Osnovna svojstva stabala i primjena na problem spajanja

Jan Berger[1] i Mario Krnić[2]

Prema bolonjskom načinu studiranja, studenti Fakulteta elektrotehnike i računarstva završavaju preddiplomski studij takozvanim *završnim radom*. U članku [2] dan je primjer jednog takvog rada iz matematike. Naravno, ukoliko je student za završni rad odabrao temu iz matematike, osim teorije očekujemo i primjenu u struci, bilo da je riječ o elektrotehnici ili računarstvu.

U ovome članku izložit ćemo dijelove završnog rada studenta Jana Bergera [1] sa studija Računarstva (modul Računarska znanost). U spomenutom završnom radu obrađena je jedna klasa jednostavnih grafova, odnosno stabla. Kao primjene, promatraju se neki praktični problemi koji se rješavaju uz pomoć stabala.

Radi lakšeg snalaženja čitatelja, u prvoj točki ovog rada navodimo osnovne definicije i oznake koje susrećemo u teoriji grafova. U drugoj točki iskazujemo i dokazujemo osnovna svojstva i karakterizacije stabala. Nadalje, formuliramo i takozvani problem spajanja te dajemo i odgovarajući algoritam pomoću kojeg se spomenuti problem rješava. Dakako, pri tome koristimo i prethodno izloženu teorijsku pozadinu. Konačno, u trećoj točki dajemo i odgovarajuću programsku implementaciju za spomenuti algoritam.

1. Osnovni pojmovi i oznake

Radi boljeg razumijevanja ovoga rada, prvo ćemo se prisjetiti osnovnih pojmova iz teorije grafova, a zatim ćemo uvesti oznake koje ćemo koristiti.

Definicija 1. Jednostavni graf G sastoji se od nepraznog konačnog skupa vrhova $V(G)$ te konačnog skupa bridova $E(G)$ (skup dvočlanih podskupova skupa $V(G)$).

U jednostavnom grafu isključena je mogućnost da su dva vrha spojena s više bridova, ili da postoji brid koji spaja vrh sa samim sobom. Ako to dopustimo, tada govorimo o **općem grafu** ili, kraće, samo grafu.

[1]Jan Berger, Fakultet elektrotehnike i računarstva, Zagreb
[2]Mario Krnić, Fakultet elektrotehnike i računarstva, Zagreb

Skup vrhova označavat ćemo s $V(G)$, a ako je jasno da se radi o grafu G, označavat ćemo ga kraće, samo s V. Broj vrhova označavat ćemo s $v(G)$ ili v. Slično, $E(G)$ odnosno E je skup bridova, a broj bridova u grafu označavat ćemo s $\varepsilon(G)$ odnosno s ε.

Definicija 2. Podgraf grafa G je graf čiji vrhovi pripadaju skupu $V(G)$, a bridovi skupu $E(G)$.

Definicija 3. Za brid $e = \{u, v\}$ kažemo da spaja vrhove u i v (kraće još možemo pisati $e = uv$). Tada kažemo da su vrhovi u i v grafa G **susjedni**. Također, kažemo da su vrhovi u i v **incidentni** s bridom e.

Jedno od važnijih svojstava grafa je svojstvo povezanosti. Da bismo ga definirali, prvo ćemo uvesti pojam *unije grafova*.

Definicija 4. Unija grafova $G_1 = \left(V\left(G_1\right), E\left(G_1\right)\right)$ i $G_2 = \left(V\left(G_2\right), E\left(G_2\right)\right)$ je graf $G_1 \cup G_2 = \left(V\left(G_1\right) \cup V\left(G_2\right), E\left(G_1\right) \cup E\left(G_2\right)\right)$.

Definicija 5. Graf je **povezan** ako se ne može prikazati kao unija dvaju disjunktnih grafova. U suprotnom kažemo da je graf **nepovezan**.

Očito, svaki se nepovezani graf može prikazati kao unija povezanih grafova. Svaki član te unije zovemo **komponentom povezanosti**. Broj komponenti povezanosti grafa G označavat ćemo s $\omega(G)$.

Ako je graf povezan, onda se može „šetati" po njemu, odnosno prelaziti iz vrha u vrh ukoliko su vrhovi susjedni. Sada ćemo precizno definirati šetnju, te pojmove usko vezane uz nju.

Definicija 6. Šetnja u grafu G je netrivijalan konačan niz $v_0 e_1 v_1 e_2 v_2 \ldots e_n v_n$ čiji su članovi naizmjence vrhovi v_i i bridovi e_i, tako da su krajevi od e_i vrhovi v_{i-1} i v_i za svako i, $1 \le i \le n$.

Definicija 7. Šetnja u kojoj nema ponavljanja bridova naziva se **staza.** Staza koja nema ponavljanja vrhova je **put.** Put kod kojeg je početni vrh v_0 jednak završnome vrhu v_n zove se **ciklus.**

Napomenimo još kako se ciklus koji se sastoji od jednog brida naziva **petlja**. Brid koji nije petlja zovemo **pravi brid** ili **karika**.

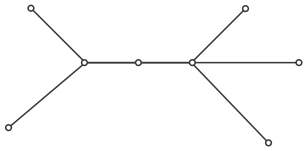
Uvedimo još i veličinu koja kazuje koliko susjednih vrhova ima svaki pojedini vrh u grafu.

Definicija 8. Stupanj vrha v grafa G, u oznaci $d(v)$ broj je bridova koji su incidentni s v. Vrh stupnja 0 zovemo izolirani vrh, dok je vrh stupnja 1 krajnji vrh.

2. Stabla

Osnovna svojstva stabala

Stablo je povezani aciklički graf, odnosno povezani graf koji ne sadrži cikluse. Aciklički grafovi još se nazivaju i *šume*, pa iz toga slijedi da su stabla povezani dijelovi šume. U nastavku navodimo i dokazujemo neka važna svojstva stabala.



Slika 1. Primjer stabla

TEOREM 1. *Svaka dva vrha u stablu povezana su jedinstvenim putem.*

Dokaz. Kontradikcijom. Neka je G stablo. Kako je G povezan, između svakog para vrhova postoji put. Ako bi postojala dva različita puta između neka dva vrha, tada bi unija tih dvaju putova tvorila zatvorenu šetnju koja bi sigurno sadržavala barem jedan ciklus, što je kontradikcija. □

TEOREM 2. *Ako je G stablo, onda je* $\varepsilon = v - 1$.

Prije dokaza Teorema 2, navedimo jedan pomoćni rezultat.

LEMA 1. *Svako netrivijalno stablo ima vrh v čiji je stupanj* $d(v) = 1$.

Dokaz Leme 1. Krenimo od nekog vrha v_1. Ako je njegov stupanj $d(v_1) = 1$, gotovi smo. Ako je pak $d(v_1) > 1$, pomičemo se iz v_1 duž nekog brida do susjednog vrha v_2. Ako je $d(v_2) = 1$, gotovi smo, a ako je pak $d(v_2) > 1$, produžimo dalje do v_3 duž nekog drugog brida. Nastavljajući tako, dolazimo do puta $v_1v_2v_3\ldots$ u grafu G. Vrhovi toga puta se ne ponavljaju jer bismo inače dobili ciklus koji stablo nema. Skup vrhova je konačan, pa stoga taj niz vrhova mora stati u nekom vrhu. Taj vrh u kojemu je niz stao mora biti stupnja 1 zbog toga što smo u njega ušli, a ne možemo ga napustiti. □

Dokaz Teorema 2. U ovome dokazu uvest ćemo tzv. Eulerovu karakteristiku grafa G tj. $\chi(G) = v - \varepsilon + 1$. Tvrdimo da za svako stablo G vrijedi $\chi(G) = 2$. Prema Lemi 1, postoji $v \in V(G)$ za koji je $d(v) = 1$. Izbacimo taj vrh i brid koji je incidentan s v. Tada opet dobivamo stablo $G_1 = G - v$. Očito, vrijedi $\chi(G_1) = (v-1) - (\varepsilon - 1) + 1 = v - \varepsilon + 1 = \chi(G)$.

Sada, opet prema Lemi 1, u stablu G_1 postoji vrh stupnja 1 pa uklanjanjem toga vrha i incidentnog brida dobivamo stablo G_2 i opet je $\chi(G_2) = \chi(G_1)$.

Nastavljajući taj proces, dobivamo stablo sa samo jednim vrhom v, a njegova Eulerova karakteristika je $\chi(v) = v - \varepsilon + 1 = 1 - 0 + 1 = 2$. Iz toga slijedi da svako stablo u tom procesu ima Eulerovu karakteristiku 2, pa je $\varepsilon = v - 1$. □

Navedimo sada jedno svojstvo koje vrijedi za bilo koji graf.

TEOREM 3. *Za svaki graf vrijedi* $\sum_{v \in V} d(v) = 2\varepsilon$.

Dokaz. Jednakost se dokazuje prebrojavanjem svih incidencija u grafu. Krenemo li od vrhova, za svaki pojedini vrh, takvih incidencija ima točno koliko je stupanj odgovarajućeg vrha. S druge strane, svaki brid ima dva kraja, pa incidencija ukupno ima 2ε. □

Sada, kao posljedicu imamo sljedeći:

KOROLAR 1. *a) Za svako stablo G vrijedi* $\sum_{v\in V} d(v) = 2v - 2$.

b) Svako netrivijalno stablo ima bar dva vrha stupnja 1.

Dokaz. a) Iz Teorema 2 i 3 slijedi $\sum_{v\in V} d(v) = 2\varepsilon = 2v - 2$.

b) Ako je *G* netrivijalno stablo, onda je $d(v) \geq 1, \forall v \in V(G)$. Tada iz a) odmah slijedi tvrdnja. □

Uočimo da je stablo, u neku ruku, najmanji povezani graf.

LEMA 2. *Za svaki brid* e *iz stabla G, preostali graf G – ε nije povezan.*

Dokaz. Prema Teoremu 2, *G* ima *v* – 1 bridova. Ako izbacimo brid, preostali graf ne može biti stablo jer ima *v* vrhova i *v* – 2 bridova. Uklanjanjem brida ne mogu se stvoriti novi ciklusi, pa preostali graf *G* – *ε* nije povezan. □

LEMA 3. *Ako je G povezani graf za koji je* $\varepsilon = v - 1$*, tada je G stablo.*

Dokaz. Ako povezani graf nije stablo, on mora imati ciklus. Uklonimo li bilo koji brid *e* iz nekog ciklusa, graf *G* – *e* i dalje će biti povezan. Tako nastali graf $G' = G - e$ ima također *v* vrhova, ali je $\varepsilon(G') = v - 2$. Graf G' je povezan pa, ako i dalje nije stablo, mora imati neki ciklus. Uklonimo ponovno neki brid iz toga ciklusa. Dobijemo i dalje povezan graf G'' itd. Konačno, doći će situacija kada se više neće moći uklanjati brid iz ciklusa, a da preostali graf bude povezan. Kada se to dogodi, doći ćemo do stabla. Ako se to dogodi u *k*-tom koraku, stablo će imati *v* vrhova i *v* – 1 – *k* bridova. Prema Teoremu 2 dobivamo da je *k* = 0, pa iz toga slijedi da je i polazni graf bio stablo. □

LEMA 4. *Ako graf G nema ciklusa i vrijedi* $\varepsilon = v - 1$*, onda je G stablo.*

Dokaz. Dovoljno je pokazati da je *G* povezan. Neka su $G_1, G_2, ..., G_k$ komponente povezanosti grafa *G* te neka je $v_i = v(G_i)$, $i = 1, 2, ..., k$. Tada je $v_1 + v_2 + ... + v_k = v$. Svaka je komponenta povezana i nema ciklus jer ga ni *G* nema. Stoga je svaki G_i stablo. Prema Teoremu 2, $\varepsilon(G_i) = v_i - 1$.

Ukupan broj bridova u *G* je $(v_1 - 1) + (v_2 - 1) + ... + (v_k - 1) = v_1 + v_2 + ... + v_k - k = v - k$.

Kako je $\varepsilon(G) = v - 1$, slijedi da je *k* = 1, pa *G* ima samo jednu komponentu. Dakle, *G* je stablo. □

Konačno, iz prethodnih lema slijedi važan rezultat u kojemu su opisane karakterizacije stabla.

TEOREM 4. *Neka je G graf. Tada su sljedeće tvrdnje ekvivalentne:*

i) G je stablo;

ii) između svaka dva vrha iz G postoji jedinstveni put;

iii) *G je povezan i* $\varepsilon = v - 1$*;*

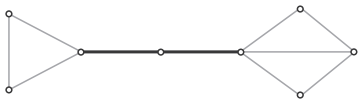
iv) *G je aciklički i* $\varepsilon = v - 1$.

Rezni bridovi i razapinjuće stablo

Izbacivanjem bilo kojega brida iz stabla dobivamo nepovezani graf. U tu svrhu, promotrimo općenitiju definiciju.

Definicija 9. Rezni brid grafa G je takav brid $e \in E(G)$ za koji je $\omega(G-e) > \omega(G)$, odnosno brid čijim se izbacivanjem graf G raspada na više komponenti povezanosti.

Na donjoj slici prikazan je graf na kojemu su podebljano istaknuti rezni bridovi:



Slika 2. Graf s istaknutim reznim bridovima

TEOREM 5. *Brid* $e \in E(G)$ *je rezni ako i samo ako e nije brid niti jednog ciklusa od G.*

Dokaz. $\Rightarrow$: Neka je e rezni brid. Tada je $\omega(G-e) > \omega(G)$, pa postoje vrhovi $u, v \in V(G)$ koji su povezani u G, a nisu u $G-e$. Stoga postoji (u,v)-put P u G, koji prolazi bridom e. Neka su x i y krajevi od e tako da x prethodi vrhu y na putu P. Vrh u je u $G-e$ povezan x dijelom puta P, dok je y u $G-e$ povezan v dijelom puta P. Kada bi e bio u ciklusu C, onda bi x i y bili povezani u $G-e$ putem $C-e$. Zbog toga bi u i v bili povezani u $G-e$, a to je kontradikcija.

$\Leftarrow$: Pretpostavimo da $e = xy$ nije rezni brid od G. Tada je $\omega(G-e) = \omega(G)$. Kako postoji (x, y)-put (brid xy) u G, to su x i y u istoj komponenti od G. Iz toga slijedi da su x i y u istoj komponenti od $G-e$ pa zbog toga postoji (x, y)-put u $G-e$. No, tada je e u ciklusu $P + e$ od G, što je kontradikcija. □

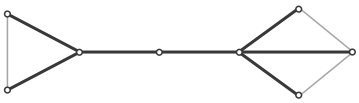
TEOREM 6. *Povezani graf G je stablo ako i samo ako je svaki brid u G rezni brid.*

Dokaz. $\Rightarrow$: Neka je G stablo i $e \in E(G)$. Kako je G aciklički, e nije sadržan u ciklusu, pa je, prema Teoremu 5, e rezni brid od G.

$\Leftarrow$: Pretpostavimo da je G povezan, ali da nije stablo. Tada G sadrži ciklus C. Prema Teoremu 5, niti jedan brid od C ne može biti rezni brid od G. □

Definicija 10. Razapinjuće stablo grafa G je razapinjući podgraf (sadrži sve vrhove) koji je stablo.

Na slici 3. podebljano je označeno razapinjuće stablo zadanoga grafa.



Slika 3. Graf s istaknutim razapinjućim stablom

KOROLAR 2. *Svaki povezani graf ima razapinjuće stablo.*

Dokaz. Neka je graf G povezan, a T njegov minimalni povezani razapinjući podgraf. Prema definiciji je $\omega(T)=1$ i $\omega(T-e)>1$ za svaki $e\in E(G)$. Iz toga slijedi da je svaki brid od T rezni pa, kako je T povezan, iz Teorema 6 slijedi da je T stablo. □

KOROLAR 3. *Ako je G povezani graf, onda je* $\varepsilon \geq v-1$.

Dokaz. Kako je G povezan, prema Korolaru 2 G sadrži razapinjuće stablo T. Stoga korištenjem Teorema 2 dobivamo $\varepsilon(G)\geq\varepsilon(T)=v(T)-1=v(G)-1$. □

TEOREM 7. *Neka je T razapinjuće stablo povezanoga grafa G,* $e\in E(G)\setminus E(T)$. *Tada T + e sadrži jedinstven ciklus.*

Dokaz. Kako je T aciklički, svaki ciklus od $T + e$ sadrži e. Nadalje, C je ciklus od $T + e$ ako i samo ako je $C – e$ put u T koji spaja krajeve brida e. Prema Teoremu 1, postoji jedinstveni takav put u T. Prema tome, $T + e$ sadrži jedinstven ciklus. □

Napomenimo još kako postoji jednostavna i elegantna rekurzija pomoću koje se računa broj razapinjućih stabala u nekome grafu. Detaljnije o tome čitatelj može vidjeti u [1] i [5].

Primjena stabala na problem spajanja i Kruskalov algoritam

Problem spajanja jedan je od najpoznatijih problema za čije se rješavanje koriste stabla. Promotrimo taj problem. Neke gradove treba povezati mrežom prometnica. Pritom je dana cijena c_{ij} direktne veze među gradovima v_i i v_j, a to treba učiniti tako da trošak izgradnje prometnica bude minimalan.

Svaki je grad vrh težinskog grafa s težinama $w(v_iv_j)=c_{ij}$ te je taj problem ekvivalentan problemu nalaženja povezanog razapinjućeg podgrafa minimalne težine u zadanom težinskom grafu G. Težine grafa su troškovi koji su veći ili jednaki nuli, pa stoga nema smisla zatvarati cikluse jer će ukupna težina biti veća. Iz toga zaključujemo da će dobiveni podgraf biti neko razapinjuće stablo T od G. Razapinjuće stablo minimalne težine u težinskome grafu G još se zove i *optimalno stablo*.

Ako je $w(e)=1$ za svaki $e\in E(G)$, tada je optimalno stablo zapravo razapinjuće stablo s minimalnim brojem bridova. Budući da svako razapinjuće stablo ima jednak broj bridova v – 1, dovoljno je naći jedno razapinjuće stablo. Imamo sljedeći induktivni algoritam za ovaj poseban slučaj:

i) Odaberi kariku e_1;

ii) ako su bridovi $e_1,...,e_i$ već odabrani, odaberi $e_{i+1}\in E\setminus\{e_1,...,e_i\}$, tako da je $G\left[\{e_1,...,e_{i+1}\}\right]$ acikličan;

iii) zaustavi postupak kada se korak ii) više ne može provesti.

Ovaj će algoritam dati jedno razapinjuće stablo jer je maksimalni aciklički podgraf povezanog grafa nužno razapinjuće stablo.

Prethodni algoritam poopćio je Joseph Kruskal 1956. godine kako bi riješio opći problem koji vrijedi za proizvoljne težine.

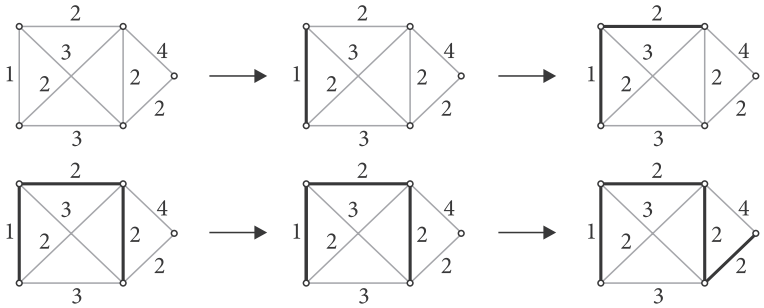
Kruskalov algoritam

i) Odaberi kariku e_1 tako da je $w(e_1)$ minimalno;

ii) ako su bridovi $e_1,...,e_i$ odabrani, odaberi $e_{i+1} \in E \setminus \{e_1,...,e_i\}$ tako da je:
a) $G\left[\{e_1,...,e_{i+1}\}\right]$ acikličan
b) $w(e_{1+1})$ minimalno pod uvjetom a);

iii) zaustavi postupak kada se korak ii) više ne može izvršiti.

Primijenimo Kruskalov algoritam na sljedeći težinski graf:



Slika 4. Određivanje optimalnog stabla Kruskalovim algoritmom

Kruskalov algoritam očito daje razapinjuće stablo, no sljedeći teorem osigurava da takvo stablo uvijek bude optimalno.

TEOREM 8. *Svako razapinjuće stablo* $T^* = G\left[\{e_1,e_2,...,e_{v-1}\}\right]$ *konstruirano Kruskalovim algoritmom optimalno je stablo.*

Dokaz. Kontradikcijom. Za svako razapinjuće stablo T od G, $T \neq T^*$, označimo $f(T) := \min\{i \mid e_i \notin E(T)\}$. Pretpostavimo da T^* nije optimalno, dok je T optimalno stablo od G za koje je $f(T)$ najveće moguće.

Neka je $f(T) = k$. Tada su $e_1,e_2,...,e_{k-1} \in E(T) \cap E(T^*)$, ali e_k ne pripada skupu $E(T)$. Prema Teoremu 7, $T + e_k$ sadrži jedinstveni ciklus C.

Neka je e'_k brid u ciklusu C koji je u T, ali nije u T^*. Iz Teorema 5 slijedi da e'_k nije rezni brid od $T + e_k$. Zbog toga je $T' = (T + e_k) - e'_k$ povezani graf s $v - 1$ bridova, pa prema Teoremu 4 i) i iii) slijedi da je T' također razapinjuće stablo od G. Očito je $w(T') = w(T) + w(e_k) - w(e'_k)$.

No u Kruskalovom algoritmu e_k je bio odabran kao brid s najmanjom težinom, tako da je $G\left[\{e_1,...,e_k\}\right]$ acikličan. Isto tako je i $G\left[\{e_1,...,e_{k-1},e'_k\}\right]$ podgraf

od T, acikličan. Stoga je, po izboru e_k iz Kruskalovog algoritma i optimalnosti od T, $w(e_k) \leq w(e'_k)$.

Iz prethodnih dviju relacija slijedi $w(T') \leq w(T)$. Prema tome, kako je T optimalno, i T' je optimalno stablo. Međutim, $f(T') > k = f(T)$, što je u kontradikciji s izborom od T. Zbog toga je $T = T^*$ pa je T^* doista optimalno stablo. □

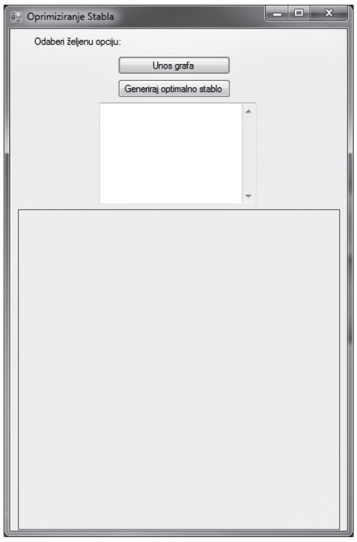
Kruskalov algoritam primjer je tzv. **pohlepnog algoritma**. On u svakome koraku dodaje brid najmanje težine tako da s već odabranim bridovima ne tvori ciklus. Taj se postupak ponavlja sve dok se ne odabere v – 1 bridova. Pohlepnost vidi samo prvi idući korak, pa takav algoritam radi tako da uvijek napravi lokalno najbolje što može. No, takav algoritam nije prikladan za sve probleme, tako npr. za određivanje najkraćeg puta pohlepni algoritam ne mora dati najkraći put.

3. Programska podrška za Kruskalov algoritam

Nakon razrade teorijskog dijela napravljen je program koji implementira Kruskalov algoritam za dobivanje optimalnog stabla iz zadanog težinskog grafa. Prvo ćemo objasniti korištenje programa, a nakon toga slijedi objašnjenje važnijih dijelova programa.

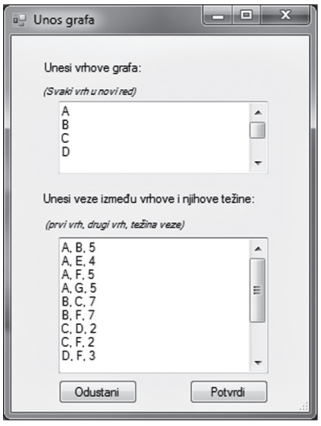
Upotreba programske podrške

Pokretanjem programa prvo se otvara početni prozor (Slika 5.) na kojemu imamo dvije tipke (Unos grafa i Generiraj optimalno stablo). Ispod tipki nalazi se prostor za ispis optimalnog stabla, dok se u najdonjem prozoru ono iscrtava.



Slika 5. Početni prozor aplikacije

Na početku rada, prvo trebamo unijeti težinski graf pa je stoga potrebno pritisnuti tipku za unos novog grafa, nakon čega se otvara novi prozor za unos (Slika 6.):

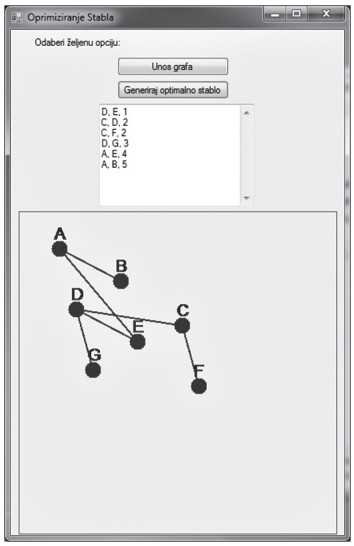


Slika 6. Prozor za unos grafa

U prvi prostor za unos teksta unose se svi vrhovi težinskoga grafa tako da se svaki novi vrh upisuje u novi red. U donji prostor za unos teksta unose se veze između vrhova s njihovim težinama. Format unosa mora biti sljedeći: *prviVrhVeze, drugiVrhVeze, težinaVeze.* Svaka veza upisuje se u novi redak.

Kada je završio unos, imamo dvije tipke s opcijama. Tipka „Odustani" pritišće se ukoliko korisnik želi odustati od unesenog grafa, odnosno ako se želi vratiti u početni prozor bez spremanja novoga grafa. Ako pak korisnik želi spremiti unesene podatke, tada treba pritisnuti tiku „Potvrdi". Pritiskom tipke „Potvrdi" korisnik pokreće algoritme za ažuriranje unesenih podataka, te se vraća na početni prozor.

Nakon što je graf uspješno unesen, korisnik može pokrenuti optimiziranje grafa te ispis i iscrtavanje optimalnog stabla pritiskom na tipku „Generiraj optimalno stablo". Ako je tipka pritisnuta, nakon pozadinskog izvođenja svih potrebnih algoritama, u bijelom prostoru ispod tipki ispisuju se sve veze optimalnog stabla u sljedećem obliku: *prviČvor, drugiČvor, težinaVeze* te se konačno u najdonjem prostoru iscrtava optimalno stablo (Slika 7.).



Slika 7. Početni prozor nakon unosa težinskoga grafa i pokretanja generiranja optimalnog stabla

Izrada programske podrške

Programska podrška napravljena je pomoću alata za razvoj *Visual Studio* te programskog jezika *C#*. U nastavku će biti objašnjena implementacija najvažnijih dijelova programa.

Na početku izvođenja programa unose se vrhovi grafa i njegove veze. Nakon što su upisani vrhovi i veze, te nakon pritiska tipke „Potvrdi", pokreće se čitanje unesenih vrijednosti te njihova obrada i spremanje u spremnike podataka u potrebnom obliku.

Obrada unosa vrhova:

```
    private void ocitajVrhove()
   {
        string[] granicnik = new string[] { "\r\n" };
                string[] vrhovi = ucitaniVrhovi.Split(granicnik,
                     StringSplitOptions.RemoveEmptyEntries);

        foreach (string vrh in vrhovi)
        {
            listaVrhova.Add(vrh);
        }
    }
```

Obrada unosa veza:

```
        private void ocitajVeze()
      {
          string[] granicnikVeza = new string[] { "\r\n" };
          string[] granicnikUnutarVeze = new string[] { ", " };
          const int indexPrvogVrha = 0;
          const int indexDrugogVrha = 1;
          const int indexTezineVeze = 2;
          string[] vezeVrhova = ucitaneVezeVrhova.Split(granicnikVeza,
StringSplitOptions.RemoveEmptyEntries);

           foreach (string veza in vezeVrhova)
           {
               string[] izdvojenaVeza = veza.Split(granicnikUnutarVeze,
StringSplitOptions.RemoveEmptyEntries);

               if (Vrhovi.Contains(izdvojenaVeza[indexPrvogVrha]) &&
Vrhovi.Contains(izdvojenaVeza[indexDrugogVrha]))
               {
                   KeyValuePair<string, string> parVrhva = new KeyVa-
luePair<string, string>(izdvojenaVeza[indexPrvogVrha], izdvojenaVeza[in-
dexDrugogVrha]);
```

```
                    rijecnikVeza.Add(parVrhva,int.Parse(izdvojenaVeza[in-
dexTezineVeze]));
                }
                else
                {
                    throw new Exception("Neispravno je unesen graf!!");
                }
            }
        }
```

Nakon unosa težinskoga grafa slijedi *optimiziranje grafa* Kruskalovim algoritmom:

```
        private void optimizacija()
        {
          Dictionary<KeyValuePair<string, string>, int> kopijaVeza = veze;
          KeyValuePair<string, string> kljucMinVeze;

            while (optimiraneVeze.Count < (vrhovi.Count - 1))
            {
                kljucMinVeze = nadiVezuNajmanjeTezine(kopijaVeza);

                bool vrhoviPostoje = vrhovi.Contains(kljucMinVeze.Key)
& vrhovi.Contains(kljucMinVeze.Value);
                bool nemaCiklusa = ispitivanjeCiklusa(kljucMinVeze);

                if (vrhoviPostoje & nemaCiklusa)
                {
                    optimiraneVeze.Add(kljucMinVeze, kopijaVeza[klju-
cMinVeze]);
                    kopijaVeza.Remove(kljucMinVeze);
                }

                else
                {
                    kopijaVeza.Remove(kljucMinVeze);
                }
            }
        }
```

Prema algoritmu, optimizacija se obavlja sve dok stablo ne sadrži $v - 1$ veza. Ako je to istinito, tada se prvo pronalazi veza s najmanjom težinom. Nakon toga ispituje se tvori li ta nova veza s već ranije odabranim vezama - ciklus. Ako ne tvori, tada se odabrana veza dodaje u skup (*Dictionary*) optimiziranih veza te se iz pomoćnog

skupa veza (koji nam služi za pronalazak preostalih veza s najmanjom težinom) izbriše odabrana veza. Ukoliko pak odabrana veza s već ranije odabranim vezama tvori ciklus, tada se ona odbacuje i obriše iz pomoćnog skupa veza.

Metoda za pronalaženje veze s najmanjom težinom izgleda ovako:

```
        private KeyValuePair<string, string> nadiVezuNajmanjeTezine (Dictionary<KeyValuePair<string, string>, int> veze)
        {
            int tezinaTrenutneVeze = 0;
            int minTezina = 0;
            KeyValuePair<string, string> kljucMinVeze = new KeyValuePair<string,string>();

            foreach (KeyValuePair< KeyValuePair<string, string>, int > vrhoviVeze in veze)
             {
                 tezinaTrenutneVeze = vrhoviVeze.Value;

                 if (tezinaTrenutneVeze < minTezina | minTezina == 0)
                 {
                     minTezina = tezinaTrenutneVeze;
                     kljucMinVeze = vrhoviVeze.Key;
                 }
             }

             return kljucMinVeze;
        }
```

Algoritam ide po spremniku svih preostalih veza i provjerava je li težina trenutne veze manja od do tada najmanje težine. Ukoliko jest, tada najmanja težina poprima vrijednost težine te veze, pa se zapamti koja je to veza.

Metoda za pronalaženje ciklusa:

```
      private bool ispitivanjeCiklusa(KeyValuePair<string, string> veza)
      {
         bool nemaCiklusa = false;
         string prviVrhVeze = veza.Key;
         string drugiVrhVeze = veza.Value;

          if (vrhoviZaOptimizaciju[veza.Key] != vrhoviZaOptimizaciju[veza.Value])
         {
             nemaCiklusa = true;
```

```
                int oznakaManjegPodstabla = Math.Min(vrhoviZaOptimizaci-
ju[veza.Key], vrhoviZaOptimizaciju[veza.Value]);

                int oznakaVecegPodstabla = Math.Max(vrhoviZaOptimizaci-
ju[veza.Key], vrhoviZaOptimizaciju[veza.Value]);

                Dictionary<string, int> pomocniVrhoviZaOptimizaciju =
new Dictionary<string, int>();

                foreach (KeyValuePair<string, int> vrh in vrhoviZaOptimi-
zaciju)
                  {
                      if (vrh.Value == oznakaVecegPodstabla)
                      {
                          pomocniVrhoviZaOptimizaciju.Add(vrh.Key, ozna-
kaManjegPodstabla);
                      }
                      else
                      {
                           pomocniVrhoviZaOptimizaciju.Add(vrh.Key, vrh.
Value);
                      }
                  }

                   vrhoviZaOptimizaciju = pomocniVrhoviZaOptimizaciju;
              }
              return nemaCiklusa;
          }
```

Spremnik podataka `vrhoviZaOptimizaciju` sadrži popis svih vrhova, te njima pridružene indekse (na početku svaki vrh ima jedinstven indeks). Ukoliko dva vrha veze imaju različite indekse, to znači da se nalaze u različitim komponentama povezanosti (indeks predstavlja komponentu povezanosti) te da njihovim povezivanjem neće doći do stvaranja ciklusa. Ako vrhovi odabrane veze zadovoljavaju taj uvjet, tada se uzima manji od njihovih indeksa za indeks novonastale komponente povezanosti te se ažuriraju indeksi svih vrhova koji su postali dio te nove komponente. Na kraju, metoda vraća logičku vrijednost postojanja ili nepostojanja ciklusa.

4. Zaključak

U ovom članku dokazana su osnovna svojstva stabala te je kao primjena riješen problem spajanja pomoću Kruskalovog algoritma. Napomenimo da su u završnome radu [1] obrađeni još neki računarski problemi za čije se rješavanje koriste stabla.

Detaljnije o tome čitatelj može naći u [1]. Konačno, možemo zaključiti da su stabla vrlo važan dio teorije grafova, te da je njihovo korištenje učestalo u rješavanju raznih računarskih problema, ali i problema iz stvarnog života.

Jedna od glavnih zadaća tijekom izrade programske implementacije bila je kako napraviti grafičko sučelje programa što jednostavnije za uporabu i bliže korisnicima. Na takav način, studentima elektrotehnike i računarstva i sama matematička pozadina problema postaje zanimljivija.

Na kraju, napomenimo da je ovaj završni rad svakako ispunio zahtjeve koji su bili postavljeni studentu - teorijsku pozadinu i praktičnu primjenu, u ovom slučaju, opisanu programsku podršku. Dakako da sa sličnim projektima treba nastaviti i u budućnosti.

5. Literatura

1. J. Berger: *Svojstva stabala i primjene na neke algoritme*, završni rad, FER, 2011.
2. M. Krnić, S. Šain: *Josipov problem i programska podrška*, Poučak - časopis za metodiku i nastavu matematike, 42, 2010.
3. Z. Mitrović: *Diskretna matematika*, http://www.scribd.com/doc/39328935/DISKRETNA-MATEMATIKA, 2011.
4. M. O. Pavčević: *Uvod u teoriju grafova,* Element, Zagreb, 2009.
5. D. Veljan: *Kombinatorika s teorijom grafova*, Školska knjiga, Zagreb, 1989.
6. *Rezni bridovi i vrhovi,* http://www.math.uniri.hr/~ajurasic/D-PETO.pdf, 2011.