

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET



Katarina M. Dimitrijević

ALGORITMI ZA ODREĐIVANJE
MINIMALNOG POVEZUJUĆEG DRVETA U
GRAFU

master rad

Beograd, 2025.

Mentor:

dr Vesna MARINKOVIĆ, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

prof. dr Filip MARIĆ, redovan profesor
Univerzitet u Beogradu, Matematički fakultet

dr Mirko SPASIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Mami, tati i bratu

Naslov master rada: Algoritmi za određivanje minimalnog povezujućeg drveta u grafu

Rezime:

Ključne reči: graf, algoritam, drvo, složenost, minimalnost

Sadržaj

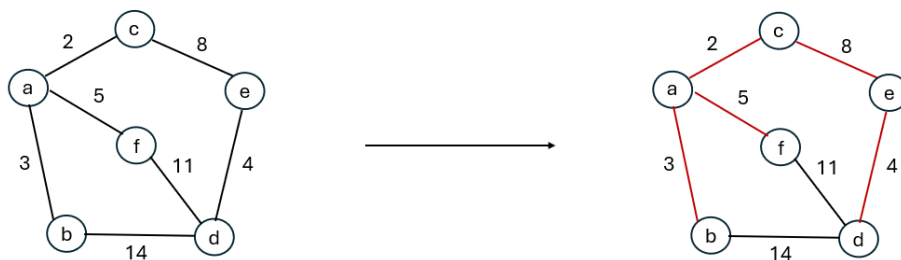
1	Uvod	1
2	Pregled algoritama za konstrukciju minimalnog povezujućeg drveta	5
2.1	Primov algoritam	5
2.2	Kruskalov algoritam	8
2.3	Boruvkin algoritam	10
2.4	Algoritam obrnutog brisanja grana	13
2.5	KKT algoritam	16
2.6	Algoritam Fredmana i Tarjana	22
2.7	Gabov algoritam paketa	27
2.8	Šazelov algoritam	30
	Literatura	38
	Bibliografija	39

Glava 1

Uvod

Grafovski algoritmi predstavljaju značajnu podoblast teorije algoritama. Jedan od fundamentalnih problema teorije grafova, koji se smatra osnovom kombinatorne optimizacije, je problem pronalaženja minimalnog povezujućeg drвета (eng. *Minimum Spanning Tree*) u neusmerenom težinskom grafu. U ostatku teksta će za minimalno povezujuće drvo često biti korišćena skraćenica MST.

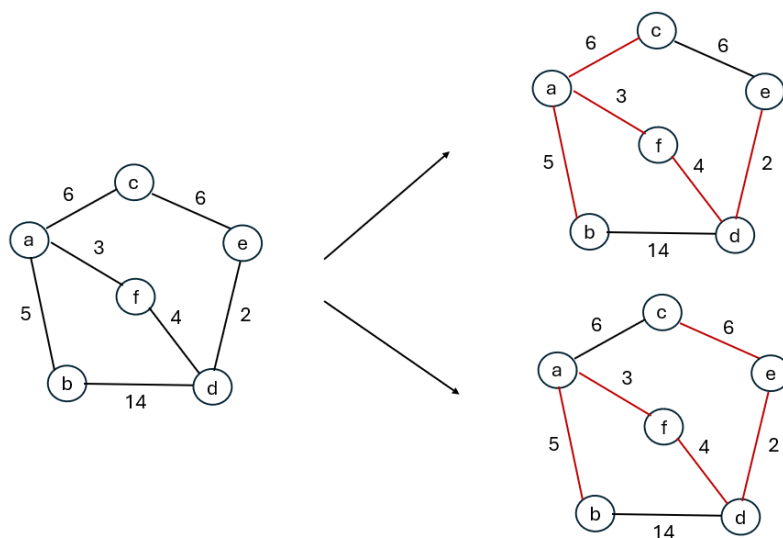
Minimalno povezujuće (razapinjuće) drvo neusmerenog težinskog grafa predstavlja povezan podgraf zadanog grafa koji sadrži sve njegove čvorove, a čija je ukupna cena grana minimalna. Na slici 1.1 je prikazan primer neusmerenog težinskog povezanog grafa i njegovog minimalnog povezujućeg drвета [10].



Slika 1.1: Primer grafa i njegovog minimalnog povezujućeg drвета

U slučaju nepovezanih težinskih grafova za svaku od komponenti se može pronaći minimalno povezujuće drvo. Skup svih minimalnih povezujućih drвета nepovezanog težinskog grafa naziva se minimalna povezujuća šuma (eng. *Minimum Spanning Forest*). U nastavku će biti referisana uz pomoć skraćenice MSF.

Ukoliko su težine svih grana u grafu međusobno različite, rešenje problema konstrukcije MST je uvek jedinstveno. U suprotnom to ne mora biti slučaj. Primer povezanog težinskog grafa, kod koga se težine svih grana međusobno razlikuju, i njegovo jedinstveno rešenje ilustrovani su na slici 1.1. Na slici 1.2 je dat primer grafa kod koga postoje dve grane iste težine. U zavisnosti od toga koja od njih je u datom trenutku izabrana moguća su dva rešenja.



Slika 1.2: Primer povezanog težinskog grafa koji ima dve grane iste težine i njegova dva različita minimalna povezujuća drvet

Poput većine važnih algoritama, i algoritmi za rešavanje ovog problema, motivisani su problemima iz prakse. Ovaj problem datira još od davnina. Naime, ljudi su od davnina pokušavali da najpre minimizuju putanje i rastojanja između geografskih tačaka, naselja ili drugih objekata u prostoru, a kasnije i troškove vodovodnih, električnih, telekomunikacionih i drugih mreža. Formulacija problema konstrukcije MST predstavlja uopštenje svega navedenog. Prva istraživanja potiču još s početka dvadesetog veka. Naime, češki matematičar Boruvka (Otakar Borůvka) je, inspirisan tada aktuelnim problemom koji se odnosio na dizajn električne mreže, u regionu Južne Moravske u Češkoj, objavio rad 1926. godine koji se bavio ovom temom [1]. Tada je po prvi put formulisan problem konstrukcije MST u matematičkim terminima (odnosno njegovo uopštenje). Interesantno je da se rad sastojao od samo jedne strane, ali se, bez obzira na to, smatra ključnim za rešavanje problema konstrukcije MST. Originalna formulacija problema, data u ovom radu, glasi: Dato je n tačaka

u ravni ili u prostoru čija su sva međusobna rastojanja različita. Cilj je napraviti mrežu koja povezuje ove tačke tako da važi:

- svake dve tačke su povezane ili direktno ili posredstvom drugih tačaka
- ukupna dužine mreže je najmanja moguća [11].

Više detalja o ovom algoritmu biće dato u poglavlju 2.3. Kasnije je objavljeno još nekoliko radova na temu ovog algoritma, nezavisno od Boruvkinog istraživanja. Inspirisan Boruvkinim radom [11], Jarník (Vojtěch Jarník) je 1930. godine objavio nešto drugačiji algoritam za rešavanje ovog problema [7]. Taj isti algoritam je objavio i Prim (Robert Prim) 1957. godine, a potom i Dijkstra (Edsger Dijkstra) 1958. godine. U literaturi ovaj algoritam je poznat kao Primov algoritam. Kruskal (Joseph Kruskal) je, 1956. godine [9] objavio algoritam zasnovan na dosta intuitivnoj ideji koristeći potpuno drugačiji pristup u odnosu na Primov. Svi do sada spomenuti algoritmi su složenosti $O(|E| \log |V|)$, gde je $|E|$ broj grana, a $|V|$ broj čvorova grafa. Naredni značajan pomak u ovoj oblasti, a i samoj teoriji grafova postigli su Fredman (Michael Fredman) i Tarjan (Robert Tarjan) 1984. godine [6] kada su konstruisali algoritam složenosti $O(|E| \log^* |V|)$ ¹ koristeći novu strukturu podataka – Fibonačijev hip. Sledeće veliko dostignuće postigao je Karger (David Karger) 1995. godine [8] sa randomizovanim algoritmom očekivane linearne vremenske složenosti. Međutim kako ovo nije bio deterministički algoritam, potraga za algoritmom manje vremenske složenosti je nastavljena. Godine 1997, Šazel (Bernard Chazelle) [2] predstavio je deterministički algoritam vremenske složenosti $O(|E| \alpha(\log |V|))$, gde je sa α označena inverzna Akermanova funkcija. Ova funkcija raste izuzetno sporo, sporije čak i od iterirane logaritamske funkcije. Ipak, ona i dalje teži beskonačnosti kada n teži beskonačnosti, tako da ovime i dalje nije pronađen deterministički algoritam linearne složenosti. Pored spomenutih, postoji još veliki broj algoritama koji predstavljaju varijacije ovih, uz manje vremenske uštede, ali za problem MST još uvek nije poznato da li postoji deterministički algoritam linearne vremenske složenosti.

Problem određivanja minimalnog povezujućeg drveta nalazi primene u različitim domenima. Jedna od tipičnih primena ovog problema je u dizajnu različitih vrsta komunikacionih mreža, kada je potrebno međusobno povezati čvorove mreže tako

¹Ovde \log^* označava iteriranu logaritamsku funkciju i predstavlja broj puta koliko je potrebno primeniti logaritamsku funkciju dok argument ne postane manji od 1.

da se ne obrazuje ciklus, a da cena mreže bude minimalno moguća. Pored komunikacionih, ovaj problem nalazi primenu i u dizajnu računarskih mreža, električnih i vodovodnih, kao i transportnih mreža.

Koristi se u problemu klasterovanja, kada je cilj grupisati slične tačke tako da se minimizuje ukupno rastojanje između klastera (kod hijerarhijskog tipa klasterovanja). Ideja je da se za zadati skup tačaka konstruiše MST korišćenjem nekog poznatog algoritma. MST se smatra jednim klasterom koji sadrži sve čvorove, a zatim se jedna po jedna grana uklanja (počevši od grane najveće težine) i ostaju grupisani elementi. Ovo je klasterovanje naniže, a uz pomoć MST se može postići i klasterovanje naviše.

Takođe, nalazi primenu i u razvijanju približnih algoritama za rešavanje nekih NP-teških problema, na primer, problema trgovačkog putnika (eng. *Traveling Salesman Problem*). Zadatak problema trgovačkog putnika je da se u datom grafu pronađe Hamiltonov ciklus sa minimalnom ukupnom cenom grana. MST se može iskoristiti za rešavanje jedne varijante problema trgovačkog putnika, kada težine grana odgovaraju euklidskim rastojanjima između čvorova. Dakle, potrebno je da trgovac obiđe n datih gradova, koje odgovaraju tačkama u ravni, tako da svaki od njih poseti tačno jednom a da pritom pređe što kraći put. Pritom se podrazumeva da su udaljenosti između svaka dva grada poznate i da odgovaraju rastojanju odgovarajućih tačaka u ravni. MST se u ovom algoritmu koristi za izgradnju približnog algoritma sa faktorom aproksimacije 2. Naime, udvostručavanjem grana MST grafa i skraćivanjem ponovljenih čvorova korišćenjem nejednakosti trougla dolazi se do obilaska grafa koji je u najgorem slučaju 2 puta duži od optimalnog rešenja problema TSP. Ovo rešenje je moguće dodatno popraviti.

Cilj ovog rada je implementacija i analiza različitih algoritama za rešavanje problema minimalnog povezujućeg drveta. Obrađeni algoritmi biće međusobno poređeni na testnom uzorku odabranih grafova.

Glava 2

Pregled algoritama za konstrukciju minimalnog povezujućeg drveta

2.1 Primov algoritam

Primov algoritam je osmišljen 1930. godine kao matematički algoritam u okviru teorije grafova, a 1957. godine je objavljen rad iz oblasti informatike u kome je, po drugi put, predstavljen isti algoritam.

Kako je minimalno povezujuće drvo određeno skupom grana datog grafa, prirodno se nameće ideja da se ono može konstruisati dodavanjem jedne po jedne grane. Naredna grana se bira tako da bude najmanje težine od svih grana koje povezuju do tada konstruisano drvo sa čvorovima koji još uvek nisu dodati u tekuće drvo. Postupak se završava kada se svi čvorovi dodaju u drvo. Dokaz korektnosti opisanog algoritma se može jednostavno sprovesti uz pomoć indukcije po broju grana dodatih u trenutni podgraf minimalnog povezujućeg drveta [10].

Induktivna hipoteza: Za povezan neusmeren težinski graf G , zadat skupom čvorova V i skupom grana E , u oznaci $G = (V, E)$, može se konstruisati drvo T_k koje sadrži k grana, $k < |V| - 1$, tako da je ono podgraf nekog MST-a zadatog grafa.

Baza indukcije: Za bazu indukcije se uzima slučaj kada je $k=0$, tada trenutno drvo T_0 ne sadrži nijednu granu i može sadržati proizvoljan čvor drveta. Pošto je zadati graf povezan, minimalno povezujuće drvo uvek postoji i sadrži sve čvorove zadatog grafa, te je baza indukcije ispunjena.

Induktivni korak: Pretpostavimo da imamo na raspolaganju drvo T_k koje zadovoljava induktivnu hipotezu i koje se sastoji od k grana. Postavlja se pitanje kako pronaći narednu granu minimalnog povezujućeg drveta koja bi proširila induktivnu

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

hipotezu.

Neka je G_k skup koji sadrži sve grane (a, b) , tako da čvor a pripada T_k , a čvor b ne pripada T_k . Ideja je da se za narednu granu bira grana minimalne težine iz skupa G_k . Neka je grana minimalne težine grana e . Tvrdimo da svaka grana, izabrana na ovaj način, sigurno pripada nekom minimalnom povezujućem drvetu. Odnosno, drugim rečima, tvrdimo da postoji minimalno povezujuće drvo T' koje sadrži T_k i granu e .

Dokaz: Neka je T minimalno povezujuće drvo grafa G . Ukoliko grana e pripada drvetu T grafa G , u tom slučaju je $T = T'$. Ukoliko drvo T ne sadrži granu $e = (a, b)$ dokaz je malo komplikovaniji. Naime, pošto je T povezujuće drvo, mora postojati put od a do b kroz grane minimalnog povezujućeg drveta i on mora biti jedinstven, jer bi u suprotnom postojao ciklus. Takođe, zaključujemo da grana e ne sme pripadati putu od a do b u T . Postoji čvor a pripada, a čvor b ne pripada drvetu T_k , na tom putu mora postojati bar još jedna grana $e' = (a', b')$, tako da se čvor a' nalazi u T_k , a čvor b' ne. Postavlja se pitanje odnosa težina grana e i e' :

- Grana e' ne može biti manje težine od grane e jer je e odabrano kao grana minimalne težine iz skupa G_k .
- Međutim e' ne može biti ni veće težine od e jer bi se u tom slučaju, uklanjanjem grane e' iz drveta T i dodavanjem grane e , dobilo drvo manje ukupne cene od T , što bi dovelo do kontradikcije.
- Na osnovu prethodnog razmatranja jasno se zaključuje da u ovom slučaju grane e i e' moraju imati iste težine. Na taj način uklanjanjem grane e' i dodavanjem grane e u drvo T , ono ostaje minimalno povezujuće drvo grafa G . Odnosno, postoji *minimalno* drvo T koji *sadrži* granu e .

Primov algoritam polazi od praznog skupa grana i u svakoj iteraciji dodaje po jednu granu čime se drvo proširuje. Do svakog čvora, iz skupa čvorova koji još uvek ne pripadaju G_k , pamti se minimalna cena grane od čvorova iz G_k , a ukoliko takva grana ne postoji smatra se da težina iznosi $+\infty$. Nakon odabira naredne grane, proveravaju se cene grana od njenog krajnjeg čvora do svih susednih čvorova i ukoliko je cena neke od tih grana manja od cene tekuće minimalne grane do tog čvora, koja je u prethodnim iteracijama izračunata, minimalna tekuća cena grane do tog čvora se ažurira. Opisani postupak je skoro pa identičan Dajkstrinom algoritmu za računanje najkraćih puteva iz zadatog čvora, izuzev što se kod Dajkstrinog algoritma

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

putanje minimizuju po ukupnoj ceni od početnog čvora, a kod Primovog od trenutno konstruisanog drveta.

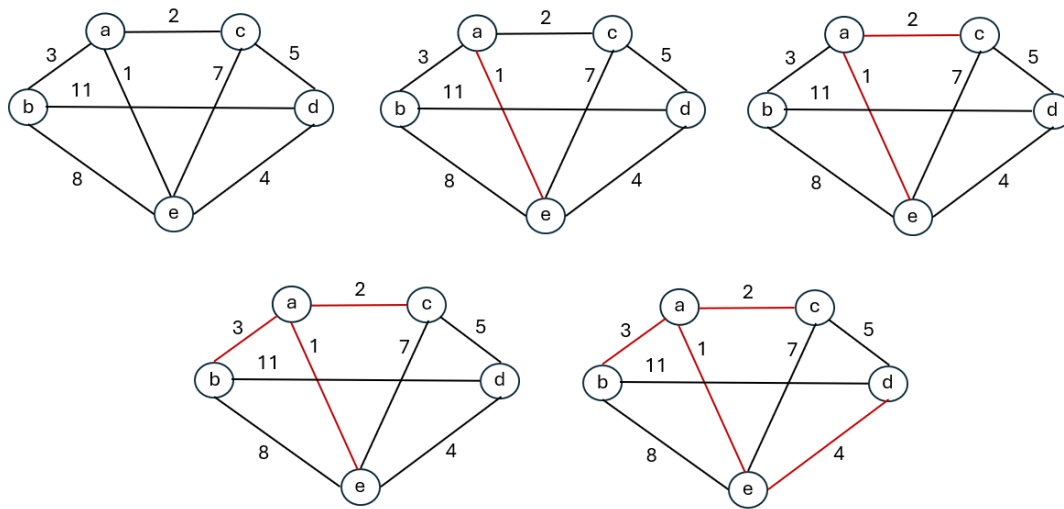
Analiza složenosti: U implementaciji Primovog algoritma koristi se red sa prioritetom. Najčešće se za implementaciju reda sa prioritetom koristi binarni hip, koji je drvolika struktura podataka i kod koga operacije dodavanja proizvoljnog elementa i uklanjanja elementa sa najvećim prioritetom imaju u najgorem slučaju logaritamsku složenost. Kako se u korenu binarnog min-hipa čuva minimalni element, pristup minimalnom elementu se izvršava u konstantnom vremenu. Ova struktura podataka se koristi za čuvanje težina grana do svih suseda trenutnog drveta T_k , tako da se u svakom trenutku može efikasno pronaći grana minimalne težine koja se dodaje u trenutno minimalno povezujuće drvo. Svaki od čvorova se najviše jednom dodaje u red sa prioritetom, pa je vreme potrebno za dodavanje svih čvorova, kao i vreme potrebno za brisanje iz reda $O(|V| \log |V|)$. Tokom konstrukcije minimalnog povezujućeg drveta grane najmanje težine do svakog od čvorova van T_k se potencijalno ažuriraju, međutim tih operacija je ukupno najviše $|E|$, a vreme koje je potrebno za sva ažuriranja iznosi $O(|V| \log |V|)$. Stoga je složenost Primovog algoritma ekvivalentna složenosti Dajkstrinog i iznosi $O((|E| + |V|) \log |V|)$.

Umesto binarnog, može se koristiti neki drugi hip, npr. Fibonačijev hip i time se može poboljšati teorijska vremenska složenost. Ovakva poboljšanja postaju značajna u slučaju veoma velikih grafova. Primov algoritam se smatra prilično efikasnim algoritmom, naročito za retke grafove (one koji imaju mali broj grana) i tada složenost iznosi $O(|V| \log |V|)$.

Na slici 2.1 je ilustovan rad Primovog algoritma, korak po korak. Minimalno povezujuće drvo za zadati graf sa slike čine grane: (a, e) , (a, c) , (a, b) i (e, d) , a njegova ukupna cena iznosi 11. Postupak čuvanja cena grana do svih čvorova i njihovo ažuriranje kroz iteracije može se videti u tabeli 2.1. Prvi element i -te vrste tabele odgovara čvoru koji je dodat u i -toj iteraciji, dok ostali elementi te vrste odgovaraju cenama minimalnih, do sada razmatranih, grana između njega i čvorova koji još uvek ne pripadaju konstruisanom drvetu. Vrednost beskonačno označava da put između neka dva čvora još uvek nije poznat.

Primov algoritam je primer *pohlepnog* algoritma, što znači da uvek bira trenutno najbolje rešenje, tj. lokalni minimum, koji vodi ka najboljem rešenju kompletnog

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA



Slika 2.1: Primer izvršavanja Primovog algoritma

	a	b	c	d	e
a	-	$a(3)$	$a(2)$	∞	$a(1)$
b	-	-	$a(2)$	$b(11)$	$a(1)$
c	-	-	-	$e(4)$	$a(1)$
d	-	-	-	-	$a(1)$
e	-	-	-	-	-

Tabela 2.1: Cene minimalne grane do svakog od čvorova van T_k kroz iteracije

problema.

2.2 Kruskalov algoritam

Kruskal je 1956. godine osmislio algoritam za konstrukciju minimalnog povezujućeg drveta, koji je zasnovan na nešto drugačijoj ideji od Primovog.

Ideja koja stoji iza Kruskalovog algoritma je da se minimalno povezujuće drvo konstruiše dodavanjem jedne po jedne grane na trenutnu šumu, a ne na trenutno drvo, kao što je to slučaj u Primovom algoritmu. Sam algoritam započinje izvršavanje nad skupom čvorova, tako da svaki čvor predstavlja zasebno drvo, koje nema grana. U svakoj iteraciji se razmatra naredna grana iz neopadajuće sortiranog skupa grana, na osnovu njihovih cena. Ukoliko čvorovi, koji su incidenti sa tekućom granom,

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

pripadaju različitim drvetima tekuće šume, ta grana se dodaje u tekuću šumu, dok se u suprotnom preskače. Dodavanjem grane se vrši spajanje drveti kojima pripadaju čvorovi trenutne grane. Na ovaj način se dodavanjem svake grane broj drveti trenutne šume smanjuje za jedan. Lako se zaključuje da se postupak završava nakon dodavanja $|V| - 1$ grane, kada se formira jedinstveno drvo, odnosno podgraf polaznog grafa koji ne sadrži cikluse. Jasno je da ovaj postupak rezultira povezujućim drvetom, samo je još potrebno pokazati da je ono i minimalno [10]. Dokaz se može sprovesti indukcijom po broju obrađenih grana iz neopadajuće sortirano niza grana na osnovu njihovih težina.

Induktivna hipoteza: Za zadati graf G i skup njegovih grana E (koje su neopadajuće sortirane po težinama), umemo da formiramo šumu K koja sadrži $k - 1$ grana iz skupa E , tako da K pripada minimalnom povezujućem drvetu T . Grane se biraju redom iz neopadajuće sortirano skupa grana, pod uslovom da ne formiraju ciklus. Drugim rečima, postoji minimalno povezujuće drvo T koje sadrži sve grane iz K , a ne sadrži nijednu granu iz $E \setminus K$ (one koje su odbačene jer zatvaraju neki ciklus).

Baza indukcije: Za bazni slučaj se uzima da je $k = 0$, odnosno da nijedna grana još uvek nije obrađena, tako da je trenutna šuma $K = \emptyset$, ali je i skup odbačenih grana takođe prazan.

Induktivni korak: Ako je na osnovu induktivne hipoteze poznato kako kreirati tekuću šumu koja se sastoji od $k - 1$ grana, postavlja se pitanje kako na nju dodati narednu granu tako da induktivna hipoteza ostane ispunjena. Ako je sledeća grana koja se razmatra grana e , moguća su tri slučaja:

- Grana e može biti odbačena. Do odbacivanja dolazi ukoliko ona formira neki ciklus sa granama iz K . Pošto se sve grane iz K nalaze i u T , sledi da bi grana e formirala ciklus i u T . Kako u drvetu T nema ciklusa, grana e ne pripada T , tako da odbacivanjem grane e induktivna hipoteza ostaje ispunjena.
- Ukoliko grana e ne formira ciklus u K , ona se dodaje. U slučaju da se razmatrana grana nalazi u drvetu T induktivna hipoteza ostaje ispunjena i dokaz je završen.
- Ukoliko grana e ne formira ciklus u K , ona se dodaje, ali se može desiti da se ova grana ne nalazi u T . Kako se ona ne nalazi u T , znači da bi se njenim dodavanjem formirao neki ciklus. U tom ciklusu mora postojati neka grana e' , koja se ne nalazi u K . Pošto se ona nalazi u T , znači da se ne nalazi u skupu odbačenih grana. Razmotrimo njenu težinu. Kako ona nije još uvek razmatrana

od strane algoritma, pa samim tim nije dodata u K , njena težina mora biti veća od težine grane e . Međutim, ako je veća onda se njenim izbacivanjem iz T i dodavanjem grane e dobija drvo manje težine što je kontradikcija jer je T već minimalno. Zaključujemo da su težine grana e i e' zapravo jednake. Na ovaj način zamenom grane e' granom e u drvetu T , ono ostaje minimalno i induktivna hipoteza ostaje ispunjena.

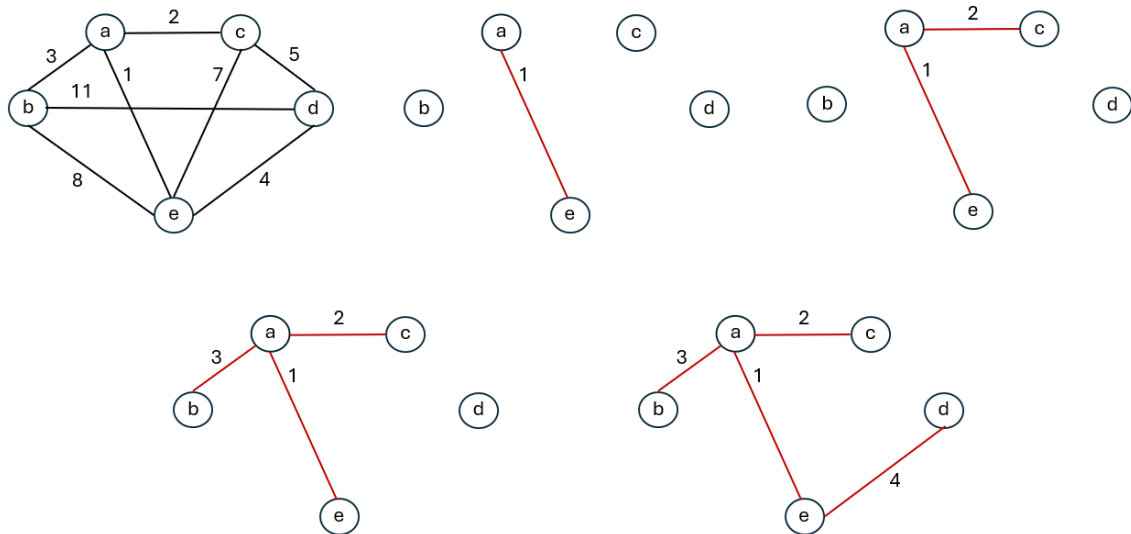
Analiza složenosti: u implementaciji ovog algoritma se za spajanje dva drveta, prilikom dodavanja grane, kao i za proveru da li krajnji čvorovi grana pripadaju istom ili različitim drvetima, može iskoristi efikasna namenska struktura DSU (eng. *Disjoint Set Union*), poznatija kao *union-find* [10]. Ova struktura podataka omogućava uniranje dva podskupa i određivanje kom podskupu pripada neki element u vremenskoj složenosti koja je logaritamska u odnosu na broj elemenata u skupu. Glavna petlja algoritma se izvršava $|E|$ puta. U petlji se pozivaju operacije pronalaženje predstavnika (eng. *find*) i spajanje drveta (eng. *union*) koje su složenosti $O(\log |V|)$. Takođe, na samom početku je potrebno inicijalizovati ovu pomoćnu strukturu što je vremenske složenosti $O(|V|)$, jer na samom početku svaki čvor predstavlja komponentu za sebe. Takođe, potrebno je izdvojiti i sortirati sve grane, što je u proseku složenosti $O(|E| \log |E|)$. Sumiranjem svega navedenog i korišćenjem poznate nejednakosti $|E| \leq |V|^2$, dolazimo do ukupne složenosti $O(|E| \log |V|)$. Slično kao i Primov, smatra se efikasnim algoritmom, ali naročito za retke grafove. Kruskalov algoritam je takođe primer *pohlepnog algoritma*.

Na slici 2.2 je prikazan primer izvršavanja Kruskalovog algoritma koji na ulazu dobija graf sa slike. Rad algoritma započinje nad skupom čvorova V , gde svaki od njih čini drvo za sebe i zajedno formiraju trenutnu šumu $\{a\}$, $\{b\}$, $\{c\}$ i $\{d\}$ i $\{e\}$. Uzima se jedna po jedna grana iz sortiranog niza i uključuje se samo ukoliko ne zatvara ciklus. Tekuću šumu nakon prve iteracije čine drveti sa skupom čvorova: $\{a, e\}$, $\{b\}$, $\{c\}$ i $\{d\}$, nakon druge: $\{a, c, e\}$, $\{b\}$ i $\{d\}$ i nakon treće $\{a, b, c, e\}$ i $\{d\}$. Na samom kraju formira se jedinstveno drvo sa skupom čvorova $\{a, b, c, d, e\}$ i skupom grana $\{(a, b), (a, c), (a, e), (d, e)\}$, odnosno, algoritam se zaustavlja kada se konstruiše drvo koje sadrži $|V| - 1$ granu.

2.3 Boruvkin algoritam

Još jedan algoritam za pronalaženje minimalnog povezujućeg drveta zadatog grafa, nastao 1926. godine, je Boruvkin algoritam. Ideja na kojoj se zasniva je veoma

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA



Slika 2.2: Primer izvršavanja Kruskalovog algoritma

intuitivna i jednostavna. Polazi se od skupa čvorova ulaznog grafa, gde se svaki od njih posmatra kao zasebna komponenta povezanosti, odnosno, drvo, kao i kod Kruskalovog algoritma. Ideja je da se te komponente spajaju dok ne ostane samo jedna, koja će zapravo predstavljati traženo minimalno povezujuće drvo početnog grafa i tada se algoritam zaustavlja. Ono što se zapravo razlikuje u odnosu na Kruskalov algoritam je način odabira grana koje ulaze u minimalno povezujuće drvo, odnosno sama konstrukcije drveta.

U prvom koraku se za svaki čvor iz skupa V pronalazi grana najmanje cene koja je incidentna sa tim čvorom i ona se dodaje na trenutnu šumu. Ovime se broj komponenti smanjuje. Ovaj korak se ponavlja, s tim što se nakon prve iteracije više ne biraju grane koje povezuju pojedinačne čvorove, već grane koje povezuju novonastale komponente. Postupak se zaustavlja kada ostane samo jedna komponenta povezanosti. Drugim rečima, u svakom koraku se bira grana koja povezuje dve različite komponente povezanosti, čime se garantuje da neće doći do zatvaranja ciklusa, što je osnovni uslov za konstruisanje povezujućeg drveta grafa. Rezultat prethodno opisanog postupka je povezujuće drvo inicijalnog grafa. Ovo tvrđenje se jednostavno može dokazati indukcijom po broju komponenti povezanosti u trenutnoj šumi.

Ostaje pokazati da je ovo drvo ujedno i minimalno, odnosno drvo najmanje ukupne cene.

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

Dokaz: Pokažimo da je povezujuće drvo dobijeno opisanim postupkom minimalno svođenjem na kontradikciju. Neka je T minimalno povezujuće drvo zadanog grafa, a T' povezujuće drvo koje je konstruisao Boruvkin algoritam. Ako su sve cene grana grafa međusobno različite, tvrdimo da mora da važi $T = T'$. Pretpostavimo suprotno, tj. da se ova dva drveta razlikuju i neka je $e' = (a, b) \in T'$ prva grana drveta T' dobijena Boruvkinim algoritmom kao grana minimalne težine iz čvora a , za koju važi da ne pripada minimalnom povezujućem drvetu T . Kako je T povezujuće drvo, postoji put od čvora a do čvora b kroz grane drveta T . Taj put mora da sadrži granu $e = (a, v)$ za $v \neq b$. Ova grana ne može biti manje težine od grane (a, b) jer je grana (a, b) izabrana u Boruvkinom algoritmu kao grana najmanje težine iz a . Stoga je težina grane e veća od težine grane e' . Međutim, onda važi da je i $T - e + e'$ manje ukupne cene od cene drveta T , što je u suprotnosti sa tim da je T MST grafa [11].

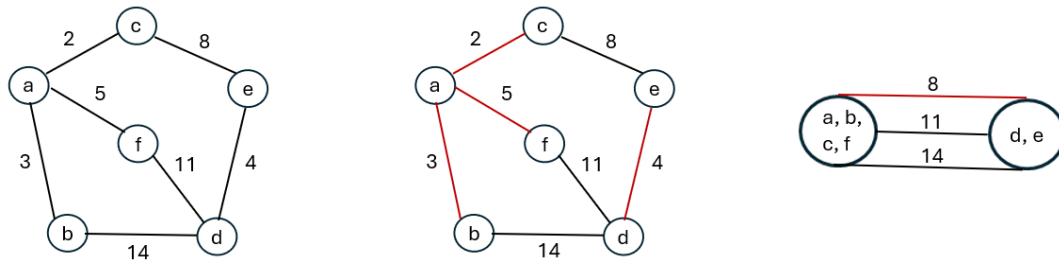
Prethodni dokaz važi u slučaju kada su sve grane zadanog grafa međusobno različite cene što je Boruvka i pretpostavio u svom prvom radu iz 1926. godine. Ukoliko bi graf mogao imati grane sa istim težinama, dokaz bi bio malo komplikovaniji, a primeri takvih dokaza dati su u poglavljima o Primovom i Kruskalovom algoritmu. Kada je nastao, ovaj algoritam je bio poznat pod nazivom *širenje šume* (eng. *parallel merging, forest growing*). Kao i prethodna dva algoritma, ovo je takođe primer *pohlepnog algoritma*.

U implementaciji ovog algoritma se novoformirana komponenta identifikuje svojim predstavnikom, a sve grane unutar jedne komponente se eliminišu. Takođe, prilikom spajanja komponenti granom najmanje cene, sve ostale grane između njih mogu biti eliminisane. Za potrebe ovih operacija se koristi efikasna, već ranije spomenuta, union-find struktura.

Analiza složenosti: U svakoj iteraciji se na trenutnu šumu dodaje bar $|V|/2$ grana, čime se broj komponenti smanjuje bar dvostruko u odnosu na početni graf, te je broj iteracija algoritma $O(\log |V|)$. U svakoj od tih iteracija se svaka grana razmatra najviše jednom. Dakle, ukupna složenost Boruvkinog algoritma iznosi $O(|E| \log |V|)$. Iako ima istu složenost kao i Primov i Kruskalov algoritam, za razliku od njih, pokazuje se znatno efikasnijim na gustim grafovima. Takođe, njegova velika prednost je mogućnost paralelizacije.

Na slici 2.3 je prikazan primer izvršavanja Boruvkinog algoritma. U prvoj iteraciji se svaki čvor grafa iz skupa $V = \{a, b, c, d, e, f\}$ posmatra kao zasebna komponenta. Za svaki od njih se pronalazi njemu incidentna grana najmanje cene tako da ga ona povezuje sa drugom komponentom. Pregled takvih grana za svaku od komponenti,

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA



Slika 2.3: Primer izvršavanja Boruvkinog algoritma

Komponenta	Grana	Cena
$\{a\}$	(a, c)	2
$\{b\}$	(a, b)	3
$\{c\}$	(a, c)	2
$\{d\}$	(d, e)	4
$\{e\}$	(d, e)	4
$\{f\}$	(a, f)	5

Tabela 2.2: Grane obeležene u prvoj iteraciji

Komponenta	Grana	Cena
$\{a, b, c, f\}$	(c, e)	8
$\{d, e\}$	(c, e)	8

Tabela 2.3: Grane obeležene u drugoj iteraciji

nakon završene prve iteracije, dat je u tabeli 2.2.

Ove grane se označavaju i tako se formiraju nove komponente koje sadrže njima incidentne čvorove. Primenom operacija unije nastaju nove komponente i one sada postaju ulaz u drugu iteraciju algoritma. Postupak se ponavlja. Grane koje su obeležene u drugoj iteraciji su date u tabeli 2.3.

Obeležavanjem poslednje grane nastaje jedinstvena komponenta i algoritam se zaustavlja. Prolaskom kroz označene grane se konstruiše minimalno povezujuće drvo polaznog grafa. Njega čine grane: (a, c) , (a, b) , (a, f) , (c, e) i (a, c) , ukupne cene 22.

2.4 Algoritam obrnutog brisanja grana

Još jedan algoritam iz grupe pohlepnih algoritama za pronalaženje minimalnog povezujućeg drveta je algoritam obrnutog brisanja grana (eng. *delete-reverse*) algo-

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

ritam. Algoritam na ulazu dobija neusmeren, težinski graf. Ideja algoritma je da se inicijalno grane sortiraju u opadajućem poretku (u opštem slučaju u nerastućem poretku), a da se zatim za svaku od njih razmatra da li nakon njenog uklanjanja graf ostaje povezan. Ukoliko je to slučaj, trenutna grana se uklanja i algoritam nastavlja dalje sa izvršavanjem. U protivnom se ta grana zadržava i prelazi se na sledeću granu iz sortiranog niza. Ideja algoritma je jednostavna i intuitivna i u velikoj meri odgovara ideji Kruskalovog algoritma ¹. Razlika je u tome što se u Kruskalovom algoritmu grane sortiraju u rastući (neopadajući) poredak, razmatraju jedna po jedna i dodaju na trenutnu šumu, osim ukoliko formiraju ciklus.

Iako se na osnovu samog opisa algoritma lako može zaključiti da on zaista vraća minimalno povezujuće drvo, u nastavku sledi dokaz. Prvi deo dokaza, odnosno tvrđenje da algoritam obrnutog brisanja grana nalazi povezujuće drvo je veoma jednostavno pokazati. Algoritam na ulazu dobija povezani, težinski graf G . Pošto je graf povezan, njegovo povezujuće drvo mora postojati. Uklanjanjem jedne po jedne grane, uz uslov da graf sve vreme ostaje povezan, na kraju ostaje graf koji sadrži inicijalni skup čvorova, ali iz koga su uklonjene sve grane koje formiraju neki ciklus, što je po definiciji povezujuće drvo grafa. Drugi deo dokaza je malo komplikovaniji i odnosi se na minimalnost ovako dobijenog drveta. Ovaj deo tvrđenja se može pokazati primenom matematičke indukcije.

Induktivna hipoteza: Prepostavimo da F , podgraf polaznog grafa G koji je dobijen obradom $k - 1$ grane sa najvećom težinom iz G , sadrži neko njegovo minimalno povezujuće drvo T .

Baza indukcije: Ovo tvrđenje je tačno pre prve iteracije algoritma jer je inicijalno $F = G$, a kako je to povezan graf, kao takav uvek sadrži minimalno povezujuće drvo T .

Induktivni korak: Neka je F podgraf grafa G koji je nastao nakon $k - 1$ iteracije algoritma. Neka je grana e k -ta po redu grana sortiranog niza. Moguća su tri slučaja:

- Grana e se ne izbacuje iz F . Pošto je na osnovu induktivne hipoteze F sadržalo neko MST T , kako se trenutna grana se ne izbacuje iz F , sledi da induktivna hipoteza ostaje nenarušena.
- Grana e se izbacuje iz F i ne pripada T . Ukoliko se grana ne nalazi u T , ona

¹I algoritam obrnutog brisanja grana i Kruskalov algoritam su objavljeni u istom radu 1956. godine, ali ih ne treba poistovetiti [9].

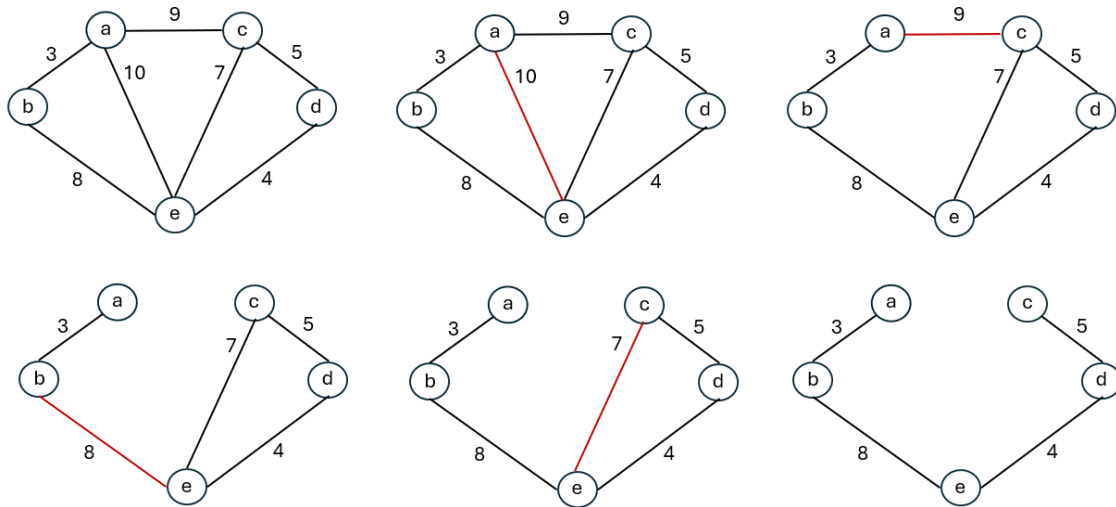
se može izbaciti iz F i time se ne narušava induktivna hipoteza jer i nakon izbacivanja trenutne grane, F i dalje sadrži neko MST T .

- Grana e se izbacuje iz F , ali pripada T . Pošto je ova grana izbačena, može se lako zaključiti da njenim izbacivanjem graf G ostaje povezan. Neka je $e = (u, v)$. U tom slučaju mora postojati neki drugi put od čvora u do čvora v . Jasno je da na tom putu postoji neka grana e' iz tog ciklusa koja se nalazi van T , jer u suprotnom T ne bi bilo povezujuće drvo. Ona mora pripadati nekom drugom drvetu T' koje je takođe podgraf od F i da važi $T' = T - e + e'$. Lako je pokazati da je da je T' zaista drvo jer nastalo izbacivanjem grane e iz drveta T i dodavanjem nove grane e' , tako da se na taj način zadržavaju svojstva drveta. Sada možemo takođe pokazati i da je drvo T' minimalno. Posmatrajmo cene grana e i e' . Lako se pokazuje da nije moguće $w(e) > w(e')$ jer je T minimalno povezujuće drvo. Takođe, nije moguće ni $w(e) < w(e')$ jer algoritam uklanja grane u opadajućem redosledu što znači da bi onda grana e' bila razmatrana pre grane e i ovo dovodi do kontradikcije u razmatranom slučaju. Iz navedenog sledi da mora važiti da ove dve grane imaju jednake cene, odnosno da je $T' = T$, što znači da u svakoj iteraciji važi da F sadrži minimalno povezujuće drvo. Na samom kraju izvršavanja, sve grane koje učestvuju u nekom ciklusu su izbačene, što znači da je samo F drvo, odnosno, samo F je minimalno povezujuće drvo.

Analiza složenosti: vreme potrebno za sortiranje grana grafa je $O(|E| \log |E|)$. Kako bismo utvrdili da li bi nakon izbacivanja svake od grana graf ostao povezan, potrebno je na preostalom grafu pokrenuti neki vid pretrage (npr. DFS pretraga) čija složenost iznosi $O(|E| + |V|)$. Dakle, ukupna vremenska složenost iznosi: $O(|E| \log |E| + |E|(|E| + |V|))$.

Na slici 2.4 je ilustrovano izvršavanje algoritma. Grane se razmatraju jedna po jedna u opadajuće sortiranom redosledu težina (crvenom bojom su obeležene grane koje se trenutno razmatraju) i uklanjaju se sve one koje ulaze u sastav nekog ciklusa. Najpre se razmatra grana (a, e) , a zatim i grana (a, c) . Za obe se zaključuje da graf nakon njihovog uklanjanja ostaje povezan, te one ne pripadaju MST. Naredna grana koja se razmatra je grana (b, e) , međutim, njenim brisanjem bi graf postao nepovezan tako da se ona preskače. Zatim se uklanja i grana (c, e) . Sledeća je na redu (b, e) , ali ni ona, niti ijedna od preostalih grana se ne smeju ukloniti jer nijedna od njih ne ulazi u sastav nekog ciklusa. Drugim rečima, grane koje su preostale čine minimalno

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA



Slika 2.4: Primer izvršavanja algoritma obrnutog brisanja grana

povezujuće drvo.

2.5 KKT algoritam

Pokazuje se da je moguće razviti i efikasniji algoritam za računanje MST ukoliko dozvolimo da algoritam koristi koncept nasumičnosti. Karger (David Karger) je 1992. godine objavio *randomizovani* algoritam za konstrukciju MST i čija očekivana vremenska složenost iznosi $O(|E| + |V| \log |V|)$. Klajn (Philip Klein) i Tardžan (Robert Tarjan) su ga 1994. godine optimizovali i popravili njegovu vremensku složenost na $O(|E| + |V|)$. Kako je originalna ideja potekla od Kargera, algoritam je u literaturi poznat i pod nazivom *Kargerov algoritam*. U nastavku će na ovaj algoritam biti referisano kao na *KKT (Karger-Klajn-Tardžanov) algoritam*. Ovaj algoritam spada u grupu tzv. podeli-pa-vladaj (eng. *divide-and-conquer*) algoritama [5].

Ideja na kojoj je zasnovan KKT algoritam je sledeća: konstruiše se nadgraf MST grafa G , a zatim se iz njega odbacuju one grane koje ne pripadaju MST korišćenjem svojstva ciklusa, koje je objašnjeno u nastavku. U ostatku grafa, koji ne sadrži prethodno odbačene grane za koje je utvrđeno da sigurno ne pripadaju MST grafa G , problem se rekurzivno rešava. Izdvajamo nekoliko bitnih svojstava:

- *Svojsvo ciklusa* (eng. *cycle property*) - za proizvoljan ciklus grafa G važi da

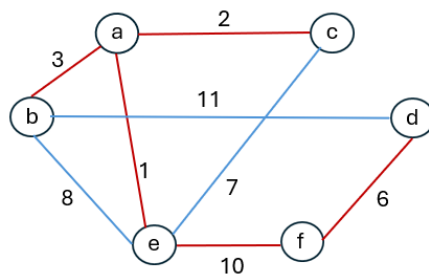
GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

grana najveće težine koja pripada tom ciklusu ne pripada minimalnom povezujućem drvetu tog grafa.

- *Svojsvo preseka* (eng. *cut property*) - za proizvoljni podskup čvorova ulaznog grafa važi da od svih grana koje povezuju taj skup čvorova sa ostatkom grafa, ona koja ima najmanju težinu pripada minimalnom povezujućem drvetu tog grafa.

Za potrebe razmatranja algoritma uvodimo pojam teških i lakih grana u odnosu na neku šumu. Neka je F šuma, tako da je ona podgraf od G . Za granu e , koja pripada G , za koju važi da se njenim dodavanjem u F zatvara neki ciklus i da je ona grana najveće težine u njemu, kažemo da je ona *teška u odnosu na F* , tj. *F -teška* (eng. *F -heavy*). U suprotnom granu e nazivamo *lakom u odnosu na F* (eng. *F -light*), tj. *F -lakom*. Iz ove dve definicije se lako zaključuje da svaka grana koja pripada MST grafa G spada u F -lake, za $F \subseteq MST(G)$. Ovo tvrđenje važi i u suprotnom smeru, odnosno, svaka F -laka grana pripada MST grafa G . Takođe, za F -teške grane važi da su one teške i u odnosu na minimalno povezujuće drvo grafa G .

Na slici 2.5 dat je primer grafa kog koga su crvenom bojom obeležene F -lake grane, a plavom bojom F -teške. Primetimo da skup lakih grana čini MST grafa. Teške grane nisu deo MST grafa i svaka od njih je grana najveće težine u nekom ciklusu trenutnog grafa.



Slika 2.5: Primer grafa na kome su F -lake grane označene crvenom bojom, a F -teške grane plavom bojom

U nastavku sledi opis samog algoritma. Najpre se iz skupa grana $E(G)$ grafa G nasumično odabere podskup grana $E'(G)$, a potom se iz njega uklone sve teške grane. Zatim se na ostatku grana rekurzivno rešava isti problem, ali manje dimen-

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

zije. Ideja koja se krije iza opisanog postupka je sledeća: većina grana u grafu se ne nalazi u MST i ako ih možemo efikasno odbaciti, početni problem se svodi na manji potproblem. Umesto da direktno proveravamo svaku granu, bira se mali uzorak grana na slučajan način i konstruiše MST za taj uzorak grana, što je efikasan proces ukoliko je uzorak mali. Ovaj uzorak koristimo da zaključimo koje se grane sigurno ne nalaze u konačnom MST pošto su označene kao teške. Njih odbacujemo i rekurzivno postupak ponavljamo na preostalom grafu. Preciznije, algoritam je sastavljen od kombinacije koraka Boruvkinog algoritma (eng. *Boruvka steps*) i nasumičnog uzorkovanja (eng. *random sampling*). Boruvkin algoritam se koristi kako bi se broj čvorova grafa najmanje dvostruko smanjio, dok nasumično uzorkovanje redukuje broj grana. KKT algoritam se sastoji iz narednih koraka:

- Inicijalno se nekoliko puta (obično tri puta) pokrene Boruvkin algoritam nad grafom $G(V, E)$, a kao rezultat ovog koraka dobija se graf $G'(V', E')$, tako da važi $V' \leq V/8$ i $E' \leq E$. Ovaj algoritam, kao što je već objašnjeno u poglavlju 2.3, za svaki čvor grafa pronalazi njemu incidentnu granu najmanje težine, a zatim sažima svaka dva čvora povezana ovim granama u jednu komponentu. U ovom koraku algoritma se uklanjaju sve petlje koje su postojale na čvorovima ulaznog grafa i zadržavaju se samo grane najmanje težine među svim granama koje povezuju novonastale komponente povezanosti, dok se sve ostale grane uklanjaju. Na ovaj način se broj čvorova smanjuje bar dvostruko. Ukoliko novonastalo drvo G' sadrži samo jedan jedini čvor, to znači da je rezultat Boruvkine faze ujedno i krajnji rezultat KKT algoritma.
- Na skupu grana E' , bira se uzorak E_1 , nezavisnim uzorkovanjem sa verovatnoćom odabira svake grane od 0.5.
- Algoritam se rekurzivno poziva na novoodabranom skupu grana E_1 čime se kao rezultat dobija minimalna povezujuća šuma $F_1 = MST(V', E_1)$.
- Pronalaze se i uklanjaju sve F_1 -teške grane. Skup preostalih grana označićemo sa E_2 .
- Algoritam se rekurzivno poziva na skupu grana E_2 , čime se kao rezultat dobija minimalna povezujuća šuma $F_2 = MST(V', E_2)$.
- Algoritam vraća uniju skupa grana koje su pronađene u Boruvkinom koraku i grana iz F_2 . Ovaj skup grana čini MST inicijalnog grafa.

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

Dokaz korektnosti algoritma se može sprovesti korišćenjem matematičke indukcije po veličini grafa.

Induktivna hipoteza: Pretpostavimo da KKT algoritam korektno vraća MST za graf G' koji je strogo manjih dimenzija od grafa G .

Baza indukcije: Ukoliko graf G sadrži samo jedan čvor ili ne sadrži nijednu granu, na osnovu induktivne hipoteze, KKT algoritam vraća MST grafa G .

Induktivni korak: Na osnovu svojstva preseka važi da sve grane označene u prvom koraku (Boruvkin algoritam) pripadaju trenutnoj minimalnoj povezujućoj šumi grafa. Boruvkina faza vrši kontrakciju čvorova i vraća graf G' koji je manjih dimenzija od inicijalnog grafa G . Prema svojstvu ciklusa važi da nijedna teška grana ne ulazi u sastav MST originalnog grafa, tako da su sve uklonjene grane van MST. Na osnovu induktivne hipoteze se zaključuje da je minimalna povezujuća šuma redukovano grafa, odnosno grafa manje dimenzije, ispravno pronađena u svim rekurzivnim pozivima, te konačno zaključujemo da algoritam vraća MST grafa.

Vremenska složenost: Očekivano vreme izvršavanja KKT algoritma je linearno po veličini grafa, ali je dokaz ovog tvrđenja nešto komplikovaniji. Za potrebe dokaza biće korišćene leme date u nastavku:

Lema 2.5.1 *Očekivani broj nasumično odabranih grana jednak je polovini ukupnog broja grana, tj. važi $E[E_1] = \frac{|E|}{2}$.*

Dokaz 1 *Dokaz sledi na osnovu svojstva linearnosti matematičkog očekivanja.*

Lema 2.5.2 *Neka je G' podgraf grafa G koji je dobijen uključivanjem svake njegove grane sa verovatnoćom 0.5, a F minimalna povezujuća šuma grafa G' . Očekivani broj F -lakah grana u G nije veći od $2n$, gde je n broj čvorova grafa G , tj važi $E[E_2] \leq 2|V|$.*

Dokaz 2 *Jasno je da se podgraf G' i njegova minimalna povezujuća šuma F konstruišu istovremeno. Za tu konstrukciju se može iskoristiti Kruskalov algoritam. Nakon utvrđivanja da li je grana F -teška ili F -laka, grana se sa verovatnoćom 0.5 dodaje u G' , što se može interpretirati kao bacanje novčića. Ukoliko je grana dodata u G' i proglašena F -lakom granom, ona se dodaje i u minimalnu povezujuću šumu F grafa G . Ovako dobijena šuma jeste minimalna povezujuća jer je dobijena primenom Kruskalovog algoritma na graf G' . Grane koje su proglašene F -teškim u trenutku obrade, ostaju teške do kraja jer se grane naknadno ne uklanjaju iz F . Slično važi i za F -lake grane: one koje su proglašene F -lakim u trenutku obrade, ostaju F -lake, jer sve grane koje su dodate posle njih imaju veću težinu. Jasno je da je prilikom*

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

bacanja novčića poznato da li je, trenutno razmatrana, grana F -teška ili F -laka. Pritom, ishodi bacanja za F -teške grane nisu relevantni jer te grane svakako neće biti dodate u F . Cilj je konstruisati minimalnu povezujuću šumu, a najveći mogući broj grana koji može biti dodat je $|V| - 1$, drugim rečima, potrebno je $|V| - 1$ uspešnih ishoda bacanja novčića u slučaju kada je trenutna grana laka. Ako bismo nastavili da bacamo novčić sve dok ne dobijemo željeni broj uspeha, gde je ukupan broj pozitivnih ishoda određen promenljivom Y , zaključujemo da Y predstavlja slučajnu promenljivu sa negativnom binomnom raspodelom, gde je ukupan broj bacanja jednak $|V|$, a verovatnoća uspeha je $p = 0.5$. Matematičko očekivanje slučajne promenljive sa ovakvom raspodelom je poznato i iznosi $2n$ [8].

Ostalo je još pokazati da je očekivano vreme izvršavanja ovog nedeterminističkog algoritma linearno i iznosi $O(|E| + |V|)$. Označimo očekivano vreme izvršavanja algoritma za graf G sa T_G . Najgore vreme izvršavanja nad svim grafovima G sa n čvorova i m grana se može predstaviti kao:

$$T_{m,n} := \max_{G=(V,E), |V|=n, |E|=m} \{T_G\}$$

Svi koraci, osim samog rekurzivnog poziva, imaju linearno vreme izvršavanja u funkciji broja čvorova i grana grafa, pa je dovoljno pokazati da se i vreme izvršavanja rekurzivnih poziva može odozgo ograničiti sa $c(n + m)$, gde je c konstanta. Neka je vreme izvršavanja algoritma u drugom i četvrtom koraku respektivno T_{G_1} i T_{G_2} , gde je $G_1 = (m_1, n')$ graf koji je dobijen odabirom grana grafa G na slučajan način, a $G_2 = (m_2, n')$ graf dobijen uklanjanjem svih teških grana. Tada važi:

$$T_G \leq c(m + n) + E[T_{G_1} + T_{G_2}] \leq c(m + n) + E[T_{m_1, n'} + T_{m_2, n'}]$$

Pretpostavićemo da važi: $T_{G'} \leq 2c(m' + n')$ za sve manje grafove $T_{G'}$, odnosno da je najgore vreme izvršavanja nad svim manjim grafovima linearno. U nastavku ćemo pokazati da to važi i za graf G [5].

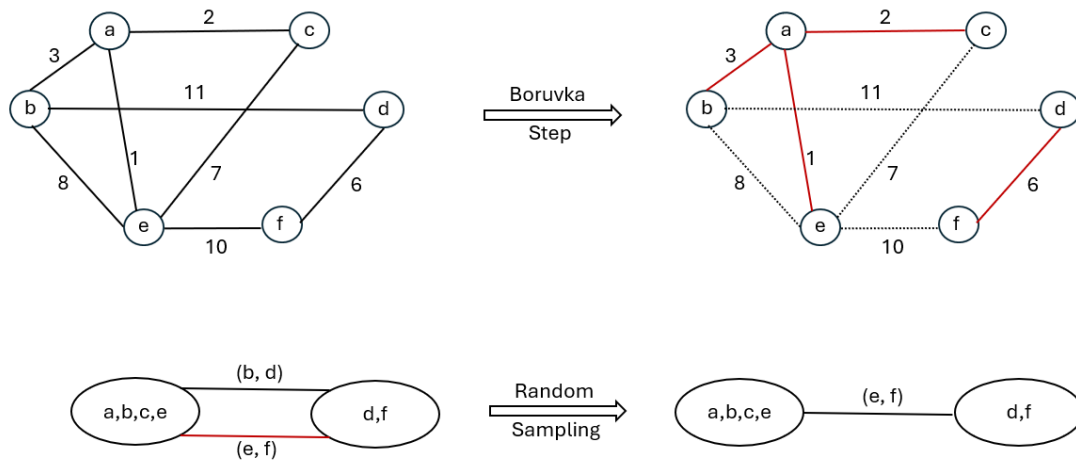
$$\begin{aligned} T_G &\leq c(m + n) + E[2c(m_1 + n')] + E[2c(m_2 + n')] \\ &\leq c(m + n) + c(m' + 2n') + 2c(2n' + n') \\ &= c(m + m' + n + 8n') \\ &\leq 2c(m + n) \end{aligned}$$

Drugi red izvođenja se dobija primenom lema 2.5.1 i 2.5.2. Primenom nejednakosti $n' \leq n/8$ i $m' \leq m$ dolazimo do krajnje nejednakosti. Oдавde sledi da je očekivano vreme izvršavanja ovog stohastičkog algoritma linearno.

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

Istaknimo i to da je ovo prvi algoritam za pronalaženje minimalnog povezujućeg drveta koji je uspešno paralelizovan tako da mu ukupno vreme izvršavanja bude linearno [4].

Na slici 2.6 je dat primer izvršavanja KKT algoritma. Prvi korak algoritma čini redukcija broja čvorova koja se vrši pomoću Boruvkinog algoritma. Nakon prvog koraka dobija se graf koji se sastoji od dve nepovezane komponente: $H_1 = \{a, b, c, e\}$ i $H_2 = \{d, f\}$. One nastaju tako što se za svaki čvor inicijalnog grafa odabere njemu incidenta grana najmanje težine. Grane (b, e) i (c, e) se eliminišu iz daljeg razmatranja pošto grade ciklus unutar komponente H_1 . Nakon uspešne redukcije grafa, sledeći korak je nasumično uzorkovanje grana sa verovatnoćom 0.5. U ovom primeru su preostale samo dve grane: (b, d) i (e, f) , a odabrana je grana (e, f) . Algoritam se rekursivno poziva nad ovom granom i ona se uključuje u krajnji rezultat. Treći korak je pronaći i eliminisati sve teške grane. Jedina takva je grana (b, d) i ona se eliminiše jer zatvara ciklus i predstavlja najtežu granu u njemu. Rezultat ovog postupka je unija grana dobijena u prvom koraku - (a, b) , (a, c) , (a, e) , (d, f) i grane (e, f) . Postupak izvršavanja algoritma je ovde ilustrovan na primeru veoma jednostavnog grafa, iako se u praksi on koristi za pronalaženje MST u znatno većim grafovima.



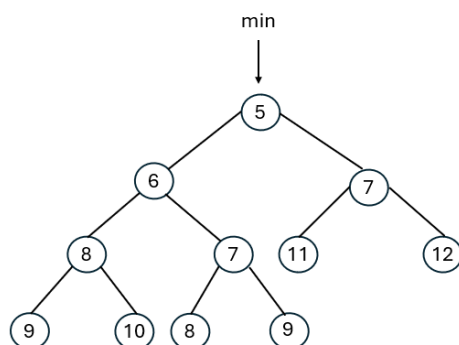
Slika 2.6: Primer izvršavanja KKT algoritma

2.6 Algoritam Fredmana i Tarjana

Fredman (Michael Fredman) i Tardžan su 1987. godine predložili modifikaciju Primovog algoritma, tako da se za implementaciju reda sa prioritetom, koji se koristi za čuvanje susednih grana i brz pristup grani minimalne težine, koristi, umesto binarnog, Fibonačijev hip. Ovim se dolazi do vremenski efikasnijeg algoritma, po cenu nešto složenije implementacije.

2.6.1 Fibonačijev hip

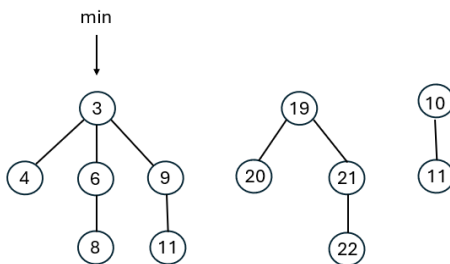
Red sa prioritetom se može implementirati na različite načine. Najčešće korišćen način je korišćenjem binarnog hipa, koji je ilustrovan na slici 2.7. Binarni hip omogućava pristupanje elementu sa najvećim prioritetom u konstantnoj vremenskoj složenosti, a takođe i umetanje proizvoljnog elementa u hip, brisanje minimalnog elementa iz hipa i ažuriranje vrednosti elementa na hipu u logaritamskoj složenosti.



Slika 2.7: Primer binarnog hipa

Fibonačijev hip je optimizovana verzija binarnog. Za razliku od binarnog hipa, Fibonačijevo drvo ne mora biti binarno. Štaviše, hip se ne sastoji od jednog drveta, sada ih može biti više (kao u ilustraciji na slici 2.8). Ovime se sprečava da jedno drvo previše poraste i time naruši ukupnu složenost, ali se istovremeno kontroliše i broj drveta u hipu [6]. Glavna ideja na kojoj je zasnovana sama implementacija ovakve strukture podataka jeste da se operacije izvode „*lenjo*“. U nastavku ćemo pretpostaviti da najveći prioritet ima element sa najmanjom vrednošću.

Pristup minimumu ima istu ideju kao i kod binarnog hipa. Održava se pokazivač na čvor u kome se nalazi najmanja vrednost. Pošto se u korenu svakog od drveta



Slika 2.8: Primer Fibonačijevog hipa

Fibonačijevog hipa nalazi najmanja vrednost iz tog drveta, minimum je zapravo minimum među vrednostima korena svih drveta. I dalje je pristup minimalnom elementu složenosti $O(1)$.

Operacija umetanja elementa se izvodi dosta jednostavnije u odnosu na binarni hip. Čvor se ubacuje u hip tako što se dodaje kao novo drvo koje se sastoji samo iz jednog jedinog čvora. Vreme koje se troši na ovu operaciju je $O(1)$, ali se broj drveta u hipu uvećava za jedan.

Operacija uklanjanja minimalnog elementa sa hipa zahteva najpre da se deca poddrveta kome minimalni element pripada odvoje od roditeljskog čvora i oni postaju drveta za sebe, čime se broj drveta povećava najviše za stepen² drveta, koji je unapred ograničen nekim prirodnim brojem. Kako broj drveta ne bi previše porastao, vrši se spajanje drveta sa istim stepenom, što oduzima dodatno vreme. Pojedinačno izvođenje ove operacije je skupo, ali se deo posla može pripisati operaciji umetanja, pošto je ona sama po sebi veoma brza, čime se dobija znatno bolja amortizovana složenost. Detaljnijom analizom se zaključuje da amortizovana vremenska složenost ove operacije zavisi samo od stepena drveta. Međutim, zahvaljujući svojstvu binomnih drveta [6], dobija se da je složenost ove operacije $O(\log n)$, jer stepen drveta logarimaski zavisi od broja čvorova.

Ažuriranje vrednosti u Fibonačijevom hipu se može veoma jednostavno izvesti u logaritamskom vremenu. Međutim, pokazuje se da je ažuriranje moguće izvesti u konstantnom vremenu. Kako bi se to postiglo, uvodi se ograničenje da svaki čvor može ostati bez najviše jednog deteta, jer se u suprotnom gubi svojstvo binomnih drveta čime se narušava složenost operacije uklanjanja. Iako se ovime delimično kviri struktura binomnih drveta jer se dozvoljava uklanjanje čvorova, a binomna drveta

²Stepen drveta označava maksimalni broj dece koji svaki čvor drveta može da ima.

imaju striktnu strukturu i za svako takvo drvo sa n čvorova važi da stepen korena tog drveta iznosi $\log(n)$, i dalje važi malo slabije svojstvo, a to je svojstvo Fibonačijevih brojeva. Ispostavlja se da i kod Fibonačijevih drveta stepen korena drveta takođe logaritamski zavisi od broja čvorova, ali sa osnovom 1.3, što je svojstvo Fibonačijevih brojeva. Ovime i dalje obezbeđuje logaritamska složenost operacije izbacivanja. Iako je operacija ažuriranja vrednosti sporija kada se izvodi pojedinačno, ona u proseku daje konstantno vreme, tako da amortizovana složenost i ove operacije iznosi $O(1)$.

2.6.2 Opis algoritma

Podsetimo se da je vremenska složenost Primovog algoritma $O((|E|+|V|) \log |V|)$. Ideja Fredmana i Tardžana je da se na neki način logaritamski faktor smanji, kako bi ukupna složenost algoritma težila linearnoj [5]. Ukoliko bi veličina hipa bila mala, postigla bi se manja cena izbacivanja elemenata iz hipa, što bi značajno uticalo na ukupnu složenost algoritma. Primova verzija algoritma konstruiše samo jedno drvo tokom izvršavanja, čuvajući sve susede trenutnog drveta na hipu, tako da je dimenzija hipa ograničena brojem čvorova u grafu. Optimizovana verzija algoritma koristi malo drugačiji pristup kako bi se veličina hipa održavala što manjom, odnosno, u zadatim granicama. Iz ovako implementiranog hipa je moguće izbacivanje elementa u amortizovanoj složenosti $O(\log K)$, gde K predstavlja maksimalnu veličinu hipa, odnosno maksimalni broj susednih čvorova trenutnom drvetu T_i . Cilj je, naravno, postići da K bude značajno manje od $|V|$.

Postavlja se pitanje kako održati veličinu hipa ispod nekog, unapred određenog, praga K , za koji ćemo najpre pretpostaviti da je fiksiran. Fredman i Tardžan su primetili da je to moguće postići kontrolisanjem broja suseda trenutnog drveta na sledeći način: započeti izvršavanje Primovog algoritma i dodavati čvorove na hip sve dok njegova veličina ne dostigne zadati prag. Nakon toga, Primov algoritam se zaustavlja i bira se novi proizvoljni čvor iz kog se opisani postupak iznova započinje i to se ponavlja sve dok se svi čvorovi inicijalnog grafa ne dodaju u tekuću šumu koju čine drvetu T_1, T_2, \dots, T_k . Nakon završetka svake iteracije algoritma, čvorovi koji pripadaju drvetima T_i za svako $i = 1, \dots, k$ kontrahuju se u novi čvor, što je slična ideja kao kod Boruvkinog algoritma. Potom se ova procedura rekursivno poziva nad ostatkom grafa. Drugim rečima, ova procedura se može opisati kao niz iteracija, pri čemu se svaka iteracija može razložiti na tri koraka:

- Na početku svake iteracije su svi čvorovi neoznačeni. Nasumično se bira jedan

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

neoznačen čvor i iz njega se pokreće Primov algoritam za konstrukciju MST. U Fibonačijevom hipu se čuvaju susedni čvorovi trenutnog drveta i time se obezbeđuje efikasan pristup najbližem od njih. Svaki čvor v za koji postoji grana $e(u, v)$, tako da $u \in T$, dok $v \notin T$ smatra se susednim čvorom tekućeg drveta. Susedni čvorovi mogu biti već označeni u prethodnim iteracijama.

- Ukoliko u toku izvršavanja prethodnog koraka veličina hipa premaši unapred zadatu granicu ili se kao susedni, na hip, doda neki već ranije označeni čvor, postupak se zaustavlja. Svi čvorovi koji su dodati u tekuće drvo se označavaju i prethodni korak se ponavlja.
- Ukoliko su svi čvorovi dodati u tekuću šumu, nakon kontrahovanja čvorova, sledi rekursivno izvršavanje algoritma na novonastalom grafu. [12]

Ovako opisan algoritam zadovoljava svojstvo korektnosti. On se oslanja na korektnost Primovog algoritma čijom primenom nastaju drveća trenutne šume. Tako nastale komponente povezanosti se dalje spajaju u rekursivnim pozivima pomoću najlakših susednih grana, koje se skidaju sa hipa.

Analiza složenosti: Probajmo da ocenimo vremensku složenost izvršavanja jedne iteracije algoritma. Označimo broj čvorova grafa sa n , a broj grana sa m . Neka je K maksimalni dozvoljeni broj elemenata hipa. Prilikom konstrukcije drveta, svaka grana se razmatra najviše dva puta, po jednom sa oba kraja, što je složenosti $O(m)$. Prilikom konstrukcije drveta dolazi do redukovanja broja čvorova, dok ne ostane jedna jedina komponenta koja predstavlja MST inicijalnog grafa i ovakvih spajanja je najviše n , koliko i čvorova inicijalnog drveta, dok sam pristup hipu, odnosno operacija uklanjanja elementa sa hipa, ima složenost $O(\log K)$. Iz svega navedenog se može zaključiti da složenost jedne iteracije iznosi $O(m_i + n_i \log K)$, gde su $n_i \leq n$ i $m_i \leq m$ respektivno broj čvorova i grana obrađenih u jednoj iteraciji algoritma. Jasno je da K predstavlja značajan faktor i ispostavlja se da on ne mora biti fiksirana vrednost, već se može uvesti kao parametar čija se vrednost menja iz iteracije u iteraciju. Za svaki označeni čvor koji se dodaje na trenutno drvo T_i , suma stepena svih čvorova koji pripadaju tom drvetu iznosi najmanje K . Čvor se označava u slučaju da njegovim dodavanjem veličina hipa premašuje K ili ukoliko je već bio označen, što znači da već pripada nekom drvetu (ono je već dostiglo maksimalni broj suseda) i tada dolazi do spajanja komponenti čime suma stepena svih čvorova postaje još veća. Ako se broj nastalih drveta u ovoj iteraciji označi sa l , a sa C_1, \dots, C_k

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

označe komponente koje nastaju na kraju ove iteracije algoritma, dolazi se do sledeće jednakosti:

$$2m = \sum_v d_v = \sum_{i=1}^l \sum_{v \in C_i} d_v \geq \sum_{i=1}^l K \geq K \cdot l$$

Iz ovako dobijene jednakosti se može izraziti broj nastalih drveća l u funkciji parametra K . Odnosno, za broj nastalih drveća u i -toj iteraciji algoritma važi:

$$l_i \leq \frac{2m}{K}$$

Broj nastalih komponenti zavisi od parametra K , a kako je poznato da se kao rezultat poslednje iteracije konstruiše MST inicijalnog grafa, odnosno, jedinstvena komponenta koja sadrži sve čvorove, intuitivno je jasno da se broj čvorova kroz iteracije smanjuje. Iz tog razloga se K može birati tako da raste kroz iteracije. Ideja je odabrati vrednost parametra K na pogodan način tako da složenost ostane u okvirima linearne po broju grana. Ako se uzme da je u i -toj iteraciji $K_i = 2^{\frac{2m}{n_i}}$, dolazimo do sledeće jednakosti [12]:

$$O(m_i + n_i \log K_i) = O\left(m_i + n_i \cdot \frac{2m}{n_i}\right) = O(m)$$

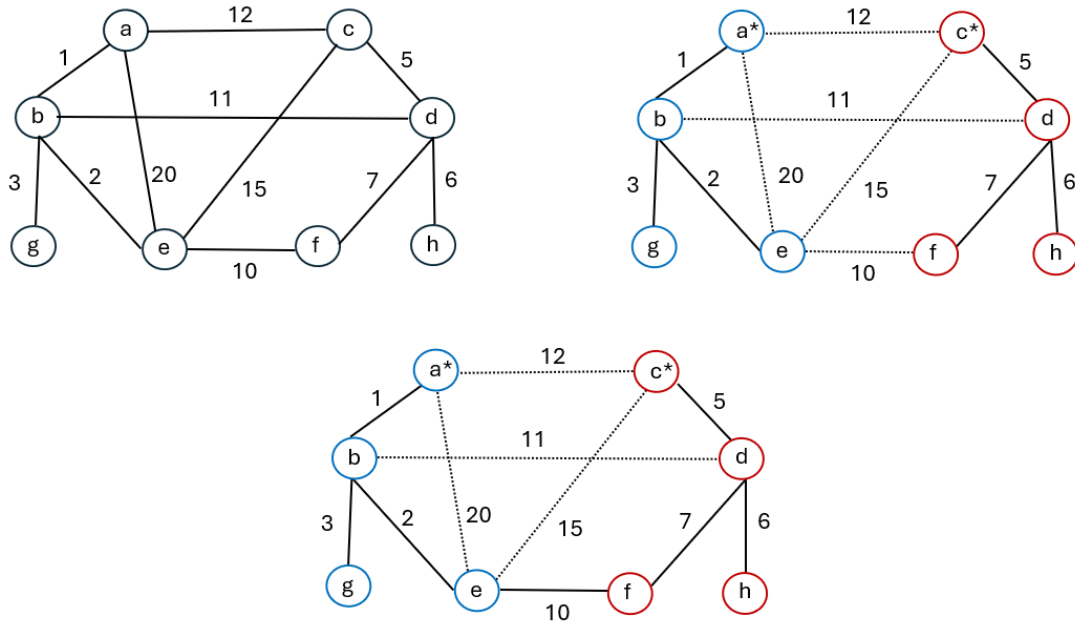
Ovako odabrano K neće pokvariti linearnu složenost, a dodatno, iz predhodne nejednakosti, koristeći da je $l \leq \frac{2m}{K_i}$, gde je l zapravo broj nastalih komponenti na kraju i -te iteracije, odnosno one koje će se koristiti u narednoj, može se zaključiti sledeće:

$$K_i \leq \frac{2m}{n_{i+1}} = \log K_{i+1} \Rightarrow K_{i+1} \geq 2^{K_i}$$

Poslednja nejednakost pokazuje da K ima eksponencijalni rast kroz iteracije, dok n istim tempom opada. Zapravo, pojavljivanje višestrukih eksponenata ukazuje da je rast brži čak i od eksponencijalnog, tzv. *tetracioni*. Broj potrebnih iteracija da K dostigne vrednost n je $\beta(m, n)$, što je funkcija koja izuzetno sporo raste. Nakon $\beta(m, n)$ iteracija, Primov algoritam može da sačuva sve preostale čvorove na hipu, bez da veličina hipa premaši granicu K i formira se minimalno povezujuće drvo inicijalnog grafa. Stoga je vremenska složenost algoritma Fredmana i Tardžana $O(m\beta(m, n))$.

Primer izvršavanja algoritma dat je na slici 2.9, za veličinu hipa $K = 2$. Izvršavanje počinje iz čvora a i on se na početku dodaje na hip. Pošto se kao grane najmanje

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA



Slika 2.9: Primer izvršavanja algoritma Fredmana i Tardžana

cene susedne trenutnom drvetu T_1 pronalaze redom grane (a, b) , (b, e) i (b, g) , čvorovi a, b, g i e se redom dodaju na hip, čime se premašuje zadata veličina hipa. Oni se označavaju, hip se prazni i procedura počinje iznova. Na slici su svi čvorovi koji su dodati u prvoj iteraciji obeleženi plavom bojom i oni formiraju drvo T_1 . Čvor u kome počinje druga iteracija je c . On se dodaje na hip, a potom i najbliže susedne grane (c, d) , (d, f) i (d, h) , odnosno čvorovi c, d, f i h . Ovime se završava druga iteracija, dodati čvorovi se označavaju i hip se ponovo prazni. Na slici su ovi čvorovi označeni crvenom bojom i oni formiraju drvo T_2 . U ovom trenutku se zaključuje da su svi čvorovi dodati u trenutnu šumu i sada se drveta sažimaju u nove čvorove i procedura se poziva rekursivno na novom grafu koji se sastoji iz dva nova čvora i preostalih grana. Bira se ona sa najmanjom cenom i time se algoritam zaustavlja. Dobijeno minimalno povezujuće drvo čine grane: (a, b) , (b, e) , (b, g) , (c, d) , (d, f) , (d, h) i (e, f) .

2.7 Gabov algoritam paketa

Algoritam Fredmana i Tardžana predstavlja značajno unapređenu verziju Primovog algoritma za nalaženje minimalnog povezujućeg drveta grafa. Međutim, moguće je algoritam dodatno optimizovati. Iako prethodno razmatrana verzija ima značajno

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

bolju vremensku složenost zahvaljujući korišćenju Fibonačijevog hipa, operacija ažuriranja minimalnih rastojanja, koja se čuvaju na hipu, i dalje predstavlja najskuplju operaciju, odnosno „usko grlo“ algoritma Fredmana Tardžana.

U Fredman-Tardžanovom algoritmu se svaka grana, kao potencijalni kandidat za dodavanje na trenutnu šumu, čuva na hipu, dok se ne premaši gornja granica veličine hipa. Tada se sve sačuvane grane odbacuju i proces kreće iznova. Kako se neki kandidati tokom iteracija veći broj puta odbacuju, može se desiti da neke od njih dođu na red tek u poslednjoj iteraciji, odnosno dodaju se čak $\beta(m, n)$ puta. U najgorem slučaju takvih grana može biti m , što nije zanemarljiv faktor u izrazu za ukupnu vremensku složenost algoritma. Sve ove grane se stavljaju na hip i uklanjaju sa njega iz istog razloga, da bi se pronašla ona koja ima najmanju cenu među svim susednim granama tekuće šume. Na ovom mestu se javlja ideja da je moguće nekako optimizovati ove upite, tako da se ne moraju sve grane svaki put iznova brisati sa hipa kada se njegova veličina premaši, a potom dodavati kako bi se sačuvala najlakša među njma.

Iako je tačno da je operacija dodavanja u Fibonačijev hip veoma efikasna, odnosno konstantne vremenske složenosti u proseku, ne sme se zaboraviti da se to vreme potroši na svaku dodatnu granu, a zatim se kao korisna uzima samo jedna, ona koja je minimalne cene, te je ukupno vreme uloženo u sve ostale, koje nisu minimalne, nezanemarljivo. Upravo se na ovom mestu javlja ideja da je potencijalno pametniji način čuvati samo minimum, a da se ostale grane, koje nisu u tom trenutku značajne, obrađuju kasnije. Naime, Gabov (Harold Gabow) se dosetio da je put koji vodi optimizaciji algoritma upravo čuvanje samo grane najmanje cene na hipu, dok sve ostale, koje su teže od nje, ne moraju biti trenutno „vidljive“. Ovaj način optimizacije će biti detaljnije opisan u nastavku i pripada klasi algoritama zasnovanoj na tehnici dinamičkog programiranja i obezbeđuje složenost $O(m \log(\beta(m, n)))$.

Ovo je moguće ostvariti tako što se m grana rasporedi u m/p paketa, gde je $p \in \mathbb{N}$ veličina paketa. Grane incidentne jednom čvoru ili klasteru se stavljaju u isti paket, a grane su unutar svakog paketa sortirane prema težinama. Kako ovo sortiranje ne bi značajno uticalo na složenost, p bi trebalo da bude što manje, jer složenost takvog sortiranja iznosi $O(m \log p)$. Sortiranje obezbeđuje da je početni element svakog paketa ujedno i grana minimalne cene, tako da je paket njome određen. Ostatak algoritma ostaje nepromenjen, osim što sada lista susedstva čvorova ne sadrži niz čvorova, već niz paketa, a na hipu se čuvaju paketi koju su određeni

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

ni svojim minimalnim elementima. Sve operacije nad Fibonačijevim hipom ostaju nepromenjene, samo se više ne izvršavaju nad granama, nego nad paketima. Spajanjem drveta tekuće šume spajaju se i liste njihovih paketa, a novi minimum se određuje kao minimum minimuma svih paketa koji se trenutno nalaze na hipu.

Analiza složenosti: Na početku algoritma se grane grafa particionišu u pakete i sortiraju, što je vremenske složenosti $O(m + m \log p)$. U svakoj od iteracija, kojih ima $\beta(m, n)$, vrši se skidanje elemenata sa hipa što je složenosti $O(n_i \log k_i)$ i stavljanje svih paketa na hip što je složenosti $O(2m/p)$. Ne treba zaboraviti da se neke grane brišu sa hipa, a samim tim i iz svojih paketa, ali se dimenzija paketa tokom vremena ne menja. Brisanja sa hipa može biti najviše $O(2m)$ jer se svaka grana može naći u dva paketa, posmatrano sa oba kraja po jednom. Pošto svaki rekurzivni poziv sledi nakon brisanja elemenata sa hipa, takvih poziva je najviše $O(2m)$. Dodatno vreme za jedan rekurzivni poziv je konstantno. Na osnovu prethodne analize sledi da ukupna vremenska složenost Gabovog algoritma iznosi:

$$O \left(m + m \log p + \sum_{i=0}^{maxIter} (n_i \log k_i + 2m/p) + 2m \right)$$

Cilj je postići da vreme izvršavanja jedne iteracije bude $O(m/p)$, umesto $O(m)$. Ovo je moguće postići adekvatnim odabirom k_i tako da važi $n_i \log k_i = O(m/p)$. Ako se dodatno odabere $p = \beta(m, n)$ postiže se sledeće:

$$O \left(m + m \log p + \sum_{i=0}^{p-1} (m/p) + 2m \right) = O(m \log p) = O(m \log \beta(m, n))$$

Pogodno je za vrednost k uzeti $k = 2^{\frac{2m}{pn_i}}$, umesto $k = 2^{\frac{2m}{n_i}}$, jer u tom slučaju potrebno vreme za $n_i \log k_i$ iznosi $O(m/p)$, pa ukupno vreme koje se troši na sumu ostaje linearno u funkciji m .

Kroz analizu prethodnog algoritma dolazi se do važnog zaključka. Naime, od ukupnog broja grana, što je $2m$, bar k_i njih biva dodato na trenutno drvo. Stoga, sledi da je takvih drveta najviše $2m/k_i$. Potrebno je uveriti se da ovo i dalje važi, uprkos manjim izmenama algoritma. Zapravo, sada bar k_i paketa, od ukupno $2m/p$ njih biva dodato na trenutno drvo, tako da se na kraju konstruiše $2m/pk_i$ drveta što je čak i manji broj i ostaje da važi da k eksponencijalno raste, te je broj iteracija ograničen beta funkcijom [12].

Još jedno bitno zapažanje jeste da se veliki deo vremena troši na inicijalno sortiranje elemenata unutar paketa. Međutim, zahvaljujući ovako sortiranim paketima, preostalo potrošeno vreme je linearno u funkciji broja grana $O(m/p)$. Ovo nikako ne znači da se, pošto se kod prethodne implementacije sve grane odbacuju i ponovo dodaju na hip $\beta(m, n)$ puta, sada svaka grana razmatra tačno jednom. Logika je nepromenjena, i dalje je moguće odbaciti grane i razmatrati ih ponovo, ali se sada to radi samo sa predstavnicima paketa, dok se razmatranje ostalih grana odlaže za kasnije. Tako da se u svakoj iteraciji razmatra najviše $2m/p$ elemenata, a ukupno vreme je baš $O(m)$.

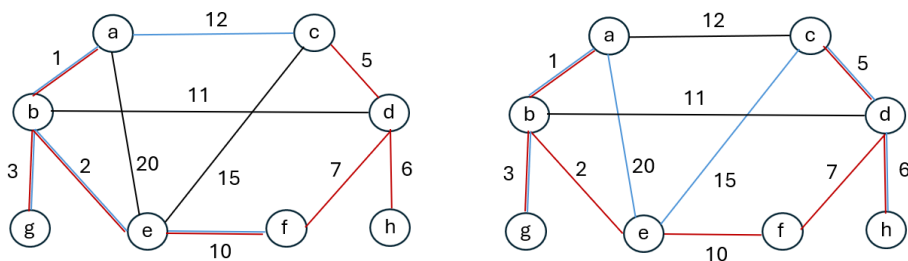
2.8 Šazelov algoritam

Poslednji algoritam za određivanje minimalnog povezujućeg drveta grafa, koji će biti obrađen u ovom radu, objavio je Šazel (Bernard Chazelle) 1999. godine. Ovo je deterministički algoritam najbolje teorijske složenosti, od svih do sada obrađenih, koja iznosi $O(m \alpha(m, n))$, gde n predstavlja broj čvorova, m broj grana, a α funkciju koja izuzetno sporo raste, tzv. inverznu Akermanovu funkciju.

Za razliku od prethodno analiziranih algoritama, koji su konceptualno i idejno jednostavni, ovaj algoritam se zasniva na nešto složenijoj ideji i potpuno drugačijem pristupu. Jedan od ključnih koncepata jeste pojam *kontraktibilnog podgraфа* (eng. *contractible subgraph*). Za podgraf C grafa G kažemo da je kontraktibilan, odnosno da se može kontrahovati u jedan jedini čvor, a da time ne utiče na odabir grana koje će biti dodate u $MST(G)$, ako njegov presek sa $MST(G)$ predstavlja povezan graf. Ovaj koncept je ilustrovan na slici 2.10 gde su crvenom bojom označene grane koje pripadaju MST , dok su plavom označene grane podgraфа za koji se proverava da li je kontraktibilan. Primetimo da je podgraf na slici levo kontraktibilan jer je njegov presek sa $MST(G)$ povezan graf, odnosno sastoji se iz samo jedne komponente povezanosti koja je zadata skupom grana $\{(a, b), (b, g), (b, e), (e, f)\}$. Dok za desni podgraf to ne važi jer njegov presek sa $MST(G)$ predstavlja podgraf od G koji se sastoji od dve komponente povezanosti koje su zadate skupovima grana: $\{(a, b), (b, g)\}$ i $\{(c, d), (d, h)\}$. Ono što kontraktibilan podgraf C čini značajnim jeste ideja da se $MST(G)$ može formirati pomoću $MST(C)$ i $MST(G')$, gde je G' graf koji nastaje od G kontrahovanjem podgraфа C . Detektovanje ovakvih podgraфа je bilo jednostavno u nekima od ranije obrađenih algoritama, ali se taj proces odvijao uporedo sa proširivanjem, odnosno formiranjem minimalnog povezujućeg drveta

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

grafa. Sada se javlja ideja da se takvi podgrafi mogu pronaći ranije i da se kao takvi upotrebe pri konstrukciji MST. Podrazumeva se da taj proces mora biti veoma efikasan. Za potrebe rešenja ovog problema definiše se jedna nova varijanta hipa koja se zove *soft hip* (eng. soft heap), koja pored operacija karakterističnih za standardni hip, omogućava i operaciju brisanja u amortizovanoj gotovo konstantnoj vremenskoj složenosti. Ovo je omogućeno time što se dozvoljava da se elementima hipa mogu u određenoj meri izmeniti, tj. pokvariti vrednosti, kao što će biti prikazano u nastavku teksta [3].

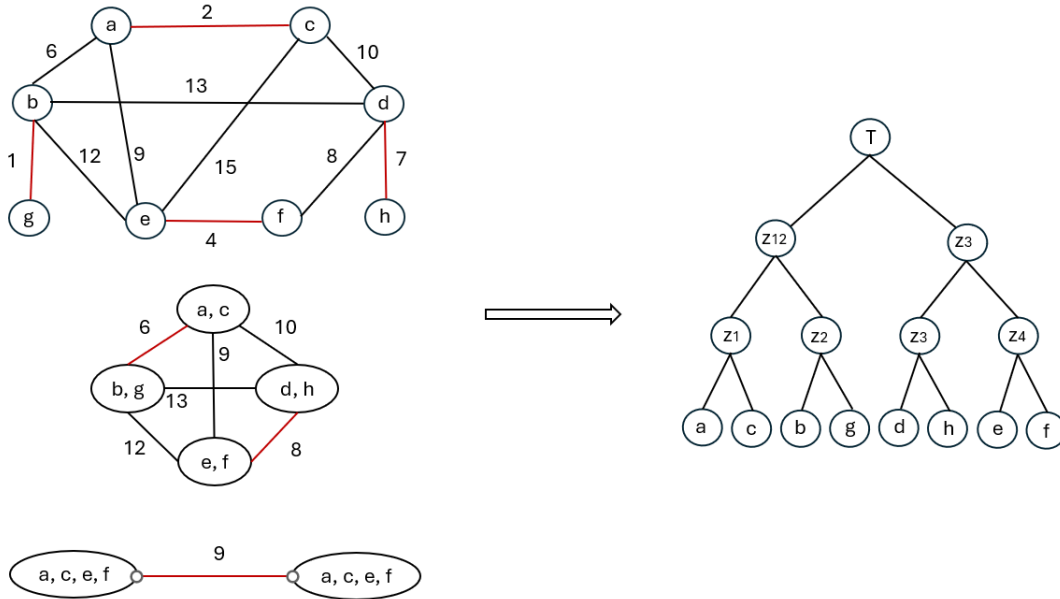


Slika 2.10: Levo je primer podgrafa koji zadovoljava svojstvo kontraktibilnosti, dok podgraf koji se nalazi desno ne zadovoljava to svojstvo.

U nastavku pasusa sledi opis algoritma, dok će svi tehnički detalji biti objašnjeni nešto kasnije. Da bi se pronašlo $MST(G)$, ideja je da se najpre graf podeli na kontraktibilne podgrafe koji su međusobno disjunktni, a potom svaki od njih kontrahuje. Na taj način nastaje novi graf, koji se naziva *minorni graf* (eng. *minor graph*) grafa G . Isti proces se može primeniti na novonastalom grafu i dalje se ponavljati sve dok ne ostane jedan jedini čvor koji predstavlja kontrakciju čitavog, inicijalnog, grafa G . Hijerarhija podgrafova dobijena tokom iteracija algoritma se može modelovati balansiranim drvetom T u kome listovi odgovaraju čvorovima polaznog grafa G , dok se u korenu nalazi jedan čvor koje je zapravo kontrahovani graf G . Svaki unutrašnji čvor z ima neke potomke z_1, z_2, z_3 itd. i svaki od njih je jedan kontrahovani podgraf čvorova sa sledećeg nivoa (idući od korena naniže). Čvoru z odgovara kontrahovani podgraf C_z , čiji su čvorovi upravo potomci čvora z . Jasno je da svaki nivo drveta minora T predstavlja jedan minorni graf od G . Kada se završi konstrukcija drveta minora, MST polaznog grafa se računa rekursivno za svaki minorni graf C_z . Pošto u tom trenutku zasigurno znamo da svaki od C_z zadovoljava svojstvo kon-

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

traktibilnosti, jer se oni konstruišu da zadovoljavaju to svojstvo, $MST(G)$ se dobija spajanjem MST od prethodno izračunatih minora [2]. Ovo je svojstvo kompozicije (eng. *composition property*) koje tvrdi da je $MST(G)$ unija svih $MST(S_i)$, gde su S_i disjunktne, kontraktibilni podgrafi od G . Na slici 2.11 dat je primer grafa (levo) i njegovog drveta minora (desno).



Slika 2.11: Primer konstrukcije drveta minora T

Pre detaljnije analize samog algoritma neophodno je uvesti pojam *izmenjenih grana* (eng. *corrupted edges*). Naime, tokom kreiranja drveta minora T cene nekih grana mogu biti uvećane i takve grane se nazivaju izmenjene grane. Cena se ne može promeniti bilo kojoj grani, već samo ukoliko je ona *granična* (eng. *border edge*). Do uvećanja cene dolazi usled korišćenja soft hipa kojim se efikasno pronalazi minimum, ali uz određenu stopu greške dovodi do uvećanja vrednosti koje čuva, što su u našem slučaju cene grana. Detaljnije o ovoj strukturi i koje su to grane granične sledi u narednoj sekciji. Vratimo se na izmenjene grane. Nisu sve izmenjene grane problematične, već samo one koje su incidentne nekoj komponenti C_z koja je u tom trenutku kontrahovana u jedan čvor. Takve grane nazivamo *lošim* (eng. *bad edges*). One se prilikom procesa kreiranja drveta minora T eliminišu iz razmatranja dok se ostale izmenjene grane smatraju prihvatljivim i prilikom svakog novog rekursivnog poziva njihove cene se vraćaju na originalne. Dobra stvar je što je broj ovakvih grana ograničen.

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

Izvršavanje Šazelovog algoritma se sastoji iz narednih koraka:

- Izvršavanje počinje primenom nekoliko rundi Boruvkinog algoritma. Cilj ove faze je redukovati broj čvorova grafa. Kao rezultat se dobija skup grana koji ćemo označiti sa E_{Bor} .
- Zatim se kreira drvo minora T i konstruiše se graf B koji se sastoji od tzv. loših grana.
- Za svaki unutrašnji čvor $z \in T$, rekurzivno se rešava problem manje dimenzije, ali bez loših grana $MST(C_z \setminus B)$. Uniranjem rezultata svih rekurzivnih poziva ovog oblika dobija se drvo F .
- Algoritam se rekurzivno primenjuje nad unijom prethodno dobijenog skupa grana F sa skupom loših grana B . Krajnji rezultat čini $MST(F \cup B) \cup E_{Bor}$.

Sledi detaljniji opis i analiza svakog od navedenih koraka.

Prvi korak: primena Boruvkinog algoritma

S obzirom na to da je ovaj algoritam rekurzivan, mogu se izdvojiti dva specijalna slučaja, kada je $n = O(1)$ i kada je $t = 1$, t je parametar koji kontroliše dubinu rekurzije. Ovo su bazni slučajevi, odnosno čine izlaze iz rekurzije. Prvi slučaj se odnosi na scenario kada ostane veoma mali broj čvorova i tada nije neophodno primenjivati ovaj složeni postupak rekurzivno, već je moguće izračunati MST direktno, korišćenjem nekog poznatog, jednostavnijeg algoritma, npr. Kruskalovog. Drugi slučaj se odnosi na scenario kada je parametar t , koji kontroliše dubinu rekurzije, jednak jedinici. U tom slučaju je u vremenu $O(n^2)$ moguće naći MST grafa samo primenom Boruvkinog algoritma dok na kraju ne ostane samo jedan čvor. Što se tiče Boruvkinih faza, to su one iste faze koje su korišćene u Kargerovom i Fredman-Tardžanovom algoritmu. Ne treba ih poistovetiti sa Boruvkinim algoritmom jer se ovime ne konstruiše nužno kompletno MST grafa, već se primenjuje nekoliko iteracija ovog algoritma. Ukratko, pronalazi se incidentna grana najmanje težine za svaki od čvorova. Nakon toga se povezani čvorovi kontrahuju u jedan, novi, čvor. Ovim postupkom se broj čvorova značajno smanjuje. U svakoj fazi se broj čvorova smanjuje bar upola, tako da se nakon c uzastopnih Boruvkinih faza, dobija novi graf G_0 u kome važi da je $n_0 \leq n/2^c$ i $m_0 \leq m$. Ova faza se izvršava u linearnom vremenu po broju grana i čvorova.

Drugi korak: kreiranje drveta minora

U ovoj fazi algoritma važnu ulogu imaju *aktivne putanje* (eng. *active path*). Neka je prilikom konstrukcije drveta T trenutni čvor z . On, kao i svaki unutrašnji čvor, nastaje kontrahovanjem čvorova sa nižeg nivoa. Označimo ih sa z_1, z_2, \dots, z_k . Oni čine trenutnu aktivnu putanju $z_1, z_2, \dots, z_k = z$. Obradom jednog po jednog čvora sa aktivne putanje, počevši od čvora z_1 nastaju grafovi $C_{z_1}, C_{z_2}, \dots, C_{z_k}$. Nakon što je poslednji graf C_{z_k} konstruisan, on se kontrahuje u $C_{z_{k-1}}$.

Za svako $i < k$ čuva se grana koja spaja komponente C_{z_i} i $C_{z_{i+1}}$, odnosno grana koja povezuje uniju do sada obrađenih čvorova na aktivnoj putanji i naredni čvor $C_{z_{i+1}}$. Takva grana se zove lančana veze (eng. *chain-link*). Cena lančane veze može biti izmenjena u soft hipu, pa se zato koristi izraz *trenutna cena* (eng. *current cost*). Takođe, važan pojam je *radna cena* (eng. *working cost*) koja je zapravo trenutna cena ukoliko je cena izmenjena, odnosno originalna ukoliko ukoliko nije. *Granična grana* (eng. *border edge*) je grana koja povezuje uniju do sada obrađenih komponenti na trenutno aktivnoj putanji sa ostatkom grafa. Drugim rečima, to je grana čiji se samo jedan kraj nalazi u obrađenom delu aktivne putanje. Grane najmanje cene od svih grana koje povezuju trenutnu komponentu sa nekom, već ranije obrađenom, komponentom se nazivaju *min-link* (eng. *min-link*).

Za razumevanje algoritma neophodno je objasniti i svrhu soft hipa. Za svako j granične grane oblika (u, v) , gde je $u \in C_{z_j}$, smeštaju se soft hip. Postoje dva oblika soft hipa koji čuvaju granične grane $H(j)$ i $H(i, j)$, gde je $i < j$. Hip oblika $H(j)$ se može posmatrati kao lokalni hip koji za svaku komponentu C_{z_j} čuva grane koje vode ka čvorovima v koji su već povezani sa nekom ranijom komponentom. Dok hip oblika $H(i, j)$ čuva grane tako da čvor v jeste sused komponente C_{z_i} , ali nije nijedne druge komponente C_{z_l} za $i < l < j$.

Tokom celog izvršavanja se održavaju dve invarijante.

Invarijanta 1: Trenutna cena lančane veze ne sme biti veća od cene bilo koje granične grane, a radna cena lančane veze mora biti manja od radne cene svake grane koja povezuje C_{z_i} i C_{z_j} , za $j \leq i$.

Invarijanta 2: Svaka grana može pripadati tačno jednom hipu u datom trenutku. Odnosno, za svako j granične grane oblika (u, v) , gde je $u \in C_{z_j}$, smeštaju se ili u soft hip sa oznakom $H(j)$ ili u $H(i, j)$, gde je $i < j$.

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

Održavanjem min-link grana, održava se prva invarijanta, odnosno, uslov koji se tiče radne cene lančane veze. Iz svega navedenog sledi da se održavanjem prve invarijante obezbeđuje opadajući niz lančanih veza, što je ključni uslov za bezbedno kontrahovanje grafova. Bitno zapažanje je da se svakim odabirom lančane veze bira prvi čvor naredne komponente $C_{z_{i+1}}$. Kako sa dodavanjem narednih čvorova komponenta raste, min-link grana može biti ažurirana na neku manju vrednost, te se njena i cena lančane veze mogu znatno razlikovati. Održavanjem hipova se čuva i druga invarijane i time se smanjuje efekat širenja „loših grana“. Obe invarijante se održavaju tokom čitavog procesa kreiranja drveta T koji se realizuje pomoću dve operacije: skraćivanjem aktivne putanje (eng. *retraction*) i njenim proširivanjem (eng. *extension*).

Operacija skraćivanja putanje

U trenutku kada i poslednji podgraf C_{z_k} sa aktivne putanje dostigne ciljanu veličinu, određenu brojem čvorova n_{z_k} , dolazi do retrakcije (skraćivanja putanje). Na taj način se kontroliše veličina komponenti i kraj aktive putanje se pomera na z_k . Kontrahovanjem komponente C_{z_k} u jedan jedini čvor, koga ćemo označiti sa a , komponenta postaje deo prethodno konstruisane komponente sa putanje $C_{z_{k-1}}$. Komponenta $C_{z_{k-1}}$ dobija novi čvor a , ali i nekoliko novih grana koje ga povezuju sa drugim čvorovima iz $C_{z_{k-1}}$. Među njima je i lančana grana koja je ranije povezivala komponente C_{z_k} i $C_{z_{k-1}}$ na aktivnoj putanji. Obe invarijante ostaju ispoštovane, iako za drugu to nije očigledno i zahteva detaljniju analizu. Naime, nakon kontrahovanja, hipovi $H(k)$ i $H(k-1)$ nam više nisu potrebni. Ovi hipovi se uništavaju, a grane koje su se nalazile u njima se smeštaju u neki od trenutnih hipova, dok se loše grane mogu odbaciti iz daljeg razmatranja. Sve grane iz $H(k)$ i $H(k-1)$ se prvo grupišu u klastere na osnovu spoljašnjeg čvora, a zatim se iz svakog klastera čuva samo ona koja ima najmanju cenu. Tada se grane raspoređuju u hipove. Ispostavlja se da ako se grana nalazila u hipu $H(k)$ i deli čvor sa nekom granom iz $H(k-1, k)$ ili se nalazila u hipu $H(k-1, k)$ može se smestiti u $H(k-1)$. Inače, grana dolazi iz $H(k)$ i može se smestiti u $H(i, k)$ za $i < k-1$. Za svako $i < k-1$ hip $H(i, k)$ se može stopiti u $H(i, k-1)$.

Operacija proširivanja putanje

Operaciji proširivanja putanje često prethodi operacija *fuzije* (eng. *fusion*). Da bi se ona izvršila, potrebno je pronaći graničnu ivicu najmanje cene među svim granama koje se trenutno čuvaju u hipovima. Neka je to grana (u, v) . Posmatrajući unazad

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

komponente na aktivnoj putanji, traži se ona koja je najdalje na putanji, odnosno najranije dodata, a da ima radnu cenu min-link grane ne veću od cene grane (u, v) . Drugim rečima, potrebno je odrediti komponentu C_{z_i} za najmanju vrednost i koja ispunjava prethodno navedeni uslov. Ideja je da se prvo sve grane, čija se oba kraja nalaze posle komponente C_{z_i} na aktivnoj putanji, kontrahuju u jedan čvor, označimo ga sa b , što je operacija fuzije. Fuzija je značajna za održavanje prve invarijante algoritma. Zatim da se izvrši kontrahovanje grane (a, b) , gde je a kraj grane koji je incidentan sa komponentom C_{z_i} , čime se dobija novi najdalji čvor na putanji i putanja se proširuje. Nakon operacije proširivanja putanje je neophodno ažurirati sve min-link grane i ažurirati hipove koristeći istu logiku kao i u operaciji skraćivanja putanje, sa jednom bitnom razlikom. Ukoliko je došlo do fuzije, što ne mora uvek biti slučaj, potrebno je ažurirati ne samo $H(k)$ i $H(k-1, k)$, već i $H(i+1), \dots, H(k)$ kao i sve hipove oblika $H(j, j')$ za $i \leq j < j'$. Time druga invarijanta ostaje održana.

Sada kada je objašnjeno kako se realizuju operacije proširivanja i skraćivanja aktivne putanje, jasno je da se pomoću njih veoma jednostavno konstruiše drvo minora T . Ako je trenutni čvor z_k , aktivna putanja se proširuje sve dok je to moguće, odnosno dok komponenta C_{z_k} ne dostigne predviđenu veličinu. Takođe, često se usput, po potrebi, dešava i fuzija. Dostizanjem predviđenog broja čvorova otpočinje operacija skraćivanje putanje.

Treći korak: rekurzija u podgrafima drveta T

Kao rezultat prethodnog koraka konstruisano je drvo T . Sada je potrebno rekurzivno pozvati algoritam za svaki njegov čvor z , odnosno za svaki podgraf C_z . U prethodnom koraku algoritma su već odbačene neke grane, koje smo okarakterisali kao loše. Međutim, još uvek postoje neke grane kojima je hip izmenio vrednost, ali pošto nisu predstavljale problem pri kreiranju drveta one nisu odbačene. Pre rekurzivnih poziva potrebno je tim granama vratiti originalne cene. Stoga se algoritam zapravo poziva nad $C_z \setminus B$ i vrednost parametra t se umanjuje što određuje visinu drveta koje se konstruiše. Rezultat unije ovakvih rekurzivnih poziva je skup F koji sadrži grane između čvorova podgrafa oblika C_z . Za svako z , jasno je da svi njegovi čvorovi predstavljaju kontrahovane grafove sa nižeg nivoa, pa se zato algoritam rekurzivno poziva za svu njegovu decu. Međutim, postavlja se pitanje šta se dešava u slučaju da neki čvorovi predstavljaju rezultat fuzija. Situacija je zapravo ista jer svaki takav čvor predstavlja dete kao i svaki drugi čvor unutar C_z , sa samo jednom

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

razlikom, a to je da on ima svoju hierarhiju čvorova koja nastaje usled fuzije. Kako je prethodno objašnjeno, prilikom fuzija značajne su grane oblika (a, b) , gde je a čvor incidentan sa komponentom kojom se proširuje putanja, dok je b čvor u kome su kontrahovane prethodne komponente sa aktivne putanje. Ukoliko nisu loše one se dodaju u F , a inače se njihova obrada ostavlja za neke naredne rekurzivne pozive.

Poslednji korak

Kao rezultat druge faze algoritma konstruisano je drvo T i dobijen skup B grana koje se ne mogu odmah obraditi jer imaju izmenjene vrednosti cena i potencijalno narušavaju invarijante. U trećem koraku algoritma kreira se skup F grana unutar kontrahovanih podgrafa koje sigurno pripadaju $MST(G)$. Konačno, kako bi se dobile grane koje povezuju već obrađene komponente, algoritam se poziva za $F \cup B$. Rezultat ovog poziva, zajedno sa prethodno pronađenim granama u prvoj, Boruvkinjoj, fazi, vraća minimalno povezujuće drvo grafa G .

Dokaz korektnosti: Za potrebe dokaza *korektnosti* ovog algoritma dovoljno je pokazati da važi sledeće: Ako grana originalnog grafa G nije proglašena lošom, niti je dodata u skup F , onda ona sigurno ne pripada $MST(G)$. U nastavku sledi skica dokaza dobijena razmatranjem dva slučaja.

Neka je e grana za koju važi da se njena oba krajnja čvora nalaze u nekom minoru C_i grafa G , a koja ni u jednom koraku algoritma nije odbačena. Pošto ona nije u F , znači da nije deo lokalnog MST, odnosno ne pripada $MST(C_i)$. Pošto grana nije označena kao loša i nije u $MST(C_i)$, na osnovu svojstva kompozicije sledi da je van $MST(G)$. Važno je prisetiti se da čim neka grana nije deo MST , to znači da ona gradi neki ciklus, a dodatno je i najskuplja grana u njemu. Ova analiza je izvršena u odnosu na radne cene grana, a kako grana e nije proglašena lošom, njena cena je zapravo sve vreme jednaka polaznoj. Ostalim granama tog ciklusa u kome je grana e grana najveće cene, se cene mogu vratiti na inicijalne, ali to znači da sada neke od njih mogu biti manje ili ostati iste, ali nikako postati veće. Ako je grana e bila najskuplja u nekom ciklusu, ona to ostaje i nakon resetovanja cena na inicijalne vrednosti.

Razmotrimo slučaj kada je grana $e(u, v)$ odbačena u fazi kreiranja drveta T . Pre nego što je odbačena, postojala je grana $e'(u', v)$ manje cene, odnosno grana koja je takođe incidentna čvoru v . Prilikom operacije skraćivanja putanje ili fuzije dolazi do

GLAVA 2. PREGLED ALGORITAMA ZA KONSTRUKCIJU MINIMALNOG POVEZUJUĆEG DRVETA

kontrahovanja obe grane u jedan čvor. Pošto je taj čvor kontraktibilan u G , radne cene ovih grana se ponašaju kao originalne u odnosu na $MST(G)$. Nijedna od ove dve grane nema izmenjenu cenu, ali je e skuplja od e' . Za dve grane koje imaju zajednički čvor, sa originalnim cenama u G , važi da ona koja je skuplja može biti izbačena i sigurno ne pripada $MST(G)$.

Analiza složenosti: Drvo T se može opisati visinom drveta minora d i broja čvorova n , gde je naročito značajan broj čvorova n_z svake od komponenti C_z . Nakon dublje analize dolazi se do vremenske složenosti izgradnje drveta T od $O(m + d^3n)$. Posmatrajući ovu vremensku složenost, postavlja se pitanje kako odabrati visinu i broj čvorova svake komponente drveta, tako da izvršavanje ostane efikasno. Ukoliko se odabere mala visina drveta, konstrukcija drveta je izuzetno brza, ali su komponente na kojima se vrši rekurzija veće, imaju više čvorova, pa je vreme koje se troši na rekurziu veliko. Ukoliko se odabere velika visina drveta, konstrukcija drveta jeste značajnije sporija nego u prvom slučaju, ali su komponente čvorovi C_z značajno manje i rekurzija je mnogo brža. Ovde je potrebno naći kompromis kako bi kompletan algoritam trošio razumno vreme na proces izgradnje drveta i rekurzivne pozive. Spomenuti kompromis se postiže uvođenjem inverzne Akermanove funkcije.

Biramo parametre tako da ceo graf može da se “smesti” u koren drveta T , pri čemu je parametar t definisan pomoću Akermanove funkcije koja raste izuzetno brzo i time se obezbeđuje da dubina hijerarhije, odnosno, broj nivoa rekurzije bude veoma mali. Ukupno potrošeno vreme na svakom nivou rekurzije je linearno po broju grana. Značajno za analizu ukupne vremenske složenosti algoritma je da je broj grana, koje su obeležene kao loše prilikom konstrukcije drveta T , ograničen sa $|B| \leq m_0/2 + d^3n_0$.

Moguće je odrediti vremensku složenost svake faze algoritma i ispravnim odabirom vrednosti parametra t uz pomoć Akermanove funkcije, dolazi se do ukupne vremenske složenosti od $O(m\alpha(m, n))$, gde je α inverzna Akermanova funkcija koja jako sporo raste. Dokaz složenosti ovog algoritma je dugačak i sadrži dosta tehničkih detalja i može se naći u [2].

Bibliografija

- [1] Otakar Boruvka. Otakar Borůvka on minimum spanning tree problem Translation of both the 1926 papers, comments, history, 1927. on-line at: <https://www.sciencedirect.com/science/article/pii/S0012365X00002247>.
- [2] Bernard Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity, 2000. on-line at: <https://mathscinet.ams.org/mathscinet/relay-station?mr=MR1866456>.
- [3] Bernard Chazelle. The soft heap: an approximate priority queue with optimal error rate, 2000. on-line at: <https://dl.acm.org/doi/10.1145/355541.355554>.
- [4] Richard Cole, Philip Klein, and Rober Tarjan. A linear-work parallel algorithm for finding minimum spanning trees, 1994. on-line at: <chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cs.brown.edu/research/pubs/pdfs/1994/Klein-1994-LWP.pdf>.
- [5] Jason Eisner. State-of-the-Art Algorithms for Minimum Spanning Trees, 1997. on-line at: www.cs.jhu.edu/~jason/papers/eisner.mst-tutorial.pdf.
- [6] Michael Fredman and Bob Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms, 1987. on-line at: <https://mathscinet.ams.org/mathscinet/relay-station?mr=MR0904195>.
- [7] Vojtěch Jarník. O jistém problému minimálním, 1930. on-line at: <https://dml.cz/handle/10338.dmlcz/500725>.
- [8] David Karger, Philip Klein, and Rober Tarjan. A randomized linear-time algorithm to find minimum spanning trees, 1995. on-line at: <https://cs.brown.edu/research/pubs/pdfs/1995/Karger-1995-RLT.pdf>.

- [9] Joseph Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem, 1956. on-line at: <https://mathscinet.ams.org/mathscinet/relay-station?mr=MR0078686>.
- [10] Filip Marić and Vesna Marinković. Konstrukcija i analiza algoritama, 2024. on-line at: <https://poincare.matf.bg.ac.rs/~filip/kiaa/kiaa.pdf>.
- [11] Jaroslav Nešetřil and Helena Nešetřilová. The Origins of Minimal Spanning Tree Algorithms - Boruvka and Jarník, 2010. on-line at: <https://www.cs.cmu.edu/~15850/notes/lec1.pdf>.
- [12] CMU School of Computer Science. Minimum Spanning Trees, 2024. on-line at: <https://www.cs.cmu.edu/~15850/notes/lec1.pdf>.