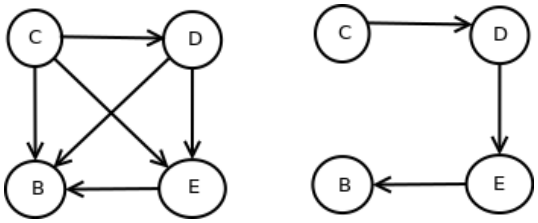
Grafovski algoritmi - čas 7

Tranzitivna redukcija

Tranzitivna redukcija predstavlja operaciju nad grafovima inverznu operaciji pronalaženja tranzitivnog zatvorenja grafa. Dakle, cilj je da se za dati usmereni graf konstruiše usmereni graf sa istim skupom čvorova i što manjim skupom grana, a da se pritom ne promeni relacija dostižnosti. Tranzitivno zatvorenje grafa G jednako je tranzitivnom zatvorenju tranzitivne redukcije grafa G. Iako je tranzitivno zatvorenje grafa G jedinstveno određeno, u opštem slučaju graf može imati više različitih tranzitivnih redukcija.

Pokazuje se da je u slučaju acikličkog usmerenog grafa G, tranzitivna redukcija grafa G jedinstvena i uvek je podgraf datog grafa (skup grana grafa koji predstavlja tranzitivnu redukciju je podskup skupa grana polaznog grafa G). Međutim, ovo ne mora da važi za grafove koji sadrže cikluse.



Slika 1: Aciklički usmereni graf i njegova tranzitivna redukcija.

Dokažimo najpre da je tranzitivna redukcija usmerenog acikličkog grafa G jedinstvena. Pretpostavimo da ovo nije tačno i da postoje dve različita grafa G_1 i G_2 tako da su oba tranzitivne redukcije polaznog grafa G. S obzirom na to da su ova dva grafa različita, a imaju isti skup čvorova, postoji bar jedna grana (u, v) u kojoj se razlikuju; bez smanjenja opštosti pretpostavimo da grana (u, v) pripada grafu G_1, a ne pripada grafu G_2. Međutim, s obzirom na to da su relacije dostižnosti ova dva grafa jednake, u grafu G_2 mora postojati usmereni put od čvora u do čvora v koji prolazi kroz neki čvor w. Na osnovu ekvivalentnosti relacije dostižnosti sledi da graf G_1 mora sadržati usmereni put od čvora u do čvora w i usmereni put od čvora w do čvora v. Ako put od čvora u do čvora w u grafu G_1 uključuje granu (u, v), onda postoji i usmereni put od čvora v do čvora w, i pritom, kako smo već zaključili, on sadrži i usmereni put od w do v te dobijamo da graf sadrži ciklus, suprotno pretpostavci. Dakle, put od u do w ne uključuje granu (u, v). Slično se pokazuje da i put od w do v ne sadrži granu (u, v). Dakle, graf G_1 sadrži put od čvora u do čvora w i put od čvora w do čvora v, a pritom nijedan od njih ne uključuje granu (u, v), te graf G_1 bez

grane (u, v) predstavlja manji graf čija je relacija dostižnosti jednaka relaciji dostižnosti grafa G_1 te graf G_1 ne može biti tranzitivna redukcija polaznog grafa. Zaključujemo da je tranzitivna redukcija acikličkog grafa jedinstvena.

Kako doći do tranzitivne redukcije acikličkog grafa? Ona se sastoji od grana koje predstavljaju jedine puteve između njenih krajeva. Dakle za proizvoljnu granu (x, y) želimo da razmotrimo da li postoji neki drugi put od čvora x do čvora y jer ako postoji drugi put, onda se grana (x, y) ne treba naći u tranzitivnoj redukciji grafa. Slično kao u algoritmu za pronalaženje tranzitivnog zatvorenja datog grafa zasnovanom na Flojd-Varšalovom algoritmu, možemo da razmatramo redom k-puteve i da ukoliko postoji $(k-1)$-put od čvora x do čvora v_k i $(k-1)$-put od čvora v_k do čvora y, onda u tom grafu ne treba da bude i grana između čvorova x i y.

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int>> listaSuseda {{1, 2, 3}, {3, 4}, {5}, {},
                    {6, 7}, {8}, {}, {}, {}};

// funkcija koja racuna tranzitivnu redukciju datog aciklickog grafa
void izracunajTranzitivnuRedukciju() {

  int brojCvorova = listaSuseda.size();
  // matrica povezanosti grafa tranzitivne redukcije
  vector<vector<bool>> tranzitivnaRedukcija(brojCvorova);

  // inicialno postavljamo sve vrednosti u matrici na false
  for (int i=0; i<brojCvorova; i++){
    tranzitivnaRedukcija[i].resize(brojCvorova,false);
  }

  // inicijalizujemo matricu povezanosti grafa tranzitivne redukcije
  // na matricu povezanosti polaznog grafa
  for (int i=0; i<brojCvorova; i++)
    for (int j=0; j<listaSuseda[i].size(); j++){
      int k = listaSuseda[i][j];
      tranzitivnaRedukcija[i][k] = true;
    }

  // prolazimo redom m-putevima i gledamo da li postoji drugi put
  // izmedju cvorova i i j
  for (int m=0; m<brojCvorova; m++)
    for (int i=0; i<brojCvorova; i++)
      if (tranzitivnaRedukcija[i][m])
        for (int j=0; j<brojCvorova; j++)
```

```
            // ako postoji put izmedju cvorova i i m
            // i put izmedju cvorova m i j
            // onda se u tranzitivnoj redukcije ne treba naci grana (i,j)
            if (tranzitivnaRedukcija[m][j])
               tranzitivnaRedukcija[i][j] = false;

  cout << "Matrica susedstva grafa tranzitivne redukcije je" << endl;
  for (int i=0; i<brojCvorova; i++){
    for (int j=0; j<brojCvorova; j++)
      cout << tranzitivnaRedukcija[i][j] << " ";
    cout << endl;
  }
}

int main() {
  izracunajTranzitivnuRedukciju();
  return 0;
}
```

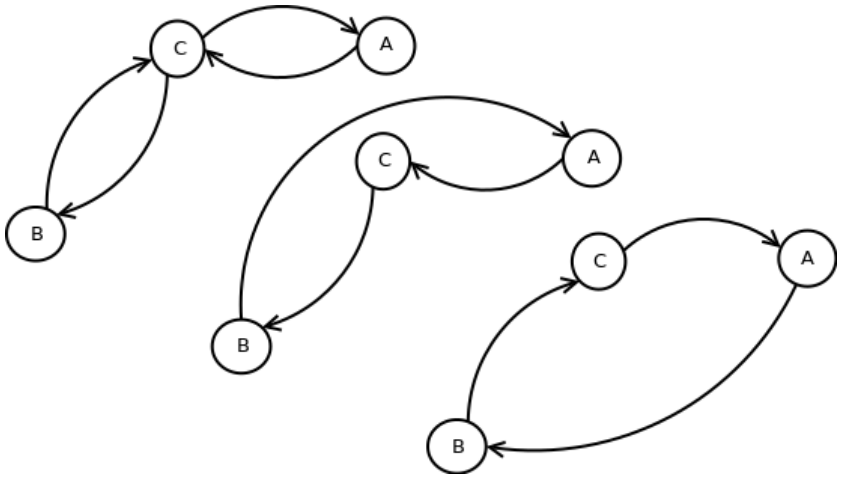
U slučaju kada polazni graf nije aciklički, ako želimo da graf tranzitivne redukcije bude nužno i podgraf polaznog grafa, dobijamo problem koji je težak za rešavanje. Zaista, razmotrimo usmereni graf koji sadrži samo jednu jaku komponentu povezanosti, dakle svaki čvor je dostižan iz svakog drugog čvora u grafu. Tranzitivnu redukciju ovog grafa činio bi prost usmereni ciklus, koji obilazi sve čvorove u grafu. Ovo je u stvari problem ispitivanja da li je graf Hamiltonov, a on je NP-kompletan. Iz ovog razloga se često ne insistira da graf koji predstavlja tranzitivnu redukciju bude podgraf polaznog grafa (i sama definicija tranzitivne redukcije ne zahteva da rezultujući graf bude podgraf polaznog grafa).

U grafu koji može imati cikluse tranzitivna redukcija ne mora biti jedinstvena: može postojati veći broj različitih grafova sa istim skupom čvorova koji ima minimalni broj grana i istu relaciju dostižnosti kao polazni graf. Dodatno, može se desiti da nijedan od ovih minimalnih grafova nije podfgraf datog grafa (slika 2).

Kako doći do tranzitivne redukcije grafa koji može sadržati cikluse (ako ne tražimo da rezultujući graf bude podgraf polaznog grafa)? Važi sledeće:

- za svaku jaku komponentu povezanosti treba dodati usmereni ciklus (koji povezuje sve čvorove te komponente) i
- treba pronaći tranzitivnu redukciju "kompresovanog" grafa čiji su čvorovi komponente jake povezanosti (taj graf je aciklički pa možemo primeniti gornji algoritam) i ako u njoj postoji grana od komponente X do komponente Y, onda za neka dva čvora $x \in X$ i $y \in Y$ treba dodati granu od x do y.

Ukupan broj grana u ovom tipu tranzitivne redukcije jednak je zbiru broju grana u tranzitivnoj redukciji kompresovanog grafa i broja čvorova u netrivijalnim



Slika 2: Usmereni graf i njegove dve različite tranzitivne redukcije.

jakim komponentama povezanosti (komponente sa više od jednog čvora).

Grane tranzitivne redukcije koje odgovaraju granama u kompresovanom grafu se mogu birati tako da pripadaju polaznom grafu, međutim ciklus u okviru svake jake komponente povezanosti se može izabrati da pripada polaznom grafu, ako ta komponenta ima Hamiltonov ciklus.

Uparivanje

Za zadati neusmereni graf $G = (V, E)$ *uparivanje* je skup disjunktnih grana (grana bez zajedničkih čvorova). Ovo ime potiče od činjenice da se grane mogu shvatiti kao parovi čvorova. Bitno je da svaki čvor pripada najviše jednoj grani. Čvor koji nije susedan ni jednoj grani iz uparivanja zove se *neupareni* čvor; kaže se takođe da čvor ne pripada uparivanju. *Savršeno uparivanje* je uparivanje u kome su svi čvorovi upareni. *Optimalno uparivanje* je uparivanje sa maksimalnim brojem grana. *Maksimalno uparivanje* je pak uparivanje koje se ne može proširiti dodavanjem nove grane. Problemi koji se svode na uparivanje pojavljuju se u mnogim situacijama, ne samo socijalnim. Mogu se uparivati radnici sa radnim mestima, mašine sa delovima, grupe studenata sa učionicama, itd.

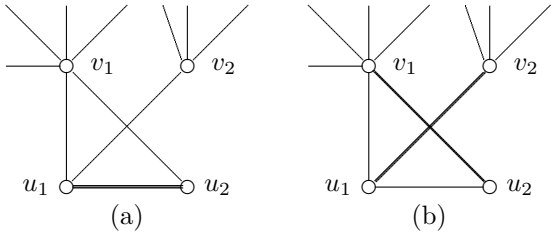
Problem nalaženja optimalnog uparivanja za proizvoljan graf je težak problem. Mi ćemo razmotriti dva specijalna slučaja grafa.

Savršeno uparivanje u vrlo gustim grafovima

Neka je $G = (V, E)$ neusmereni graf kod koga je $|V| = 2n$ i stepen svakog čvora je bar n. Prikazaćemo algoritam za nalaženje savršenog uparivanja u ovakvim

grafovima. Posledica ovog algoritma je da ako graf zadovoljava navedene uslove, onda u njemu uvek postoji savršeno uparivanje. Koristićemo indukciju po veličini m uparivanja. Bazni slučaj $m = 1$ rešava se formiranjem uparivanja veličine jedan od proizvoljne grane grafa. Pokazaćemo da se proizvoljno uparivanje koje nije savršeno može proširiti ili dodavanjem jedne grane ili zamenom jedne grane dvema novim granama. U oba slučaja veličina uparivanja se povećava za jedan.

Posmatrajmo uparivanje M sa m grana u grafu G, pri čemu je $m < n$. Najpre proveravamo sve grane van uparivanja M da ustanovimo da li se neka od njih može dodati u M. Ako pronađemo takvu granu, problem je rešen — nađeno je veće uparivanje. U protivnom, M je maksimalno uparivanje. Ako M nije savršeno uparivanje, postoje bar dva neuparena čvora v_1 i v_2. Iz ta dva čvora po pretpostavci izlazi ukupno najmanje $2n$ grana. Sve te grane vode ka uparenim čvorovima (u protivnom bi se u uparivanje mogla dodati nova grana, suprotno pretpostavci da je ono maksimalno). Pošto u uparivanju M ima manje od n grana, a iz v_1 i v_2 izlazi bar $2n$ grana, u uparivanju M postoji grana (u_1, u_2) koja je susedna sa ("pokriva") bar tri grane iz v_1 i v_2. Pretpostavimo, bez smanjenja opštosti, da su to grane (u_1, v_1), (u_1, v_2) i (u_2, v_1), videti sliku 3(a). Lako je videti da se uklanjanjem grane (u_1, u_2) iz uparivanja M i dodavanjem dveju novih grana (u_1, v_2) i (u_2, v_1) dobija veće uparivanje, slika 3(b).



Slika 3: Proširivanje uparivanja.

Opisani algoritam je primer pohlepnog pristupa. U svakom koraku proširivanja uparivanja za jednu granu razmatraju se četiri čvora i nekoliko grana koje ih povezuju. U ovoj situaciji je to bilo dovoljno; međutim, u opštem slučaju je nalaženje dobrog uparivanja teži problem. Uključivanje jedne grane u uparivanje utiče na izbor drugih grana čak i u udaljenim delovima grafa. Pokazaćemo sada kako se ovaj pristup može primeniti na drugi specijalni slučaj problema uparivanja.

Bipartitno uparivanje

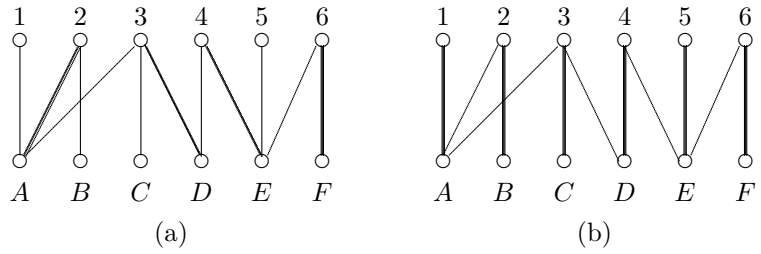
Bipartitni graf je graf čiji se čvorovi mogu podeliti na dva disjunktna podskupa tako da u grafu postoje samo grane između čvorova iz različitih podskupova.

Neka je $G = (V, E, U)$ bipartitni graf u kome su V i U disjunktni skupovi čvorova, a E je skup grana koje povezuju neke čvorove iz V sa nekim čvorovima iz U.

Problem: Pronaći uparivanje sa maksimalnim brojem grana u bipartitnom grafu G.

Jedan od načina da se formuliše problem je sledeći: V je skup devojaka, U je skup mladića, a E je skup "potencijalnih" parova. Cilj je pod ovim uslovima oformiti što veći broj parova mladića i devojaka.

Direktan pristup je formirati parove u skladu sa nekom strategijom, do trenutka kad dalja uparivanja više nisu moguća — u nadi da će nam strategija obezbediti optimalno ili rešenje blisko optimalnom. Može se, na primer, pokušati sa pohlepnim pristupom, uparujući najpre čvorove malog stepena, u nadi da će preostali čvorovi i u kasnijim fazama imati neuparene partnere. Drugim rečima, najpre uparujemo stidljive osobe, one sa manje poznanstava, a o ostalima brinemo kasnije. Umesto da se bavimo analizama ovakvih strategija (što nije jednostavan problem), pokušaćemo sa pristupom korišćenim kod prethodnog problema. Pretpostavimo da se polazi od maksimalnog uparivanja, koje ne mora biti optimalno. Možemo li ga nekako popraviti? Pogledajmo primer na slici 4(a), na kome je uparivanje prikazano podebljanim granama. Jasno je da se uparivanje može povećati zamenom grane $2A$ sa dve grane $1A$ i $2B$. Ovo je transformacija slična onoj koju smo primenili u prethodnom problemu. Međutim, ne moramo se ograničiti zamenama jedne grane dvema granama. Ako pronađemo sličnu situaciju u kojoj se nekih k grana mogu zameniti sa $k+1$ grana, dobijamo algoritam većih mogućnosti. Na primer, uparivanje se može dalje povećati zamenom grana $3D$ i $4E$ sa tri grane $3C$, $4D$ i $5E$, slika 4(b).



Slika 4: Proširivanje bipartitnog uparivanja.

Razmotrimo detaljnije ove transformacije. Cilj je povećati broj uparenih čvorova. Polazimo od neuparenog čvora v i pokušavamo da ga uparimo. Pošto polazimo od maksimalnog uparivanja, svi susedi čvora v su već upareni; zbog toga smo prinuđeni da iz uparivanja uklonimo neku od grana koje "pokrivaju" susede v. Pretpostavimo da smo izabrali čvor u, susedan sa v, koji je prethodno bio uparen sa čvorom w, na primer. Raskidamo uparivanje u sa w, i uparujemo v sa u. Sada preostaje da pronađemo para za čvor w. Ako je w povezan granom sa nekim neuparenim čvorom, onda smo postigli cilj; takav je bio prvi od gornjih slučajeva. Ako to nije slučaj, onda nastavljamo dalje sa raskidanjem parova i formiranjem novih parova. Da bismo na osnovu ove ideje konstruisali algoritam, potrebno je da uradimo dve stvari:

- da obezbedimo da se procedura uvek završava,
- da pokažemo da ako je poboljšanje moguće, da će ga ovako opisana procedura sigurno pronaći.

Najpre ćemo formalizovati navedenu ideju.

Alternirajući put (ili *povećavajući put*) P za dato uparivanje M je put od neuparenog čvora $v \in V$ do neuparenog čvora $u \in U$, pri čemu su grane puta P naizmenično u $E \setminus M$, odnosno M. Drugim rečima, prva grana (v, w) puta P ne pripada M (jer je v neuparen), druga grana (w, x) pripada M, i tako dalje do poslednje grane (z, u) puta P koja ne pripada M. Zapazimo da su upravo alternirajući putevi u gornjim primerima omogućavali povećavanje uparivanja. Specijalno, ako je put dužine jedan, onda je to grana koja povezuje dva neuparena čvora; takve grane ne postoje u odnosu na maksimalno uparivanje. Broj grana na putu P mora biti neparan, jer P polazi iz V i završava u U. Pored toga, među granama puta P grana u $E \setminus M$ ima za jednu više od grana u M. Prema tome, ako iz uparivanja izbacimo sve grane P koje su u M, a uključimo sve grane P koje su u $E \setminus M$, dobićemo novo uparivanje sa jednom granom više. Na primer, prvi alternirajući put korišćen za povećanje uparivanja na slici 4(a) je put $(1A, A2, 2B)$, i on omogućuje zamenu grane $A2$ granama $1A$ i $2B$; drugi alternirajući put $(C3, 3D, D4, 4E, E5)$ omogućuje zamenu grana $3D$ i $4E$ granama $C3$, $D4$ i $E5$.

Jasno je da ako za dato uparivanje M postoji alternirajući put, onda M nije optimalno uparivanje. Ispostavlja se da je tačno i obrnuto tvrđenje.

Teorema[Teorema o alternirajućem putu (Beržova teorema)]: Uparivanje je optimalno ako i samo u odnosu na njega ne postoji alternirajući put.

Za dokaz ovog tvrđenja biće nam potrebna naredna lema:

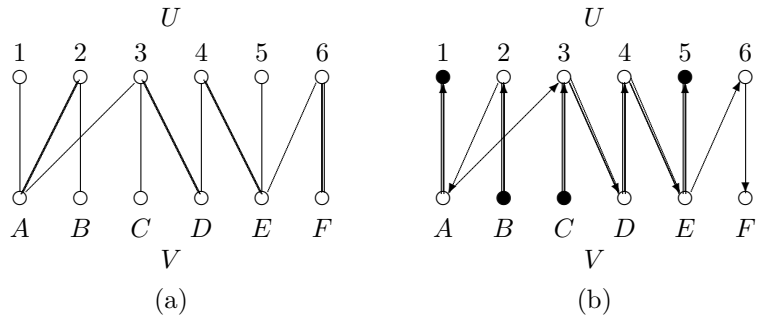
Lema: Neka je stepen svakog čvora u datom grafu najviše 2. Tada je svaka komponenta povezanosti tog grafa ili izolovani čvor ili put ili ciklus.

Dokaz: Posmatrajmo proizvoljni neizolovani čvor u grafu. Svi njegovi susedi (kojih može biti maksimalno dva) imaju takođe maksimalni stepen 2. Dakle komponenta povezanosti koja sadrži ovaj čvor je ili put ili ciklus.

Dokažimo sada drugi smer gornje teoreme. Pretpostavimo da uparivanje M nije optimalno, a da u grafu u odnosu na njega ne postoji alternirajući put. Neka je M' optimalno uparivanje, odnosno važi $|M| < |M'|$. Razmotrimo graf sa skupom grana $M \oplus M'$, svaki čvor ovog grafa je stepena najviše 2 (jer ova dva uparivanja mogu doprineti stepenu svakog čvora maksimalno sa 1), te se na ovaj graf može primeniti prethodna lema, odnosno važi da je svaka komponenta povezanosti izolovani čvor ili put ili ciklus. S obzirom na to da grane na svakom ciklusu alterniraju u smislu pripadnosti skupovima M i M', to znači da ciklusi moraju biti parne dužine. Dakle, M' može biti duže od M jedino posredstvom puteva, s obzirom da ciklusi otpadaju kao mogućnost. Dakle, postoji barem jedan put iz $M \oplus M'$ koji ima više grana iz M' nego iz M, ali takav put je alternirajući za dato uparivanje M te je pretpostavka da u odnosu na uparivanje

M ne postoji alternirajući put netačna. Dakle, uparivanje mora biti optimalno, te smo dokazali i drugi smer gornje teoreme.

Teorema o alternirajućem putu direktno sugeriše algoritam, jer proizvoljno uparivanje koje nije optimalno ima alternirajući put, a alternirajući put daje povećano uparivanje. Započinjemo sa pohlepnim algoritmom, dodajući grane u uparivanje sve dok je to moguće. Onda prelazimo na traženje alternirajućih puteva i povećavanje uparivanja, sve do trenutka kad više nema alternirajućih puteva u odnosu na poslednje uparivanje. Dobijeno uparivanje je tada optimalno. Pošto alternirajući put povećava uparivanje za jednu granu, a u uparivanju ima najviše $n/2$ grana (gde je n broj čvorova), broj iteracija je najviše $n/2$. Preostaje još jedan problem — kako pronalaziti alternirajuće puteve? Problem se može rešiti na sledeći način. Transformišemo neusmereni graf G u usmereni graf G' usmeravajući grane iz M od U ka V, a grane iz $E \setminus M$ od V ka U. Slika 5(a) prikazuje polazno maksimalno uparivanje za graf sa slike 4(a), a slika 5(b) prikazuje odgovarajući usmereni graf G'. Alternirajući put u G tada odgovara usmerenom putu od neuparenog čvora u V do neuparenog čvora u U u grafu G'. Takav usmereni put može se pronaći bilo kojim postupkom obilaska grafa, npr. pomoću DFS. Složenost obilaska (pretrage) je $O(|V| + |E|)$, pa je složenost algoritma $O(|V|(|V| + |E|))$.



Slika 5: Nalaženje alternirajućih puteva

Pošto kompletan obilazak grafa u najgorem slučaju može da traje koliko i nalaženje jednog puta, može se pokušati sa nalaženjem više alternirajućih puteva jednom pretragom. Potrebno je, međutim, da budemo sigurni da su ovi putevi nezavisni, odnosno da njihovi skupovi čvorova budu disjunktni. Ako su putevi disjunktni, onda utiču na uparivanje različitih čvorova, pa se mogu istovremeno iskoristiti. Novi, poboljšani algoritam za nalaženje alternirajućih puteva je sledeći. Najpre primenjujemo BFS na graf G' od skupa neuparenih čvorova u V, sloj po sloj, do prvog sloja k u kome su pronađeni neupareni čvorovi iz U. Zatim iz grafa indukovanog pretragom u širinu vadimo maksimalni skup disjunktnih puteva u G', kojima odgovaraju alternirajući putevi u G do neuparenih čvorova na nivou k. To se izvodi pronalaženjem prvog puta, uklanjanjem njegovih čvorova, pronalaženjem narednog puta, uklanjanjem njegovih čvorova, itd (rezultat nije optimalni, nego samo maksimalni skup ovakvih puteva). Biramo maksimalni

skup, da bismo posle pretrage dobili što veće uparivanje; svaki novi disjunktni put povećava uparivanje za jednu granu. Na kraju povećavamo uparivanje korišćenjem pronađenog skupa disjunktnih puteva. Proces se nastavlja sve dok je moguće pronaći alternirajuće puteve, odnosno dok je u grafu G' neki neupareni čvor iz V dostižan iz nekog neuparenog čvora iz U.

U svakoj iteraciji pronalazi se skup najkraćih alternirajućih puteva koji su svi međusobno disjunktni po skupovima čvorova.

Ako sa M' označimo optimalno uparivanje u grafu G, broj alternirajućih puteva koji su međusobno disjunktni po skupu čvorova jednak je $|M'| - |M|$.

Neka je l dužina najkraćeg alternirajućeg puta. Svaki po čvorovima disjunktni alternirajući put ima barem $l+1$ čvorova, i kako je dužina najkraćeg alternirajućeg puta l, iz toga sledi da je $(|M'| - |M|) \cdot (l + 1) \leq n$. Ovu nejednakost možemo zapisati i na ovaj način $|M'|-|M| \leq \frac{n}{l+1}$ i na taj način ona daje gornje ograničenje ukupnog mogućeg broja različitih puteva za kojima se traga u svakoj iteraciji petlje.

Kako se dužina najkraćeg alternirajućeg puta povećava u svakoj iteraciji, to znači da će nakon $\sqrt{n}$ iteracija dužina najkraćeg takvog puta biti barem $\sqrt{n}$, a primenom poslednje nejednakosti znamo da je ukupan broj preostalih (neistraženih) najkraćih alternirajućih puteva $\frac{n}{\sqrt{n}+1} \leq \sqrt{n}$. Dakle, nakon ove iteracije čak i ako u svakoj iteraciji otkrivamo samo po jedan alternirajući put, biće nam potrebno još najviše $\sqrt{n}$ iteracija. Dakle maksimalni broj iteracija petlje je $2\sqrt{n}$, odnosno $O(\sqrt{n})$. Ukupna vremenska složenost algoritma je dakle $O((|V| + |E|)\sqrt{|V|})$. Ovaj algoritam predložili su Hopkroft i Karp 1973. godine.

Optimizacija transportne mreže

Problem optimizacije transportne mreže je jedan od osnovnih problema u teoriji grafova i kombinatornoj optimizaciji. Intenzivno je proučavan više od 40 godina, pa su za njega razvijeni mnogi algoritmi i strukture podataka. Problem ima mnogo varijanti i uopštenja. Osnovna varijanta mreže može se formulisati na sledeći način. Neka je $G = (V, E)$ usmereni graf sa dva posebno izdvojena čvora: s (izvor), sa ulaznim stepenom 0, i t (ponor) sa izlaznim stepenom 0. Svakoj grani $e \in E$ pridružena je pozitivna težina $c(e)$, *kapacitet* grane e. Kapacitet grane je mera toka koji može biti propušten kroz granu. Za ovakav graf kažemo da je *transportna mreža* (ili jednostavnije mreža). *Tok* je funkcija f definisana na E koja zadovoljava sledeće uslove:

1. $0 \leq f(e) \leq c(e)$: tok kroz proizvoljnu granu ne može da premaši njen kapacitet;
2. za sve čvorove $v \in V \setminus \{s, t\}$ je $\sum_u f(u, v) = \sum_w f(v, w)$: ukupan tok koji ulazi u proizvoljni čvor v različit od s, t jednak je ukupnom toku koji izlazi iz njega ("nestišljivost", zakon očuvanja, odnosno konzervacije toka).

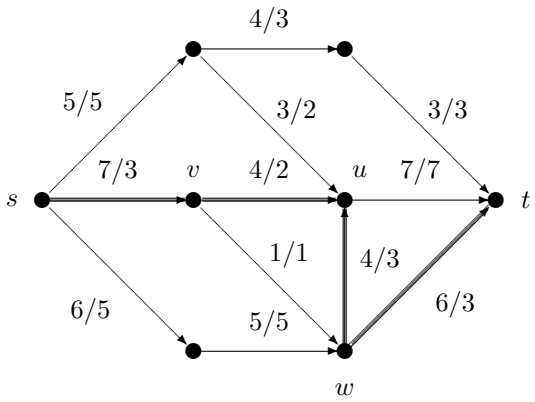
Ova dva uslova imaju za posledicu da je ukupan tok koji izlazi iz s jednak ukupnom toku koji ulazi u t. U to se možemo uveriti na sledeći način. Uvešćemo najpre pojam *preseka*. Neka je A proizvoljan podskup skupa V takav da sadrži s, a ne sadrži t. Označimo sa $B = V \setminus A$ skup preostalih čvorova. Presek određen skupom A je skup grana $(v, u) \in E$ takvih da $v \in A$ i $u \in B$. Intuitivno, presek je skup grana koje razdvajaju s od t. Indukcijom po broju čvorova u A lako se pokazuje da ukupan tok kroz presek ne zavisi od A. Specijalno, za $A = \{s\}$ presek obuhvata grane koje izlaze iz s, a za $A = V \setminus \{t\}$ presek čine grane koje ulaze u t. Prema tome, ukupan tok koji izlazi iz s jednak je ukupnom toku koji ulazi u t. Problem koji nas zanima je *maksimiziranje toka*. U slučaju kad su kapaciteti grana realni brojevi, nije očigledno čak ni da maksimalni tok uvek postoji; pokazaćemo da on uvek postoji. Jedan način da se opisani problem shvati kao realan fizički problem, je da zamislimo da mrežu čine cevi za vodu. Svaka cev ima svoj kapacitet, a uslovi koje tok treba da zadovolji su prirodni. Cilj je "proterati" kroz mrežu što veću količinu vode u jedinici vremena.

Povećavajući put u odnosu na zadati tok f je usmereni put od s do t, koji se sastoji od grana iz G, ne obavezno u istom smeru; svaka od tih grana (v, u) treba da zadovolji tačno jedan od sledeća dva uslova:

1. (v, u) ima isti smer kao i u G, i $f(v, u) < c(v, u)$. U tom slučaju grana (v, u) je *direktna grana*. Direktna grana ima kapacitet veći od toka, pa se kroz nju tok može povećati. Razlika $c(v, u) - f(v, u)$ zove se *slek* te grane.
2. (v, u) ima suprotan smer u G, i $f(u, v) > 0$. U ovom slučaju grana (v, u) je *povratna grana*. Deo toka iz povratne grane može se "pozajmiti".

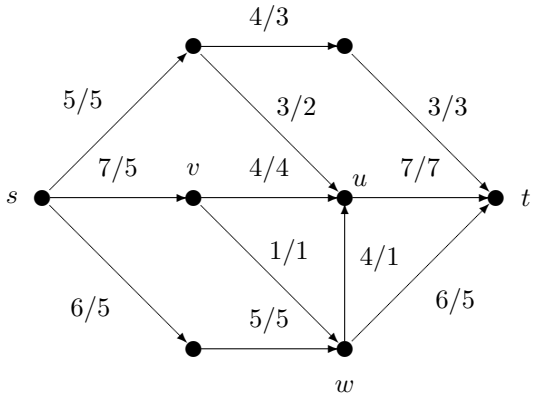
Povećavajući put ima isti smisao za transportne mreže kao alternirajući put za bipartitno uparivanje. Ako postoji povećavajući put u odnosu na tok f, onda f nije optimalni tok. Tok f može se povećati povećavanjem toka kroz povećavajući put na sledeći način. Ako su sve grane povećavajućeg puta direktne grane, onda se kroz njih može povećati tok, tako da sva ograničenja i dalje ostanu zadovoljena. Najveće moguće povećanje toka je u ovom slučaju tačno jednako minimalnom sleku među granama puta. Slučaj povratnih grana je nešto složeniji, videti primer na slici 6. Svaka grana označena je sa dva broja a/b, pri čemu je a kapacitet, a b trenutni tok. Jasno je da se ukupan tok ne može direktno povećati, jer ne postoji put od s do t koji se sastoji samo od direktnih grana. Ipak, postoji način da se ukupan tok poveća.

Put $s - v - u - w - t$ je povećavajući put. Dopunski tok 2 može se sprovesti do u od s (2 je minimalni slek na direktnim granama do u). Tok 2 može se oduzeti (pozajmiti) od $f(w, u)$. Time se postiže zadovoljenje uslova (2) (zakona očuvanja toka) za čvor u, jer je u imao povećanje toka za 2 iz povećavajućeg puta, a zatim smanjenje dotoka za 2 iz povratne grane. U čvoru w je sada izlazni tok smanjen za 2, pa ga treba povećati kroz neku izlaznu granu. Sa "proterivanjem" toka može se nastaviti na isti način od w, povećavanjem toka kroz direktne grane i smanjivanjem toka kroz povratne grane. U ovom slučaju postoji samo još jedna direktna grana (w, t) koja dostiže t, i problem je rešen. Pošto samo direktne grane mogu da izlaze iz s, odnosno da ulaze u t, ukupan tok je povećan.



Slika 6: Primer mreže sa povećavajućim putem.

Povećanje je jednako manjem od sledeća dva broja: minimalnog sleka direktnih grana, odnosno minimalnog toka povratnih grana. Na slici 7 prikazana je ista mreža sa promenjenim tokom; ispostavlja se da je novi tok u stvari optimalan.



Slika 7: Rezultat povećavanja toka u mreži sa slike 6.

Iz rečenog sledi da ako u mreži postoji povećavajući put, onda tok nije optimalan. Obrnuto je takođe tačno.

Teorema[Teorema o povećavajućem putu]: Tok kroz transportnu mrežu je optimalan ako i samo ako u odnosu na njega ne postoji povećavajući put.

Dokaz: Dokaz u jednom smeru smo već videli — ako u mreži postoji povećavajući put, onda tok nije optimalan. Pretpostavimo sada da u odnosu na tok f ne postoji ni jedan povećavajući put, i dokažimo da je tada f optimalni tok. Za proizvoljan presek (određen skupom A, $s \in A$, $t \in B \equiv V \setminus A$) definišemo kapacitet, kao zbir kapaciteta njegovih grana koje vode iz nekog čvora skupa A u neki čvor skupa B. Jasno je da ni jedan tok ne može biti veći od kapaciteta proizvoljnog preseka. Zaista, ukupan tok iz s jednak je zbiru tokova kroz grane

preseka od A ka B, umanjenom za tok kroz grane preseka od B ka A, pa je manji ili jednak od zbira kapaciteta grana koje vode od A ka B, odnosno od kapaciteta preseka. Prema tome, ako pronađemo tok sa vrednošću koja je jednaka kapacitetu nekog preseka, onda je taj tok optimalan. Sa dokazom nastavljamo u tom pravcu: pokazaćemo da ako u odnosu na tok ne postoji povećavajući put, onda je ukupan tok jednak kapacitetu nekog preseka, pa dakle optimalan.

Neka je f tok u odnosu na koji ne postoji povećavajući put. Neka je $A \subset V$ skup čvorova v takvih da u odnosu na tok f postoji povećavajući put od s do v (preciznije, postoji put od s do v takav da na njemu za sve direktne grane e važi $f(e) < c(e)$, a za sve povratne grane e' važi $f(e') > 0$). Jasno je da $s \in A$ i $t \notin A$, jer po pretpostavci za f ne postoji povećavajući put. Prema tome, A definiše presek. Tvrdimo da za sve grane (v, w) tog preseka važi $f(v, w) = c(v, w)$ ako je $v \in A$, $w \in B$ ("direktne" grane), odnosno $f(v, w) = 0$ ako je $v \in B$, $w \in A$ ("povratne grane"). Zaista, u protivnom bi direktna grana (v, w) produžavala povećavajući put do čvora $w \notin A$, suprotno pretpostavci da takav put postoji samo do čvorova iz A. Slično, povratna grana (v, w) produžavala bi povećavajući put do čvora $v \notin A$. Dakle, ukupan tok jednak je kapacitetu preseka određenog skupom A, pa je tok f optimalan.

Dokazali smo sledeću važnu teoremu.

Teorema[Teorema o maksimalnom toku i minimalnom preseku]: Optimalni tok u mreži jednak je minimalnom kapacitetu preseka.

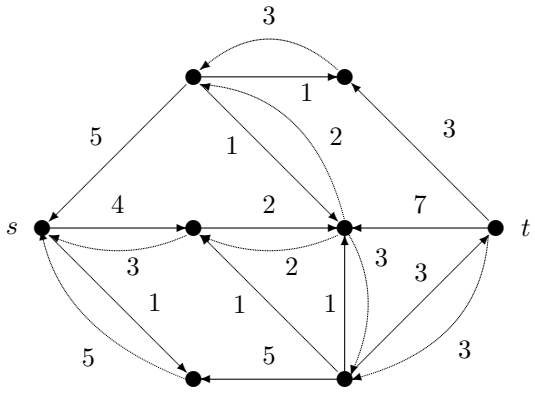
Teorema o povećavajućem putu ima za posledicu i sledeću teoremu.

Teorema[Teorema o celobrojnom toku]: Ako su kapaciteti svih grana u mreži celobrojni, onda postoji optimalni tok sa celobrojnom vrednošću.

Dokaz: Tvrđenje je posledica teoreme o povećavajućem putu. Svaki algoritam koji koristi samo povećavajuće puteve dovodi do celobrojnog toka ako su svi kapaciteti grana celobrojni. Ovo je očigledno, jer se može krenuti od toka 0, a onda se ukupan tok posle svake upotrebe povećavajućeg puta povećava za celi broj. Do istog zaključka dolazi se i na drugi način: kapacitet svakog preseka je celobrojan, pa i minimalnog.

Teorema o povećavajućem putu neposredno se transformiše u algoritam. Polazi se od toka 0, traže se povećavajući putevi, i na osnovu njih povećava se tok, sve do trenutka kad povećavajući putevi više ne postoje. Traženje povećavajućih puteva može se izvesti na sledeći način. Definišemo *rezidualni graf* u odnosu na mrežu $G = (V, E)$ i tok f, kao mrežu $R = (V, F)$ sa istim čvorovima, istim izvorom i ponorom, ali promenjenim skupom grana i njihovih težina. Svaku granu $e = (v, w)$ sa tokom $f(e)$ zamenjujemo sa najviše dve grane $e' = (v, w)$ (ako je $f(e) < c(e)$; kapacitet e' jednak je sleku grane e: $c(e') = c(e) - f(e)$), odnosno $e'' = (w, v)$ (ako je $f(e) > 0$; kapacitet e'' je $c(e'') = f(e)$). Ako se na ovaj način dobiju dve paralelne grane, zamenjuju se jednom, sa kapacitetom jednakom zbiru kapaciteta paralelnih grana. Na slici 8 prikazan je rezidualni graf za mrežu sa

slike 6, u odnosu na tok zadat na toj slici. Grane rezidualnog grafa odgovaraju mogućim granama povećavajućeg puta. Njihovi kapaciteti odgovaraju mogućem povećanju toka kroz te grane. Prema tome, povećavajući put je običan usmereni put od s do t u rezidualnom grafu. Konstrukcija rezidualnog grafa zahteva $O(|E|)$ koraka, jer se svaka grana proverava tačno jednom. Ovaj algoritam naziva se Ford-Fulkersonov algoritam.



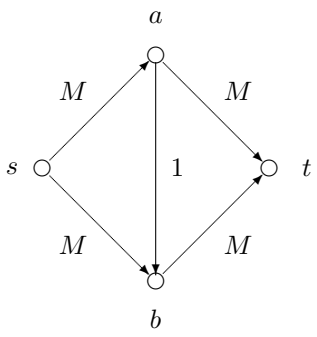
Slika 8: Rezidualni graf mreže sa slike 6 u odnosu na tok definisan na toj slici.

Na nesreću, izbor povećavajućeg puta na proizvoljan način može se pokazati vrlo neefikasnim. Vreme izvršenja takvog algoritma u najgorem slučaju može da čak i ne zavisi od veličine grafa. Posmatrajmo mrežu na slici 9. Optimalni tok je očigledno $2M$. Međutim, mogli bismo da krenemo od povećavajućeg puta $s - a - b - t$ kroz koji se tok može povećati samo za 1. Zatim bismo mogli da izaberemo povećavajući put $s - b - a - t$ koji opet povećava tok samo za 1. Proces može da se ponovi ukupno $2M$ puta, gde M može biti vrlo veliko, bez obzira što graf ima samo četiri čvora i pet grana.

Primetimo da Ford-Fulkersonov algoritam ne precizira način na koji se dolazi do povećavajućeg puta: njih možemo naći korišćenjem DFS ili BFS pretrage. Ako su svi kapaciteti u mreži celobrojni, onda se za svaki povećavajući put tok kroz mrežu povećava barem za 1. Stoga je složenost Ford-Fulkersonovog algoritma $O(|E| \cdot T)$, gde je sa T označen maksimalni tok kroz mrežu. U slučaju racionalnih vrednosti kapaciteta, algoritam se zaustavalja, ali složenost nije ograničena.

Gore navedena mogućnost je vrlo nepoželjna, ali se može izbeći. Edmonds i Karp su 1972. godine pokazali da ako se među mogućim povećavajućim putevima uvek bira onaj sa najmanjim brojem grana, onda je broj povećavanja najviše $O(|V| \cdot |E|)$, odnosno to vodi algoritmu koji je u najgorem slučaju složenosti $O(|V| \cdot |E|^2)$. Dokaz se zasniva na dvema činjenicama.

U odnosu na rezidualni graf, rastojanje svakog čvora od izvora se nikada ne smanjuje tokom algoritma, već se samo može povećati (pod rastojanjem se podrazumeva broj grana na najkraćem putu). Prilikom nalaženja povećavajućeg



Slika 9: Primer mreže na kojoj traženje povećavajućih puteva može biti neograničeno neefikasno.

puta, nekim od grana su u potpunosti iskorišćeni kapaciteti te se te grane više ne nalaze u rezidualnom grafu. Stoga ako su neki najkraći putevi koristili tu granu, oni su se potencijalno povećali.

Pored toga, prilikom pronalaženja povećavajućeg puta, bar jedna grana na tom putu imaće u potpunosti ispunjen kapacitet. Zvaćemo takvu granu kritičnom. Interesuje nas koliko puta jedna grana (u, v) može biti kritična. Nakon prve runde, ona nestaje iz rezidualnog grafa, pa ako je ponovo bila kritična moralo je da se desi da je pronađen povećavajući put kroz granu (v, u). Međutim, s obzirom na to da uvek tražimo najkraći povećavajući put, to znači da najkraći put do čvora u ide kroz čvor v. U prvoj rundi najkraći put je išao kroz granu (u, v), odnosno važilo je da je $d(v) \geq d(u) + 1$. Nakon toga rastojanje čvora v se nije smanjilo, pa sada najkraći put do u ima dužinu bar za jedan veću od rastojanja čvora v koje je bilo bar za 1 veće prethodne vrednosti $d(u)$. Stoga se rastojanje čvora u povećalo bar za 2.

Na osnovu ovoga važi da ako je grana dva puta bila kritična, to znači da se rastojanje njenog početka od izvora u međuvremenu povećalo. Očigledno ovo rastojanje nije nikada veće od $|V| - 1$, te s obzirom da se rastojanja ne smanjuju, svaka grana može postati kritična najviše $O(|V|)$ puta. Stoga postoji najviše $O(|V||E|)$ trenutaka kada neka grana postaje kritična a važi da u svakoj fazi algoritma bar jedna grana postaje kritična. Stoga je broj faza najviše $O(|V||E|)$, a svaka traje $O(|E|)$.

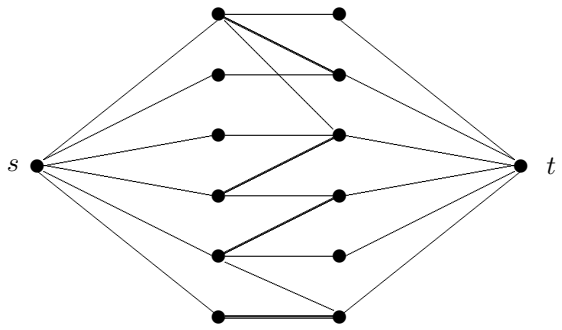
```
#include<iostream>
#include<vector>
#include<map>
#include<queue>
#include <limits>

using namespace std;
```

```
// graf koji odgovara mrezi, izvor je cvor 0, a ponor cvor 7
vector<vector<pair<int,int>>> listaSuseda {{{1,5},{2,7},{3,6}},
                {{4,4}, {5,3}}, {{5,4},{6,1}}, {{6,5}}, {{7,3}},
        {{7,7}}, {{5,4},{7,6}}, {}};

bool bfs(int rezidualniGraf[8][8], int izvor, int ponor, vector<int> &roditelj) {
  int brojCvorova = listaSuseda.size();
  vector<bool> posecen(brojCvorova, false);
  queue<int> red;
  // izvor dodajemo u red i postavljamo da je posecen
  red.push(izvor);
  posecen[izvor] = true;
  roditelj[izvor] = -1;

  while (!red.empty()) {
    int cvor = red.front();
    red.pop();
    for (int sused=0; sused<brojCvorova; sused++){
      if (!posecen[sused] && rezidualniGraf[cvor][sused]>0){
        red.push(sused);
        roditelj[sused] = cvor;
        posecen[sused] = true;
      }
    }
  }
  // ako smo BFS obilaskom stigli od izvora do ponora, vracamo true
  // inace vracamo false
  return (posecen[ponor] == true);
}

void edmondsKarp(int izvor, int ponor){

  int brojCvorova = listaSuseda.size();

  int rezidualniGraf[8][8];
  for(int i=0; i<brojCvorova; i++)
    for(int j=0; j<brojCvorova; j++)
      rezidualniGraf[i][j]=0;

  // inicijalizujemo matricu povezanosti rezidualnog grafa
  // na matricu povezanosti polaznog grafa
  for(int i=0; i<brojCvorova; i++){
    for(int j=0; j<listaSuseda[i].size(); j++){
      rezidualniGraf[i][listaSuseda[i][j].first]=listaSuseda[i][j].second;
    }
  }
```

```
cout << "Polazni rezidualni graf je: " << endl;
for(int i=0; i<brojCvorova; i++){
  for(int j=0; j<brojCvorova; j++)
    cout << rezidualniGraf[i][j] << " ";
  cout << endl;
}

vector<int> roditelj(brojCvorova);
int maxTok = 0;

// sve dok postoji put od izvora do ponora
while(bfs(rezidualniGraf,izvor,ponor,roditelj)){

  // pronalazimo minimalni kapacitet grana duz puta
  int tok = numeric_limits<int>::max();

  // vracamo se unazad od ponora ka izvoru nizom roditeljskih cvorova
  for (int cvor=ponor; cvor!=izvor; cvor=roditelj[cvor]){
    int u = roditelj[cvor];
    // azuriramo vrednost minimalnog kapaciteta na putu
    if (rezidualniGraf[u][cvor]<tok)
      tok = rezidualniGraf[u][cvor];
  }

  // azuriramo kapacitete u rezidualnom grafu
  for (int cvor=ponor; cvor!=izvor; cvor=roditelj[cvor]){

    int u = roditelj[cvor];
    rezidualniGraf[u][cvor]-=tok;
    rezidualniGraf[cvor][u]+=tok;

  }

  // povecavamo vrednost maksimalnog toka
  maxTok+=tok;
  cout << "Tok je povecan za " << tok << endl;
  cout << "Rezidualni graf je sada" << endl;
  for(int i=0; i<brojCvorova; i++){
    for(int j=0; j<brojCvorova; j++)
       cout << rezidualniGraf[i][j] << " ";
    cout << endl;
  }

}
```

```
  cout << "Optimalni tok ima vrednost " << maxTok << endl;
}

int main(){
  edmondsKarp(0,7);
  return 0;
}
```

Pokazaćemo sad da se problem bipartitnog uparivanja može svesti na problem mrežnog protoka. Zadatom bipartitnom grafu $G = (V, E, U)$ u kome treba pronaći uparivanje sa najvećim mogućim brojem grana (optimalno uparivanje) dodajemo dva nova čvora s i t, povezujemo s granama sa svim čvorovima iz V, a sve čvorove iz U povezujemo sa t. Označimo dobijeni graf sa G' (videti sliku 10, na kojoj su sve grane usmerene sleva udesno). Pošto svim granama dodelimo kapacitet 1, dobijamo regularan problem transportne mreže na grafu G'. Neka je M neko uparivanje u G. Uparivanju M može se na prirodan način pridružiti tok u G'. Dodeljujemo tok 1 svim granama iz M i svim granama koje s ili t povezuju sa uparenim čvorovima. Svim ostalim granama dodeljujemo tok 0. Ukupan tok jednak je tada broju grana u uparivanju M.



Slika 10: Svođenje bipartitnog uparivanja na optimizaciju transportne mreže (sve grane usmerene su sleva udesno).

Može se pokazati da je M optimalno uparivanje ako i samo ako je odgovarajući *celobrojni* tok u G' optimalan. U jednom smeru dokaz je jednostavan: ako je tok optimalan i odgovara uparivanju, onda se ne može naći veće uparivanje, jer bi mu odgovarao veći tok.

Dokažimo sada i drugi smer. Jasno je da svaki alternirajući put u G odgovara povećavajućem putu u G', i obrnuto. Ako je M optimalno uparivanje, onda za njega ne postoji alternirajući put, pa u G' ne postoji povećavajući put, a odgovarajući tok je optimalan.

Ako postoji optimalni celobrojni tok, on mora da odgovara uparivanju, jer je svaki čvor u V povezan samo jednom granom (sa kapacitetom 1) sa s; zbog toga, ukupan tok kroz svaki čvor iz V može da bude najviše 1. Isto važi i za čvorove

iz skupa U. Ovo uparivanje mora biti optimalno, jer ako bi se moglo povećati, onda bi postojao veći ukupni tok.