G_f}(v)$ i tada Push(e).},
u surotnome, prekidamo algoritam.

Push(e):
① Stavljamo $\gamma := \min\{\text{ex}_f(v), u_f(e)\}$,
② Povećavamo f duž e za γ.

Relabel(v):
① Stavljamo $\psi(v) := \min\{\psi(w) + 1 \mid e = (v, w) \in E(G_f)\}$.

Propozicija 3. Funkcije f i ψ zadržavaju svojstva (s, t) - protoka i označavajuće udaljenosti respektivno, tokom cijele obrade GT algoritma.

Dokaz. Lako se vidi da je funkcija f definirana korakom ① (s, t) - protok te da operacija Push čuva svojstva protoka. Kako operacija Relabel ne mijenja funkciju f, zaključujemo istinitost prve tvrdnje propozicije.

Također se lako vidi da je funkcija ψ definirana korakom ② označavajuća udaljenost te da operacija Relabel čuva svojstva označavajuće udaljenosti. Preostaje pokazati da operacija Push čuva svojstva označavajuće udaljenosti obzirom na novodobiveni (s, t) - protok f. Dakle, moramo provjeriti da za sve novodobivene bridove $(a, b) \in G_f$ vrijedi $\psi(a) \leq \psi(b) + 1$. Međutim, primjenom operacije Push(e) za neki $e = (v, w)$, jedini mogući novonastali brid jest $\overleftarrow{e} = (w, v)$, pa kako znamo da je brid e dopustiv (u suprotnom operacija Push(e) nebi bila moguća) zaključujemo da vrijedi $\psi(w) = \psi(v) - 1$.

■

Lema 4. Neka je f (s, t) - protok i neka je ψ označavajuća udaljenost obzirom na f. Vrijedi:
(a) Vrh s dostižan je iz svakog aktivnog vrha u G_f,
(b) Ako je vrh w dostižan iz vrha v u G_f, tada vrijedi $\psi(v) \leq \psi(w) + n - 1$,
(c) Vrh t nije dostižan iz vrha s u G_f.

Dokaz. (a): Neka je v aktivan vrh i neka je R skup vrhova dostižnih iz v u G_f. Lako se vidi da tada za svaki $e \in \delta^-_G(R)$ vrijedi $f(e) = 0$ (u suprotnom, postojao bi vrh izvan skupa R dostižan iz v). Slijedi:

$$\sum_{w \in R} \text{ex}_f(w) = \sum_{e \in \delta^-_G(R)} f(e) - \sum_{e \in \delta^+_G(R)} f(e) \leq 0.$$

Kako je v aktivan, iz $\text{ex}_f(v) > 0$ zaključujemo kako mora postojati vrh $w \in R$ t.d. vrijedi $\text{ex}_f(w) < 0$. Kako je f (s,t) - protok taj vrh mora biti s. Dakle, $s \in R$.

Tvrdnja (b) slijedi direktno iz $|V(G)| = n$ te iz svojstva funkcije označavajuće udaljenosti. Tvrdnja (c) slijedi iz (b) te iz $\psi(s) = n$, $\psi(t) = 0$ (lako se vidi da navedene jednakosti vrijede tokom cijele obrade algoritma).

■

Teorem 9. GT algoritam daje ispravne rezultate, tj. nakon sprovedenog algoritma dobiveni f jest (s,t) - tok maksimalne vrijednosti.

Dokaz. Kako pri završetku algoritma ne postoji aktivan vrh, iz definicija (s,t) - toka i (s,t) - protoka te iz dokazane tvrdnje da je f (s,t) - protok tokom cijele obrade algoritma direktno zaključujemo da je f (s,t) - tok. Kako je svaki (s,t) - tok ujedno i (s,t) - protok, iz Leme 4.(c) zaključujemo da ne postoji f - rastući put. Tada iz tvrdnje dokazane u Teoremu 5. slijedi dokaz teorema.

■

Lema 5.
(a) Vrijednost $\psi(v)$ strogo raste primjenom operacije Relabel(v) te se tokom obrade algoritma ista nikad ne smanjuje, za svaki $v \in V(G)$,
(b) Nejednakost $\psi(v) \leq 2n - 1$ vrijedi u svakoj fazi algoritma, za svaki $v \in V(G)$,
(c) $2n - 1$ jest maksimalan broj izvedene Relabel(v) operacije, za svaki fiksni $v \in V(G)$. Ukupan broj izvedenih Relabel operacija tokom obrade algoritma jest najviše $2n^2 - n$.

Dokaz. (a): Operacija Relabel(v) strogo povećava vrijednost $\psi(v)$ zbog toga što je ψ označavajuća udaljenost tokom cijele obrade algoritma te iz činjenice da brid $e \in \delta_G^+(v)$ nije mogao biti dopustiv prije same operacije Relabel(v) (u suprotnome bila bi pokrenuta operacija Push(e)). Kako se oparacijom Push vrijednost $\psi(v)$ ne mijenja, tvrdnja je dokazana.

(b): Primijetimo kako se vrijednost $\psi(v)$ može promijeniti jedino u slučaju kada je vrh v aktivan pa iz Leme 4.(a) i Leme 4.(b) imamo $\psi(v) \leq \psi(s) + n - 1 = 2n - 1$. Tvrdnja (c) slijedi iz (a) i (b) te iz $|V(G)| = n$.

■

Definicija 15. Za izvedenu operaciju Push(e) kažemo da je **zasićena** ako nakon njene izvedbe vrijedi $u_f(e) = 0$. U suprotnom kažemo da je operacija Push(e) **nezasićena**.

Lema 6. Ukupan broj zasićenih Push operacija tokom obrade algoritma jest najviše $2mn$.

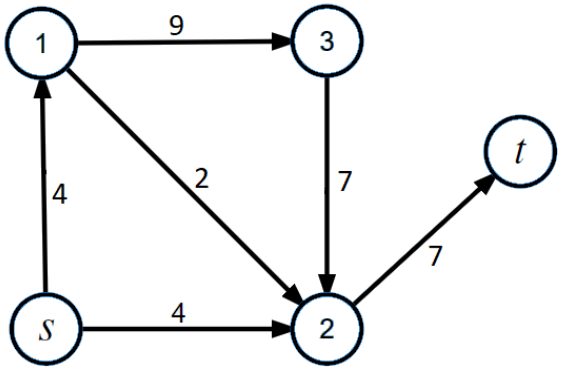
Dokaz. Nakon svake zasićene operacije Push((v,w)), ista se može dogoditi tek nakon što se vrijednost $\psi(w)$ poveća barem za 2, dogodi se operacija Push((w,v)) te se vrijednost $\psi(v)$ poveća barem za 2. Dakle, između svake dvije zasićene operacije Push((v,w)) nad fiksnim bridom (v,w) vrijednosti $\psi(v)$ i $\psi(w)$ povećaju se barem za 2 pa iz Leme 5.(a) i Leme 5.(b)

zaključujemo da postoji najviše n zasićenih operacija Push((v,w)), za svaki $(v,w) \in E(\overleftrightarrow{G})$. ■

Može se pokazati kako je ukupan broj nezasićenih Push operacija tokom obrade algoritma najviše $8n^2\sqrt{m}$ te kako je složenost GT algoritma $O(n^2\sqrt{m})$. Dokazi navedenih tvrdnji tehnički su zahtjevni stoga ih ovdje nećemo obrađivati.

Primjer GT algoritma

Neka je zadan Graf G sa slike 27.



Slika 27: Graf G.

Promotrimo sliku 28.

Izraz x, y na svakome vrhu grafa označava vrijednosti $\psi(v)$,ex$_f(v)$ u trenutnoj fazi algoritma. Na grafu (a) izvršeni su koraci ① i ②. Na slici (a) iz ex$_f(2) = 4$ zaključujemo da je vrh 2 aktivan, te ga izabiremo u koraku ③. Kako u grafu G_f ne postoji dopustiv brid $e \in \delta^+_{G_f}(2)$, pokrećemo operaciju Relabel(2) i dobivamo $\psi(2) := \min\{\psi(w)+1 \mid e = (2,w) \in E(G_f)\} = \psi(t) + 1 = 1$ (kako je u vrijednosti $\psi(t)$ postignut minimum, na slici (b) označili smo ju crvenom bojom). Brid $e = (2,t)$ sada je dopustiv pa pokrećemo operaciju Push($(2,t)$): stavljamo $\gamma := \min\{\text{ex}_f(2), u_f((2,t))\} = 4$ te povećavamo f duž $(2,t)$ za $\gamma = 4$. Operacija je prikazana na slici (b).

Na slici (b) iz ex$_f(1) = 4$ zaključujemo da je vrh 1 aktivan, te ga izabiremo u koraku ③. Kako u grafu G_f ne postoji dopustiv brid $e \in \delta^+_{G_f}(1)$, pokrećemo operaciju Relabel(1) i dobivamo $\psi(1) := \min\{\psi(w)+1 \mid e = (1,w) \in E(G_f)\} = \psi(3) + 1 = 1$. Brid $e = (1,3)$ sada je dopustiv pa pokrećemo operaciju Push($(1,3)$): stavljamo $\gamma := \min\{\text{ex}_f(1), u_f((1,3))\} = 4$ te povećavamo f duž $(1,3)$ za $\gamma = 4$. Operacija je prikazana na slici (c).

Na slici (c) iz ex$_f(3) = 4$ zaključujemo da je vrh 3 aktivan, te ga izabiremo u koraku ③.

Kako u grafu G_f ne postoji dopustiv brid $e \in \delta^+_{G_f}(3)$, pokrećemo operaciju Relabel(3) i dobivamo $\psi(3) := \min\{\psi(w)+1 \mid e=(3,w) \in E(G_f)\} = \psi(2)+1 = 2$. Brid $e=(3,2)$ sada je dopustiv pa pokrećemo operaciju Push$((3,2))$: stavljamo $\gamma := \min\{\text{ex}_f(3), u_f((3,2))\} = 4$ te povećavamo f duž $(3,2)$ za $\gamma = 4$. Operacija je prikazana na slici (d).

Na slici (d) iz $\text{ex}_f(2) = 4$ zaključujemo da je vrh 2 aktivan, te ga izabiremo u koraku ③. U grafu G_f postoji dopustiv brid $(2,t) \in \delta^+_{G_f}(2)$, pa pokrećemo operaciju Push$((2,t))$: stavljamo $\gamma := \min\{\text{ex}_f(2), u_f((2,t))\} = 3$ te povećavamo f duž $(2,t)$ za $\gamma = 3$. Operacija je prikazana na slici (e).

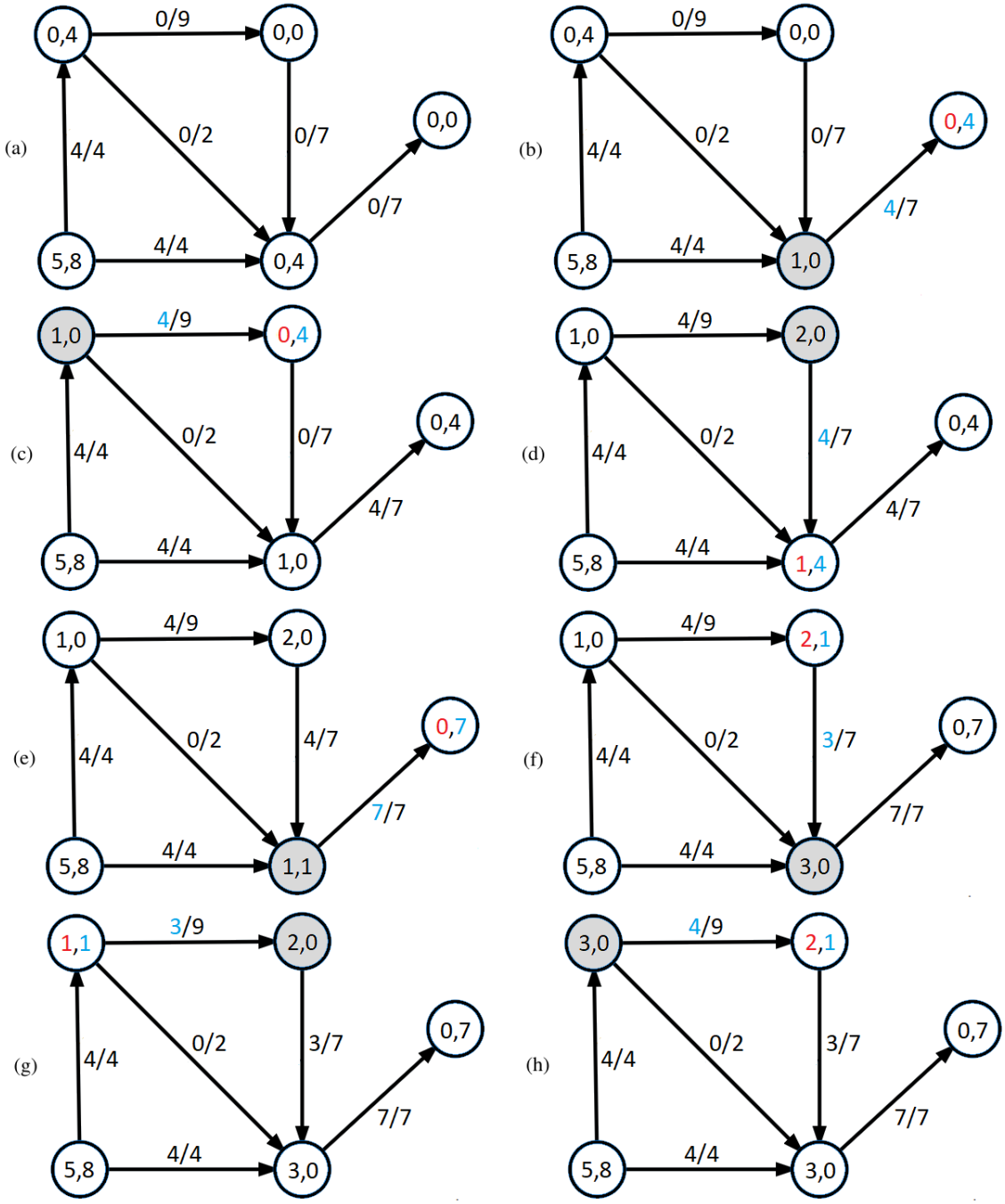
Na slici (e) iz $\text{ex}_f(2) = 1$ zaključujemo da je vrh 2 aktivan, te ga izabiremo u koraku ③. Kako u grafu G_f ne postoji dopustiv brid $e \in \delta^+_{G_f}(2)$, pokrećemo operaciju Relabel(2) i dobivamo $\psi(2) := \min\{\psi(w)+1 \mid e=(2,w) \in E(G_f)\} = \psi(3)+1 = 3$. Brid $e=(2,3)$ sada je dopustiv pa pokrećemo operaciju Push$((2,3))$: stavljamo $\gamma := \min\{\text{ex}_f(2), u_f((2,3))\} = 1$ te povećavamo f duž $(2,3)$ za $\gamma = 1$. Operacija je prikazana na slici (f).

Na slici (f) iz $\text{ex}_f(3) = 1$ zaključujemo da je vrh 3 aktivan, te ga izabiremo u koraku ③. U grafu G_f postoji dopustiv brid $(3,1) \in \delta^+_{G_f}(3)$, pa pokrećemo operaciju Push$((3,1))$: stavljamo $\gamma := \min\{\text{ex}_f(3), u_f((3,1))\} = 1$ te povećavamo f duž $(3,1)$ za $\gamma = 1$. Operacija je prikazana na slici (g).

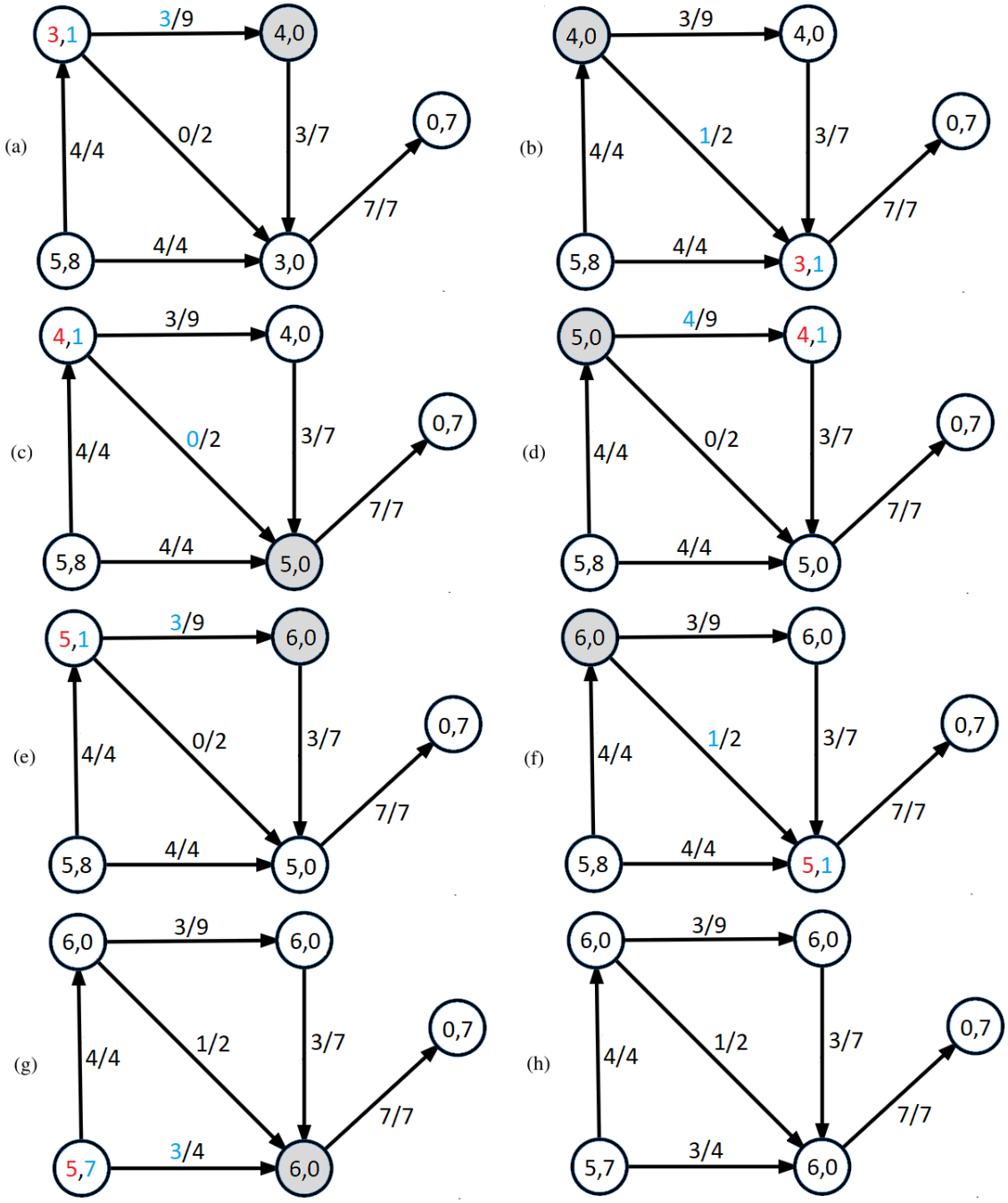
Na slici (g) iz $\text{ex}_f(1) = 1$ zaključujemo da je vrh 1 aktivan, te ga izabiremo u koraku ③. Kako u grafu G_f ne postoji dopustiv brid $e \in \delta^+_{G_f}(1)$, pokrećemo operaciju Relabel(1) i dobivamo $\psi(1) := \min\{\psi(w)+1 \mid e=(1,w) \in E(G_f)\} = \psi(3)+1 = 3$. Brid $e=(1,3)$ sada je dopustiv pa pokrećemo operaciju Push$((1,3))$: stavljamo $\gamma := \min\{\text{ex}_f(1), u_f((1,3))\} = 1$ te povećavamo f duž $(1,3)$ za $\gamma = 1$. Operacija je prikazana na slici (h).

Promotrimo sliku 29.

Postupak nastavljamo te dolazimo do situacije prikazane na slici (f). Iz $\text{ex}_f(2) = 1$ zaključujemo da je vrh 2 aktivan, te ga izabiremo u koraku ③. U grafu G_f postoji dopustiv brid $(2,s) \in \delta^+_{G_f}(2)$, pa pokrećemo operaciju Push$((2,s))$: stavljamo $\gamma := \min\{\text{ex}_f(2), u_f((2,s))\} = 1$ te povećavamo f duž $(2,s)$ za $\gamma = 1$. Operacija je prikazana na slici (g). Kako nijedan vrh $v \in V(G)\backslash\{s,t\}$ više nije aktivan, prekidamo algoritam. Graf prikazan na slici (h) jest traženo rješenje problema.



Slika 28: GT algoritam (1/2).

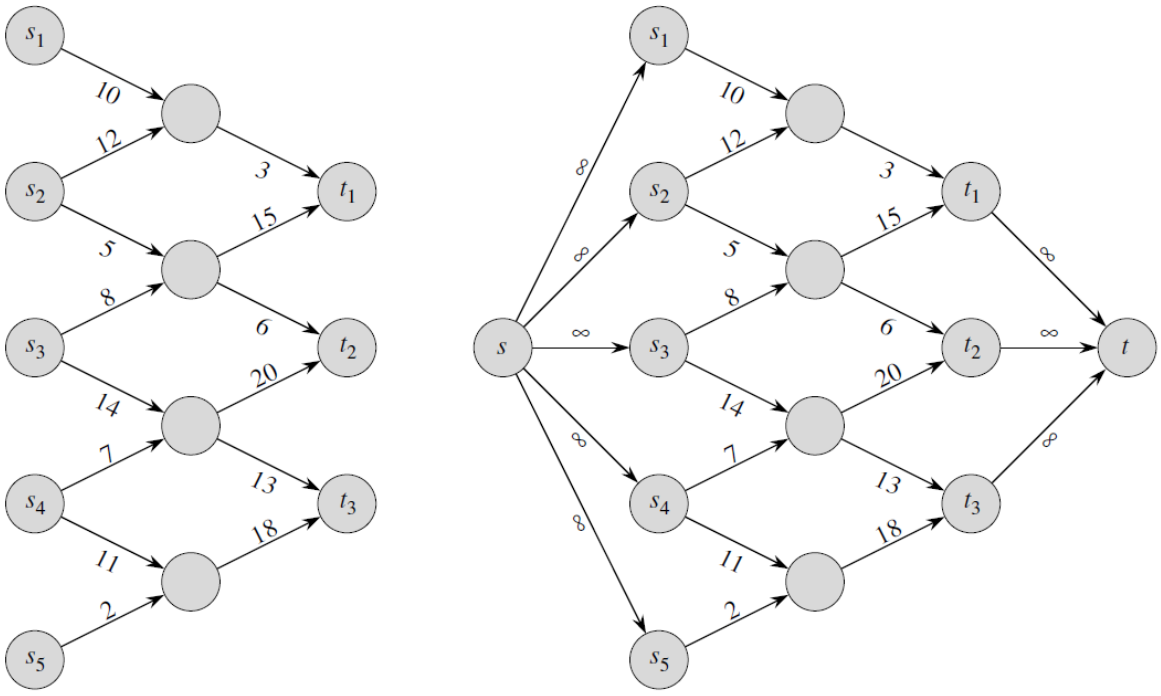


Slika 29: GT algoritam (2/2).

4.2 Tokovi s više ponora i izvora

Problem traženja toka maksimalne vrijednosti može se proširiti na više izvora i ponora. Neka je zadan k_1 izvor i k_2 ponora, $k_1, k_1 \in \mathbb{N}$. Vrijednost toka f koju maksimiziramo dana je sa $|f| := -\sum_{i=1}^{k_1} \text{ex}_f(s_i)$.

Ovaj problem na jednostavan način svodimo na razmatrani problem s jednim izvorom i ponorom. Stvaramo novi vrh s koji zovemo **superizvor** i dodajemo bridove (s, s_i), $i = 1, ..., k_1$ sa pripadnim težinama $c(s, s_i) := \infty$, $i = 1, ..., k_1$. Također stvaramo novi vrh t koji zovemo **superponor** i dodajemo bridove (t_j, t), $j = 1, ..., k_2$ sa pripadnim težinama $c(t_j, t) := \infty$, $j = 1, ..., k_2$. Korespondencija dvaju problema je jasna.



Slika 30: Traženje toka maksimalne vrijednosti s više ponora i izvora. [2]

5 Literatura

[1] B. Korte, J. Vygen, *Combinatorial Optimization Theory and Algorithms, Third Edition*, Springer, Bonn, svibanj 2005.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms, Second Edition*, The MIT Press, London, svibanj 2001.

[3] Wikipedia, *Edmonds–Karp algorithm*, dostupno na `https://en.wikipedia.org/wiki/Edmonds%E2%80%93Karp_algorithm` (veljača 2018.).

[4] Wikipedia, *Dinic's algorithm* , dostupno na `https://en.wikipedia.org/wiki/Dinic%27s_algorithm` (veljača 2018.).

6 Sažetak

Nakon svih obrađenih algoritama upoznali smo nekoliko načina dolaska do rješenja dvaju problema koje smo promatrali. Područje diskretne matematike s vremenom sve se više razvija, te što više ulazimo u dubinu, teorijsko poznavanje upravo ovih ”osnovnijih” algoritama daje dobru podlogu za daljnji rad te uvelike pomaže. Upravo zbog svojstva matematičke teorije koja se postupno nadograđuje sama na sebe, veoma je zahvalno shvatiti ovu osnovnu podlogu, među kojom spada i gradivo obrađeno u ovome radu. Iz tog razloga, dobro je što postoji poveći broj algoritama za rješavanje osnovnijih problema diskretne matematike kako bi se daljnja znanost mogla razvijati u što više smjerova. Obrađeni algoritmi daju dobru bazu za ulazak u kompleksnije matematičke probleme, te razvijaju logičko razmišljanje.

7 Summary

After all the algorithms we’ve processed, we’ve come up with a few ways to get to the solution of the two problems we’ve been observing. The field of discrete maths is evolving over time, and the more we enter into depth, the theoretical knowledge of these ”basic” algorithms provides a good foundation for further work and greatly helps. Because of the mathematical theory that gradually improves itself, it is very grateful to understand this basic foundation, among which the material covered in this paper. For this reason, it is good that there is an increasing number of algorithms for solving the most basic problems of discrete maths so that further science can evolve into as many directions as possible. The processed algorithms provide a good basis for entering complex mathematical problems and develop logical thinking.

8 Životopis

Zovem se Apolinar Barbiš. Rođen sam 5. listopada 1994. godine u Rijeci. Živim u Malinskoj, mjestu od oko 2000 stanovnika na otoku Krku u kojem sam proveo svoje djetinjstvo i ranu mladost. Završio sam osnovnu školu Fran Krsto Frankopan, također u Malinskoj. Srednju školu pohađao sam u Pazinu, i to Pazinski Kolegij - Klasičnu Gimnaziju. Nakon toga sam upisao, i u roku s odličnim uspjehom završio tri godine Preddiplomskog Sveučilišnog studija Matematike u Rijeci te time stekao status prvostupnika matematike. Slijedeći svoju želju za nastavkom studija u tom pravcu, akademske godine 2016./17. upisujem prvu godinu diplomskog studija na Prirodoslovno Matematičkom Fakultetu u Zagrebu, smjer: Financijska i Poslovna Matematika. Trenutno završavam drugu godinu studija. Svoje slobodno vrijeme najviše provodim u raznoraznim aktivnostima "molitvene zajednice Srce Isusovo" iz Zagreba čiji sam odnedavno član. Osim toga, rekreacijski igram nogomet, tenis i šah.