Univerzitet u Beogradu

Matematički fakultet

Algoritmi za crtanje grafova

— Master rad —

autor: Luka Tomašević

mentor: dr Predrag Janičić

Beograd
2008

Sadržaj

Predgovor

Mada se ideja grafova veoma davno pojavila, problem vizualizacije grafova dugo nije bio formalizovan. Zbog značaja grafova, pojava automatizivanih i poluautomatizovanih tehnika za crtanje grafova je veoma važna. Ove tehnike daju crtanje grafa na osnovu opisa grafa. Postoji mnogo različitih vrsta crtanja kako za opšte grafove, tako i za specijalne vrste grafova. U zavisnosti od vrste crtanja i od vrste grafa varira kompleksnost algoritama za crtanje.

U ovom radu će biti opisane tri metode za crtanje grafova (dve metode za crtanje opštih grafova i jedna metoda za crtanje planarnih grafova). U glavi ,,Uvod" je predstavljen uvod u problematiku crtanja grafova. U glavi ,,Osnovni pojmovi i notacija" je dat pregled teorijskih osnova koje su potrebne za rad sa grafovima i njihovo crtanje. U glavi ,,Crtanje opštih grafova" su predstavljene dve metode za crtanje opštih grafova — lučno-slojevita metoda i baricentrična metoda, dok je u glavi ,,Pravolinijsko crtanje planarnih grafova" predstavljena jedna metoda za pravolinijsko crtanje planarnih grafova — metoda pomeraja. Ove tri metode su implementirane u okviru bilbioteke koja je razvijena kao deo ovog rada. Pored samih metoda za crtanje grafova, u ovoj biblioteci je implementiran i veliki broj metoda koji se ne odnosi na crtanje grafova. Detalji implementacija algoritama za crtanje grafova su predstavljeni u glavi ,,Implementacija algoritama za crtanje grafova". U glavi ,,Pregled postojećih alata" je dat pregled alata i biblioteka koji se mogu koristiti za crtanje grafova. Poslednje dve glave čine ,,Zaključci i dalji rad" i ,,Primeri".

Glava 1

Uvod

Graf se sastoji od skupa temena i skupa grana (svaka grana može da spaja najviše dva temena). Ljudi od davnina koriste grafove da bi predstavili ideje, koncepte, strukture, itd. Crtanje grafa se može razmatrati kao crtanje dijagrama, koji se sastoji od skupa objekata koji odgovaraju temenima i linija koje spajaju te objekte. Crtanje grafa je vrsta vizualizacije informacija koje taj graf predstavlja.

1.1 Istorija crtanja grafova

Tačno poreklo grafova nije poznato. Mada se Ojleru-u[1] pripisuju zasluge za prve rezultate teorije grafova 1736. godine, crteži grafova su bili korišćeni i mnogo pre njega. Te 1736. godine, Ojler je rešio problem poznat kao problem sedam mostova Königsberg-a (*eng. Seven Bridges of Königsberg*). Ovaj problem je bio inspirisan stvarnim mestom i situacijom. Naime, grad Königsberg[2] (Slika 1.1 na strani 8), koji je tada pripadao Pruskoj se nalazi na Pregel reci. U okviru grada se nalaze dva ostrva koja su spojena međusobno i sa kopnom sa sedam mostova. Pitanje je bilo da li je moguće kretati se tako da se svaki most pređe tačno jednom i da se dođe ponovo u tačku iz koje se krenulo. Smatra se da je rešenje ovog problema prva teorema u teoriji grafova. Inače, odgovor na ovo pitanje je odričan, jer ne postoji odgovarajući Ojlerov put[3] (*eng. Eulerian circuit*)[4]. U dokazu ove teoreme, Ojler je formulisao problem u okvirima teorije grafova. U prvom koraku je eliminisao sve iz problema osim kopna i mostova, a zatim je zamenio svako deo kopna tačkom, koju je nazvao temenom, a svaki most linijom koju je nazvao granom (Slika 1.2 na strani 8).

Pored ovog problema, Ojler je otkrio formulu koja danas nosi ime po njemu, a koja definiše odnos između broja temena, ivica i strana u ravanskom grafu.

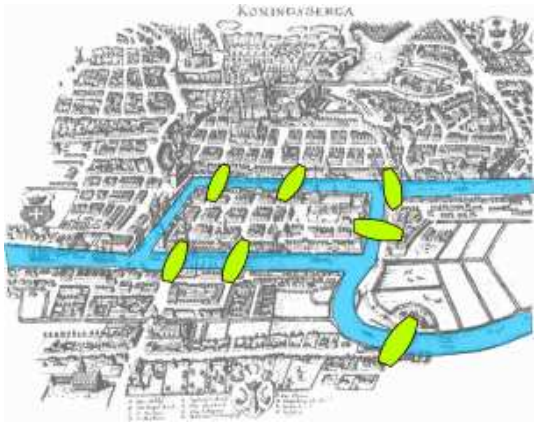
Industrijska potreba za crtanjem grafova se javila kasnih 60-tih godina XX veka, kada je crtanje komponenti velikih elektronskih šema postalo previše kom-

[1] Leonhard Paul Euler (1707 - 1783) — švajcarski matematičar i fizičar.

[2] današnji Kaljiningrad u Rusiji

[3] Ojlerova putanja (*eng. Eulerian path*) je putanja u grafu takva da je svako teme posećeno tačno jednom. Slično Ojlerov put je Ojlerova putanja koja počinje i završava se u istom temenu

[4] Zapravo, od 2005. godine odgovor na ovo pitanje je potvrdan, jer je sagrađen još jedan most



Slika 1.1: Mapa Königsberg-a u Ojlerovo vreme

Slika 1.2: Ojlerova formulacija problema

pleksno za crtanje rukom. Već tada se počelo sa pisanjem algoritama koji bi ovaj zadatak automatizovali. Zanimljivo je da je jedan od problema koji je bio posebno interensatan u to vreme (a i danas je) bio problem kreiranja jednostrane štampane ploče (*eng. single layered PCB (Printed Circuit Board)*). Naime, s obzirom da je kod ove vrste štampanih ploča moguće pravljenje veza samo sa jedne strane potrebno je obezbediti da se veze na njoj ne seku. Ukoliko ovaj problem razmatramo kao problem crtanja grafa, to znači da je potrebno opisati planarni graf koji odgovara električnoj šemi te ploče (u ovu svrhu se koristi ortogonalno crtanje (strana 39)). Krajem 80-ih godina, pojavila se potreba za generisanjem crteža grafova koji će zadovoljiti neke estetske kriterijume, uglavnom u cilju vizuelnog prezentovanja informacija u tehnici i proizvodnom procesu. U poslednjih nekoliko godina, polje crtanja grafova je veoma razvijeno. Od 1993. godine, svake godine se, održava internacionalna konferencija[5] na temu crtanja grafova.

Ukratko, cilj crtanja grafa jeste dobijanje pogodne reprezentacije grafa tako da je struktura grafa lako razumljiva. Uz to, crtanje bi trebalo da zadovolji neke estetske kriterijume bitne za kontekst strukture koju graf reprezentuje.

[5]International Symposium on Graph Drawing

1.2 Primene crtanja grafova

Crtanje grafova ima primene u mnogim oblastima nauke i tehnologije (a i šire). U ovom poglavlju će biti predstavljene neke od oblasti u kojima se grafovi veoma često koriste:

- **Dizajniranje** (*eng. floorplaning*)
  - **VLSI dizajn** — u ovom slučaju, ulaz je ravanski graf S, koji predstavlja funkcionalne entitete čipa, koji se nazivaju modulima, kao i veze među tim modulima. Svako teme grafa S predstavlja modul, a grane među temenima su veze između modula. Izlaz ovog problema za ulazni graf S jeste deljenje pravougaone oblasti poligona u manje pravougaonike. Uz to, ukoliko su dva modula spojena vezom, tada pravougaonici koji njima odgovaraju moraju biti susedni tj. moraju imati zajedničku ivicu (ovo je primer korišćenja pravougaonog crtanja (vidi stranu 40)).
  - **Arhitektonski tlocrt** — i u ovom slučaju radi se o sličnom problemu. Prilikom gradnje nekog objekta, raspored prostorija u njemu može biti prikazan preko grafa, gde svaka soba predstavlja jedno teme, a susednost soba je predstavljena granom grafa. Za ovaj problem se može koristiti pravougaono crtanje (vidi stranu 40) ili box-pravougaono crtanje (vidi stranu 41).
- **Socijalne mreže** - socijalna mreža predstavlja socijalnu strukturu koja se sastoji od temena (koja predstavljaju individue ili organizacije), koja su međusobno povezana jednom ili različitim vrstama veza (koje mogu predstavljati vrednosti, vizije, ideje, prijateljstvo, itd). Rezultujuće strukture su često veoma složene.
- **Kartografija**
- **Simuliranje molekularne strukture** — veoma često se u farmaceutskoj industriji analizira molekularna struktura poređenjem ilustracija njihove 2D strukture. Korišćenjem pogodnih grafova koji odgovaraju ovim strukturama, ovaj zadatak može značajno da se pojednostavi, jer se na taj način sličnosti i razlike između različitih crteža mogu mnogo lakše uočiti.
- **Softverska industrija** — grafovi se veoma često koriste za vizuelizaciju mnogih koncepata u ovoj oblasti. Prilikom dizajna velikih, kompleksnih objektno orijentisanih sistema, grafovima se mogu prikazati struktura problema. Temena u takvim grafovima predstavljaju strukture, klase, pakete, itd. Veze među ovim entitetima su predstavljene granama grafova. Ovi entiteti su najčešće predstavljeni jednostavnim geometrijskim figurama (pravougaonicima, krugovima) različitih boja. Linije su predstavljene na različite načine (različitim bojama, različitim oblikom) da bi na pravi način oslikale odnos između entiteta (nasleđivanje, način korišćenja, itd). Na ovaj način održavanje i upotreba ovako velikih sistema je veoma pojednostavljena. Danas postoje alati koji automatski generišu grafove na osnovu ovakvih struktura.

Vizuelizacija grafovima se veoma često koristi i u dizajnu bazi podataka. Tako na primer, u entitet-odnos modelu[6] (*eng. entity-relationship model*), metodu koji se koristi za semantičko modeliranje baze, graf se koristi da bi se ovaj proces olakšao. Entitet koji predstavlja diskretni objekat je zapravo teme grafa i u zavisnosti od vrste entiteta predstavljen je na različite načine, različitim geometrijskim figurama (pravougaonikom, rombom, elipsom, itd), dok je veza između entiteta grana tog grafa. U zavisnosti od vrste odnosa između entiteta, grane su predstavljene na vizuelno različite načine.

- **Teorija grafova** — i u samoj teoriji grafova je veoma bitno da graf bude ,,lepo" nacrtan. Važno je da crtež bude takav da se sa njega vidi što više strukturalnih osobina grafa - na primer, povezanost, broj ciklusa, da li graf pripada nekoj od poznatih klasa grafova (kompletni, bipartitni, stablo, itd), zatim simetrije, postojanje interesantnih podgrafova, da je graf eventualno dobijen primenom neke od operacija nad grafovima, stepeni temena, da li neka temena imaju mnogo zajedničkih suseda, itd.

- **Ostale nauke** — u praktičnom svim naukama, grafovi se obilato koriste radi prikaza odnosa među entitetma koje te nauke proučavaju: sociologiji, psihologiji, biologiji, istoriji, geografiji, itd. Korišćenje grafova u ovim oblastima omogućava uočavanje odnosa koji bi inače teže bili uočeni ili možda uopšte ne bi bili uočeni.

Ovo su samo neke od oblasti koje grafove koriste da bi olakšale prikaz nekih odnosa među entitetima. U zavisnosti od koncepta problema, razne vrste crtanja se mogu koristiti, bilo u kombinaciji, bilo pojedinačno. Sa razvojem hardvera, kao i različitih algoritama za crtanje grafova, sve je više alata koji ovaj proces mogu automatizovati ili poluautomatizovati.

[6] Model koje je 1976. godine predložio Dr. Pin-Shan Chen.

Glava 2

Osnovni pojmovi i notacija

U ovoj glavi će biti date neke osnovne definicije i teoreme teorije grafova. One će biti korišćene u ostatku rada.

2.1 Grafovi i multigrafovi. Podgrafovi. Putevi i ciklusi. Lanci.

Definicija 2.1.1 Graf *je uređeni par* (V, E) *dva skupa. Elemente skupa* V *nazivamo* temena*, dok su elementi skupa* E *grane. Svaka grana predstavalja par (ne obavezno različitih) temena. Ukoliko su skupovi* V *i* E *konačni, onda i za graf kazemo da je* konačan*. Grana koja je određena jednim istim temenom je* petlja*. Graf kod kog skup* E *sadrži iste elemente nazivamo* multigraf*, dok za granu koja se u tom grafu pojavljuje više puta kažemo da je* višestruka*. Graf bez petlji i višestrukih grana se još naziva* prost graf*.*

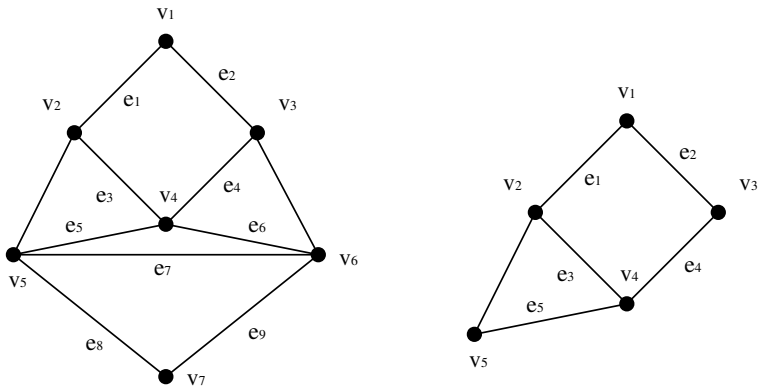
U nastavku ovog rada ćemo grafom nazivati prost graf, ukoliko ne postoji mogućnost zabune.

Kažemo da grana spaja dva temena kojima je (u skladu sa prethodnom definicijom) određena. Često skup temena grafa G označavamo sa $V(G)$, a skupa njegovih grana sa $E(G)$. Broj temena grafa G često označavamo sa n, tj. $n = |V|$.

Granu koja spaja temena u i v iz G označavamo sa (u, v), ili uv. Ako važi $uv \in E$, tada kažemo za temena u i v da su susedna (*eng. adjacent*) u G, a za granu uv kažemo da je *incidentna* (*eng. incident*) sa temenima u i v. Takođe, za teme u kažemo da je *sused* temena v i obratno. *Stepen* (*eng. degree*) $d(v, G)$ temena v u grafu G je broj grana incidentnih sa v u grafu G. Često se radi jednostavnosti stepen temena označava sa $d(v)$. Sa $\Delta(G)$ označavamo maksimalni stepen svih temena u G i zovemo ga *maksimalni stepen* grafa G.

Primer 2.1.1 *Graf* G *na slici 2.1(a) je graf koji ima 7 temena i 9 grana. Važi* $d(v_1) = 2$*. Temena* v_1 *i* v_2 *su susedna temena. Grana* e_1 *je incidentna temenima* v_1 *i* v_2*. Važi* $\Delta(G) = 4$*.*

Definicija 2.1.2 Podgraf *grafa* $G = (V, E)$ *je graf* $G' = (V', E')$*, takav da je* $V' \subseteq V$ *i* $E' \subseteq E$*. Ako* G' *sadrži sve grane iz* G *koje spajaju temena iz* V'*, tada* G' *nazivamo* podgrafom indukovanim sa V' *(eng. subgraph induced by* V'*). Ako*



Slika 2.1: (a) Graf G; (b) Podgraf G' grafa G indukovan temenima v_1, v_2, v_3, v_4, v_5

je $V' = V$, tada G' zovemo povezujućim podgrafom *(eng. spanning subgraph) od G.*

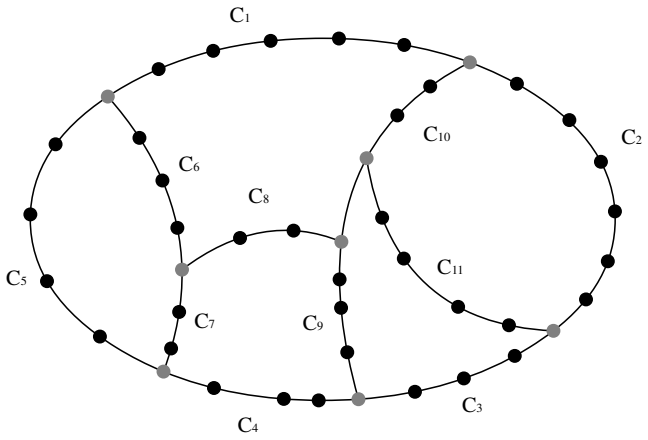
Primer 2.1.2 *Na slici 2.1(b) je prikazan graf G' koji je podgraf grafa G indukovan temenima v_1, v_2, v_3, v_4, v_5.*

Često nove grafove konstruišemo tako što od postojećih brišemo temena. Ako je v teme grafa $G = (V, E)$, tada je $G - v$ podgraf od G, koji je dobijen brisanjem temena v i svih grana incidentnih sa v iz grafa G. Opštije, ukoliko je V' podskup od V, tada je $G - V'$ podgraf od G koji je dobijen brisanjem svih temena iz V' i svih grana koje su incidentne sa temenima iz V' iz grafa G. Prema tome, $G - V'$ je podgraf od G indukovan sa $V - V'$. Slično, ukoliko je e grana grafa G, tada je $G - e$ podgraf od G dobijen brisanjem grane e. Opštije, ukoliko je $E' \subseteq E$, tada je $G - E'$ podgraf od G, dobijen iz G brisanjem svih grana iz E.

Definicija 2.1.3 Šetnja *(eng. walk)* $v_0, e_1, v_1, \ldots, v_{l-1}, e_l, v_l$ *u grafu G je naizmenični niz temena i grana iz G, koji počinje i završava se temenom, i u kojem je svaka grana incidentna dvema temenima — temenu koje je neposredno ispred grane i temenu koje je neposredno iza grane. Ako su temena $v_0, v_1, \ldots, v_l$ različita (osim eventualno temena v_0, v_l), tada šetnju nazivamo* putem *(eng. path) i ona se obično označava ili nizom temena $v_0, v_1, \ldots, v_l$ ili nizom grana $e_1, e_2, \ldots, e_l$. Dužina puta je l, za jedan manja od broja temena u putu. Put, odnosno šetnja je* zatvorena *ukoliko je $v_0 = v_1$. Zatvoreni put koja sadrži barem jednu granu se naziva* ciklusom *(eng. cycle).*

Definicija 2.1.4 *Neka je $P = v_0, v_1, v_2, \ldots, v_{l+1}$, $l \geq 1$ put grafa G takav da je $d(v_0) \geq 3, d(v_1) = d(v_2) = \ldots = d(v_l) = 2$ i $d(v_{l+1}) \geq 3$. Tada potput $P' = v_1, v_2, \ldots, v_l$ puta P nazivamo* lancem *od G (eng. chain). Temena v_0 i v_{l+1} nazivamo* nosačima *(eng. supports) lanca P'.*

Definicija 2.1.5 *Za dva lanca kažemo da su* susedna *ako imaju zajedničke nosače.*



Slika 2.2: Ravanski graf G koji sadrži 11 lanaca

Primer 2.1.3 *Na slici 2.2 se nalazi ravanski graf G (Definicija 2.3.2 na strani 17) koji sadrži 11 lanaca ($C_1 - C_{11}$). Temena koja predstavljaju nosače lanaca su obojena svetlijom bojom. Lanci C_1 i C_2 su susedni lanci.*

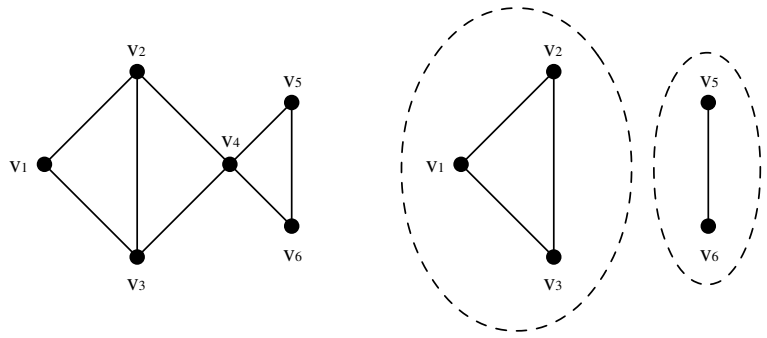
2.2 Povezanost. Stabla i šume. Kompletni i bipartitni grafovi.

Definicija 2.2.1 *Graf G je* povezan *(eng. connected), ukoliko za proizvoljna dva temena u i v postoji put između u i v u G. Graf koji nije povezan je* nepovezan *(eng. disconnected).* (Povezana) komponenta *(eng. connected component) grafa G je njegov maksimalni povezani podgraf.*

Primer 2.2.1 *Na slici 2.3(a) je dat primer jednog povezanog grafa G. Na slici 2.3(b) je dat primer nepovezanog grafa $G - v_4$ (dobijen brisanjem temena v_4), koji ima dve povezane komponente.*

Definicija 2.2.2 Povezanost *(eng. connectivity) $k(G)$ grafa G je minimalni broj temena, čije brisanje rezultuje nepovezanim grafom ili grafom koji se sastoji od samo jednog temena K_1 (eng. single-vertex graph). Za graf G kažemo da je* k-povezan *(eng. k-connected), ukoliko važi $k(G) \geq k$.*

Definicija 2.2.3 *Skup temena u povezanom grafu G nazivamo* separatorom *(eng. separator) ili (eng. vertex-cut), ukoliko brisanje tih temena iz grafa G rezultuje nepovezanim grafom ili grafom koji se sastoji samo od jednog temena. Ako separator sadrži samo jedno teme, tada to teme nazivamo* teme razdvajanja *(eng. cut-vertex). Ukoliko separator sadrži dva temena, tada ih nazivamo* par razdvajanja *(eng. separation pair).*



Slika 2.3: (a) povezan graf G; (b) nepovezan graf $G - v_4$ sa dve povezane komponente (zaokružene su isprekidanim linijama)

Definicija 2.2.4 Stablo *je povezani graf bez ijednog ciklusa.*

Korensko stablo (*eng. rooted tree*) je stablo u kome se jedno teme razlikuje od ostalih. To teme se naziva *koren* (*eng. root*) stabla. Koren stabla se obično crta na vrhu. Ako korensko stablo razmatramo kao usmereni graf kod koga je svaka grana usmerena odozgo na dole, tada je svako teme u različito od korena povezano sa nekim drugim temenom p, koje nazivamo *roditeljem* (*eng. parent*). Teme u nazivamo *detetom* (*eng. child*) temena p. Roditelja nekog temena crtamo iznad tog temena. *List* (*eng. leaf*) je teme stabla koje nema nijedno dete. *Unutrašnje teme* (*eng. internal vertex*) je teme koji ima jedno ili više dece. Prema tome, svako teme u stablu je ili list ili unutrašnje teme. Odnos dete-roditelj, može biti proširen na pretke i potomke. Pretpostavimo da je $u_1, u_2, \ldots, u_l$ niz temena u stablu i da važi da je u_1 roditelj od u_2, koje je roditelj od u_3 i tako dalje. Tada teme u_1 zovemo *pretkom* temena u_l, a teme u_l nazivamo potomkom temena u_1. Koren je predak svakog temena u stablu i svako teme je potomak korena.

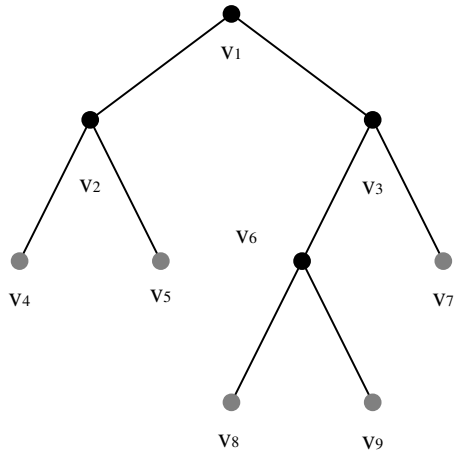
Primer 2.2.2 *Na slici 2.4 je dat primer jednog stabla. Koren stabla je v_1. Teme v_2 je dete temena v_1, dok je v_1 roditelj v_2, v_8 je potomak v_3, v_3 je unutrašnje teme. Listovi stabla su obojeni svetlije.*

Definicija 2.2.5 *Graf bez ciklusa se naziva* šuma.

Svaka povezana komponenta šume je stablo. Povezujući podgraf grafa G se naziva *stablo* grafa G, ukoliko je taj podgraf stablo. Drvo povezanog grafa G se naziva *povezujuće stablo* (*eng. spanning tree*) grafa G.

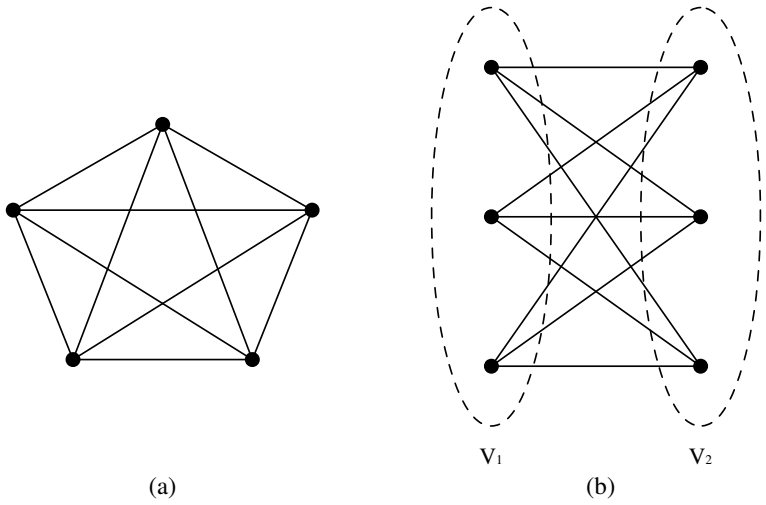
Definicija 2.2.6 *Graf kod kojeg je svaki par različitih temena susedan se naziva* kompletan *graf (eng. complete graph). Kompletni graf sa n temena se označava sa K_n.*

Definicija 2.2.7 *Neka se skup temena V grafa $G = (V, E)$ može podeliti u dva disjunktna skupa V_1 i V_2, tako da svaka grana grafa G spaja teme iz skupa*



Slika 2.4: Primer stabla

V_1 *sa temenom iz skupa* V_2. *Graf* G *u tom slučaju zovemo* bipartitan graf *(eng. bipartite graph). Ako je svako teme iz skupa* V_1 *spojeno sa svakim temenom iz skupa* V_2, *tada graf* G *zovemo* kompletnim bipartitnim *grafom (eng. complete bipartite graph) i označavamo ga sa* $K_{s,r}$, *gde je* $s = |V_1|$, *a* $r = |V_2|$.

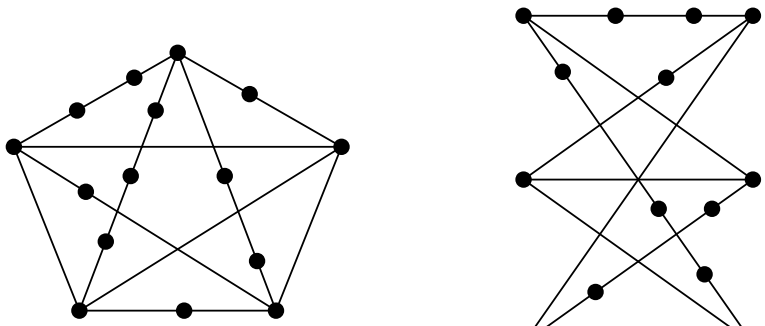


Slika 2.5: (a) kompletan graf K_5; (b) bipartitan graf $K_{3,3}$

Primer 2.2.3 *Na slici 2.5(a) je prikazan kompletni graf* K_5, *dok je na slici 2.5(b)* $K_{3,3}$ *prikazan bipartitni graf* $K_{3,3}$.

Definicija 2.2.8 Razlaganje *(eng. subdividing) grane* (u, v) *grafa* G *je operacija u kojoj se grana* (u, v) *briše, a umesto nje se dodaje put* $u(= w_0), w_1, w_2, \ldots, w_k$, $v(= w_{k+1})$, *kroz nova temena* $w_1, w_2, \ldots, w_k$, $k \geq 1$, *stepena dva.*

Definicija 2.2.9 *Za graf G kažemo da je* razlaganje *(eng. subdivision) grafa G', ukoliko je G dobijen od grafa G' razlaganjem nekih grana iz G'.*



(a) (b)

Slika 2.6: (a) razlaganje grafa K_5; (b) razlaganje grafa $K_{3,3}$

Primer 2.2.4 *Na slici 2.6(a) je dat primer razlaganja grafa K_5, dok je na slici 2.6(b) dat primer razlaganja grafa $K_{3,3}$.*

2.3 Planarni grafovi. Ojlerova formula.

Definicija 2.3.1 *Graf je* planaran *ako može biti utopljen (eng. embedded) u ravan, na takav način da mu se nikoje dve grane ne seku, osim u temenu kome su obe incidentne.*

Treba primetiti da planarni graf može da ima eksponencijalni broj utapanja.

Jedna od najlepših teorema u teoriji grafova je teorema Kuratovskog[1], koja daje karakterizaciju planarnih grafova preko ,,zabranjenih grafova". Sama karakterizacija je iznenađujuće jednostavna.

Teorema 2.3.1 (Kuratovski 1930) *Graf je planaran akko ne sadrži razlaganje od K_5, niti razlaganje od $K_{3,3}$.*

Kao što se može videti iz ove teoreme, da bi se pokazalo da je graf planaran potrebno je i dovoljno dokazati da ne sadrži razlaganje K_5, kao ni razlaganje $K_{3,3}$. Primena ove teoreme nije praktična, jer bi algoritam koji bi proveravao da li je graf planaran koristeći ovu teoremu radio u eksponencijalnom vremenu, međutim ova teorema je bila prva koja je dala karakterizaciju planarnih grafova.

Svojevremeno je, ova teorema bila poznata pod imenom teorema Pontrjagin[2]-Kuratovski, zbog stava da je dokaz ove teoreme prvi dao Pontrjagin u nekim svojim neobjavljenim radovima.

[1] Kazimierz Kuratowski (1896 - 1980), poljski matematičar
[2] Lev Semenovich Pontryagin (1908 - 1988), ruski matematičar

Definicija 2.3.2 Ravanski graf *(eng. plane graf) je planarni graf sa fiksiranim utapanjem u ravni.*

Ravanski graf možemo razmatrati kao planarni graf koji je već nacrtan u ravni. Ravanski graf takođe možemo posmatrati i kao planarni graf sa definisanim preslikavanjima iz skupa temena u skup tačaka u ravni i iz skupa grana u skup krivih u ravni, takvih da su krajnje tačke tih krivih dobijene preslikavanjem temena koje te krive spajaju. Pri tom te krive nemaju zajedničkih tačaka osim krajnjih tačaka. Ravanski graf deli ravan na regione koji se nazivaju *strane* (*eng. faces*). Neomeđena oblast ravni se naziva *spoljašnja strana* (*eng. outer face*) grafa G. Granica strane je u opštem slučaju zatvorena šetnja, a ukoliko je G 2-povezan i ima najmanje tri temena, tada je ciklus. Granica spoljašnje strane se naziva *spoljašnja granica* (*eng. outer boundary*) od G i označava se sa $C_0(G)$. Ukoliko je $C_0(G)$ ciklus, tada se naziva *spoljašnji ciklus* (*eng. outer cycle*). Teme v na grafu G nazivamo *spoljašnjim temenom* (*eng. outer vertex*) ukoliko je $v \in C_0(G)$. U suprotnom, v je *unutrašnje teme* (*eng. inner vertex*) od G. Na sličan način definišemo *spoljašnju ivicu* (*eng. outer edge*) i *unutrašnju ivicu* (*eng. inner edge*) grafa G.

Prirodno pitanje koje se nameće u radu sa grafovima je u kakvom su odnosu elementi jednog grafa: temena, grane i strane. Kao što je već ranije pomenuto, Ojler je dokazao teoremu koja se bavi ovim pitanjem.

Teorema 2.3.2 (Ojler 1750) *Neka je G povezani ravanski graf i neka su sa n, m i f redom označena temena, grane i strane grafa G. Tada važi:*

$$n - m + f = 2$$

Dokaz: Dokaz izvodimo indukcijom po m. Pretpostavimo da G ima m grana. Ovde razlikujemo dva sličaja. Tvrđenje očigledno važi za $m = 0$ i $m = 1$. Pretpostavimo da je $m \geq 2$ i da je tvrđenje tačno za sve povezane ravanske grafove koji imaju manje od m grana.

Ukoliko je G drvo, tada G ima teme v, za koje važi da je $d(v) = 1$. Povezani ravanski graf $G - v$ ima $n - 1$ temena, $m - 1$ grana i $f = 1$ strana, pa na osnovu induktivne pretpostavke važi:

$$(n - 1) - (m - 1) + f = 2$$

odakle sledi:

$$n - m - f = 2$$

Ukoliko G nije drvo, znači da G ima granu e koja se nalazi na ciklusu. U tom slučaju povezani ravanski graf $G - e$ ima n temena, $m - 1$ ivica i $f - 1$ strana, pa direktno iz induktivne pretpostavke sledi:

$$n - m + f = 2$$

□

Definicija 2.3.3 *Za planarni graf G kažemo da je* maksimalan*, ukoliko ne može da mu se doda ni jedna grana, a da on ostane planaran.*

Iz ove definicije neposredno sledi da je u svakom ravanskom utapanju maksimalnog planarnog grafa sa $n \geq 3$ temena, granica svake strane trougao, pa se stoga utapanje ovakvog grafa naziva *triangulirani planarni graf*. Mada graf u opštem slučaju može imati do $n(n-1)/2$ grana, to ne važi za planarne grafove.

Pored Ojlerove teoreme (Teorema 2.3.2 na strani 17) i sledeća teorema se bavi odnosom temena i ivica.

Teorema 2.3.3 *Neka je G planaran graf sa $n \geq 3$ temena i m grana. Tada važi:*

$$m \leq 3n - 6$$

Jednakost važi ukoliko je G maksimalni planarni graf.

Dokaz: Bez smanjivanja opštosti možemo pretpostaviti da je G maksimalni planarni graf (u suprotnom maksimalni planarni graf možemo dobiti dodavanjem ivica, bez dodavanja temena, sve dok ne dobijemo maksimalni planarni graf).

Razmatrajmo ravansko utapanje grafa G. Svaka strana je omeđena sa tačno tri ivice, dok je svaka ivica između tačno dve strane. Iz toga sledi da je $3f = 2m$. Primenom teoreme 2.3.2 dobijamo:

$$m = 3n - 6$$

□

2.4 Strukture podataka za reprezentovanje grafa

U ovom poglavlju će biti predstavljeni različiti načini reprezentovanja grafa, kao i njihove mane i prednosti. Algoritmi i strukture podataka koje ovde budu predstavljene će biti predstavljene na nivou pseudojezika, dok će sama implementacija ovih metoda i struktura podataka biti predstavljena kasnije u radu.

Pre nego što bude predstavljene strukture podataka za reprezentovanje grafova, uvešćemo nekoliko koje su nam neophodne za to. Ove strukture podataka ćemo kasnije koristiti u implementaciji algoritama.

Vektor je skup promenljivih koje se obično čuvaju kao (1-dimenzioni) niz, dok *matrica* 2-dimenzioni niz. Glavna prednost rada sa vektorima (i matricama) je u činjenici da se njihovim elementima može direktno pristupati preko indeksa, koji jednoznačno određuje element.

Lista je kao strukturu podataka koja se sastoji od homogenih slogova koji su linearno povezani. U zavisnosti od vrste lista, svaki slog sadrži jednu ili više struktura (promenljivih) u kojima se skladište sami podaci, kao i jedan ili više pokazivača. Razlikujemo razne vrste listi:

- *jednostruko povezane*
- *dvostruko povezane*
- *kružne*

U nastavku teksta će posebno biti korišćene dvostruko povezane liste. Kod ovih listi, svaki slog sadrži pokazivač na prethodni, kao i pokazivač na sledeći slog. Ovo svojstvo nam omogućava da možemo dodati nove elemente ili obrisati prethodne, bez poznavanja pozicije prethodnog elementa.

Dve specifične vrste jednostruko povezanih lista su:

- *stek* (*eng. stack*)
- *red* (*eng. queue*)

Stek je lista kod koje je dozvoljeno unošenje i brisanje elemenata samo sa jednog kraja te liste koji se naziva *vrh steka*. Stek se često naziva i *LIFO* lista (*eng. last in, first out*), jer važi pravilo da element koji je poslednji unet je element koji se prvi briše. Za razliku od steka, red je lista kod koje je unošenje novih elemenata dozvoljeno samo na jednom kraju, koji se naziva *rep reda* (*eng. tail*), dok je brisanje dozvoljeno samo na drugom kraju, koji se naziva *glava reda* (*eng. head*). Iz ovog razloga, redovi se često nazivaju i *FIFO* listama (*eng. first in, first out*). Obe ove vrste listi (i stek i red) se realizuju pomoću pokazivača koji pokazuju na vrh, rep ili glavu liste (u zavistnosti od vrste liste koju koristimo).

Složenost nekog algoritma definišemo kao funkciju ulaza algoritma. Postavlja se pitanje šta razmatrati kao ulaz algoritma koji radi sa grafovima. Da bi reprezentovali graf u računaru moramo ga prvo kodirati kao niz simbola iz fiksirane azbuke. U tom slučaju dužina tog niza je ulaz algoritma koji radi sa grafovima.

Sada dolazimo do pitanja kako kodirati graf u neki niz simbola neke fiksirane azbuke, odnosno kako predstaviti graf u računaru. Mada graf možemo predstaviti na mnogo različitih načina, dva načina reprezentovanja se najviše koriste:

- *matrica susedstva* (*eng. adjacency matrix*)
- *lista susedstva* (*eng. adjacency list*)

Graf $G = (V, E)$ možemo predstaviti matricom susedstva $A = (a_{i,j})$, koju konstruišemo tako da važi:

$$\begin{aligned} a_{i,j} &= 1, \quad \text{za } (v_i, v_j) \in E \\ a_{i,j} &= 0, \quad \text{inače} \end{aligned}$$

Postoje prednosti i mane rada sa matricama susedstva. Na primer, ukoliko je G prost graf, tada su elementi na glavnoj dijagonali matrice A svi jednaki nuli, a ona je simetrična. Mana ove reprezentacije je velika potrebna memorija. Naime, za predstavljanje grafa od n temena, potreban je prostora veličine $O(n^2)$. Naravno, ovo je veoma neekonomično, pogotovo ukoliko graf nema mnogo grana. Takođe, pretraga svih suseda temena v zahteva n koraka, čak i ukoliko je stepen $d(v)$ znatno manji od n.

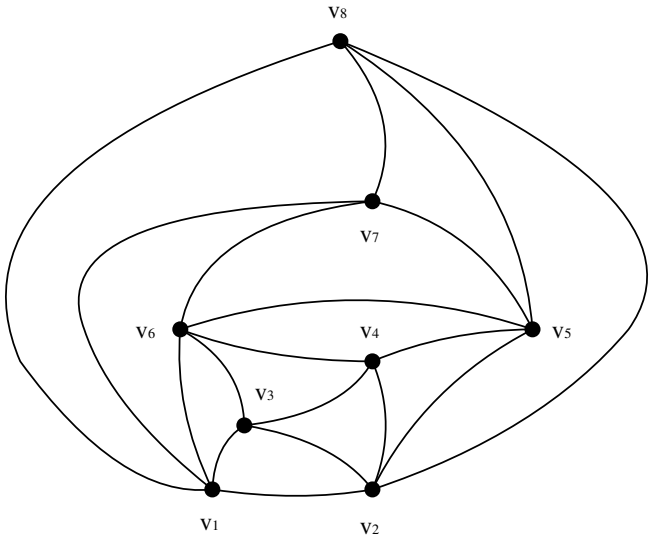
S druge strane, prednost ove reprezentacije je činjenica da je odgovor na pitanje da li grana $(v_i, v_j) \in E$ direktan, odnosno sve što treba proveriti je da li je $a_{i,j} = 1$. Ukoliko jeste, to znači da $(v_i, v_j) \in E$. Slično, brisanje grane (v_i, v_j) je takođe trivijalno, sve što treba uraditi je postavljanje $a_{i,j}$ na 0.

Drugi način predstavljanja grafa je preko listi susedstva. Ovaj način se češće koristi, jer je često znatno bolji za rad sa grafovima. Listu susedstva konstruišemo tako što za svako teme $v \in V$, pamtimo i skup svih suseda $N(v)$. Ovi skupovi se pamte kao liste (primer 2.4.1 na strani 20) $Adj(v)$. Za razliku od matrica susednost, prostor potreban za listu susedstva je:

$$O(\sum_{v \in V}(1 + d(v))) = O(n + m)$$

Očigledno je da je ova reprezentacija grafova mnogo ekonomičnija, što posebno dolazi do izražaja ukoliko graf nema mnogo grana. Pored toga, pretraga svih suseda jednog temena se može obaviti u $d(v)$ koraka kroz listu $Adj(v)$.

Primer 2.4.1 *Na slici 2.7 je prikazan graf G. Kao što je rečeno, on može biti reprezentovan matricom susedstva (slika 2.9) ili listom susedstva (slika 2.8).*



Slika 2.7: Graf G sa 8 temena

U svakom slučaju, od vrste problema koju treba rešavati može se koristi jedna odnosno druga struktura podataka. U biblioteci koja dolazi uz ovaj rad moguće je raditi i sa matricama susedstva i sa listama susedstva.

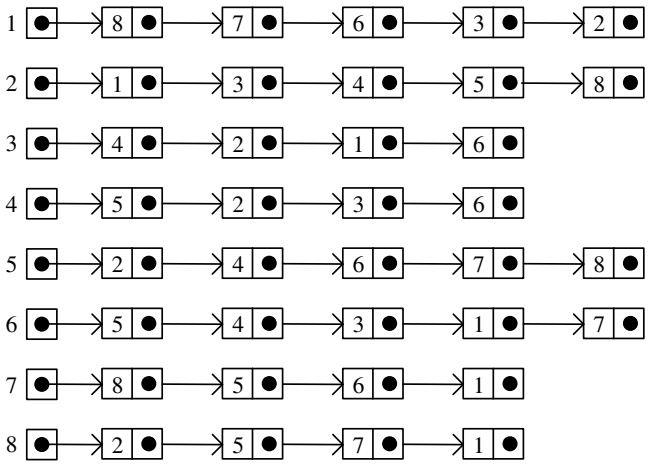
Postavlja se pitanje koju veličinu koristiti za aproksimaciju grafa. S obzirom da možemo pretpostaviti da je $m \geq n/2$, m možemo koristiti za aproksimaciju veličine grafa. U slučaju planarnih grafova, za aproksimaciju veličine grafa se najčešće koristi n, jer na osnovu teoreme 2.3.3, važi $m \leq 3n$. Prema tome, za analizu kompleksnosti planarnih grafova koristimo n, a za analizu grafova u opštem slučaju koristimo i n i m.

Kao što ćemo videti u sledećim glavama najviše što možemo očekivati od algoritma za rad sa grafovima je da bude linearne kompleksnosti $O(n + m)$.

2.5 Obilazak grafa. DFS i BFS obilazak.

Algoritmi za obilazak grafova predstavljaju neke od osnovnih algoritama za rad sa grafovima. Pored činjenice da se mogu koristiti za prolazak kroz sva temena grafa, oni se koriste i za neke druge specifične algoritme. Postoje dva osnovna algoritma za obilazak grafa:

- *DFS* (*eng. Depth First Search*) — obilazak grafa po dubini;
- *BFS* (*eng. Breadth First Search*) — obilazak grafa po širini.



Slika 2.8: Lista susedstva grafa G

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Slika 2.9: Matrica susedstva grafa G

Kod oba ova metoda, svaka grana se prolazi tačno jednom u oba smera (kada se prvi put ,,posećuje" neko teme i kada se ,,vraća" od tog temena). Prema tome, oba algoritma su linearne složenosti.

Osnovna ideja DFS algoritma je da se temena obilaze na sledeći način. Odaberemo početno teme v. Pretpostavimo da je x poslednje posećeno teme. Pretragu nastavljamo odabirom neke neistražene grane (x, y), koja je incidentna sa x. Ukoliko je y teme koje je već obiđeno, tada biramo neku drugu granu koja je neistražena, a incidentna je sa x. Ukoliko teme y nije obiđeno, tada ga obilazimo i pretragu nastavljamo od temena y. Pošto završimo pretragu kroz sve puteve koje počinju u y, pretragu vraćamo na teme x — teme sa koga je y i prvi put obiđeno. Algoritam se nastavlja sve dok ima neobiđenih temena. Na osnovu opisa algoritma, jasno je zašto se zove algoritam pretrage grafa po dubini — sve dok je to moguće, po dubini se rekurzivno biraju nova (neobiđena) temena, kada to više nije moguće, algoritam se vraća unazad iz rekurzije i tako sve dok se ne obiđu sva temena.

Kada se DFS algoritam primeni na neusmereni graf $G = (V, E)$, ovaj algoritam skup grana E deli na dva skupa: T i B. U opštem slučaju skup T predstavlja povezujuću šumu od G. Ukoliko je G povezan, T predstavlja povezujuće stablo.

Granu (x, y) smeštamo u T ukoliko je teme y obiđeno po prvi put iz temena x. Grane iz T nazivamo *grane stabla* (*eng. tree edges*). Sve preostale grane smeštamo u B. Te ivice iz skupa B zovemo *povratne grane* (*eng. back edges*).

Algoritam 2.5.1 *DFS algoritam za obilazak grafova*

```
procedure DFS();
begin
     T=0;
     for each v in V do
          oznaci v sa "novo";
     while postoji v in V oznaceno sa "novo" do
          Trazi(v);
end

procedure Trazi(v);
begin
     oznaci v sa "staro";
     for each w in Adj(v) do
     begin
          if w je oznaceno sa "novo" then
          begin
               Dodaj (v,w) u T;
               Trazi(w);
          end;
     end;
end;
```

Za razliku od DFS algoritma za obilazak grafa, koji ide po dubini, odnosno obilazak grafa „ide" rekurzivno dok god je to moguće, BFS algoritam obilazi graf na način koji je više sistematski. Temena se obilaze po nivoima.

Na početku biramo proizvoljno teme i upisujemo ga u red temena Q koja treba obići. Tokom rada algoritma, brišemo teme x sa reda Q i pretražujemo listu susedstva $Adj(x)$. Tom prilikom u red Q unosimo sve susede od x koji nisu bili do sada unešeni u Q. Radi jednostavnosti, možemo pretpostaviti da je graf povezan. U suprotnom, BFS algoritam primenjujemo na (povezane) komponente grafa. BFS algoritam deli skup grana E povezanog grafa u povezujuće stablo T i skup povratnih grana B.

Algoritam 2.5.2 *BFS algoritam za obilazak grafova*

```
procedure BFS();
begin
     T=0;
     Q=prazan red;
     for each v in V do
          oznaci v sa "nije dostignuto";
     Odaberi proizvoljno teme r iz V kao pocetno teme
        i dodaj r u Q;
     oznaci r sa "dostignuto";
     LEVEL(r)=1;
```

```
     while Q nije prazan do
     begin
          x=glava(Q);
          Q=Q-x;
          Pretrazi(x);
     end;
end;

procedure Pretrazi(x)
begin
     for each y in Adj(x) do
     begin
          if y je oznaceno sa "nije dostignuto" then
          begin
               Dodaj granu (x,y) u T;
               LEVEL(y)=LEVEL(x)+1;
               Dodaj y u Q;
               Oznaci y sa "dostignuto";
          end;
     end;
end;
```

Vidimo da se kod ovog algoritma pretraga grafa odvija na drugi način — pretraga po širini. Prvo se pretraže sva temena na jednom nivou. Tek kada su sva temena na jednom nivou posećena, prelazi se na sledeći nivo i tako dok se ne obiđe svako teme. Ovaj algoritam je za razliku od rekurzivnog DFS algoritma iterativne prirode.

T drvo koje se dobija tokom rada BFS algoritma ima sledeća svojstva:

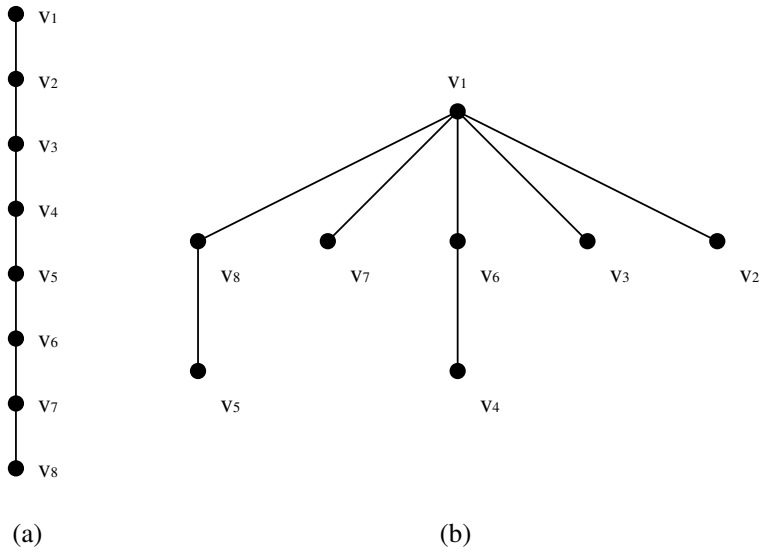
- svaka grana iz G, bez obzira da li pripada T ili B skupu spaja temena čiji se nivoi razlikuju najviše za jedan;
- nivo temena v jednak je dužini (broju grana) najkraćeg puta od korena r do v.

2.6 Strukture podataka za reprezentovanje planarnih grafova

Mada se za rad sa planarnim grafovima mogu koristiti iste strukture kao i za rad sa grafovima u opštem slučaju, postoje neke modifikacije ovih struktura koje omogućavaju efikasniji rad sa planarnim grafovima. Ove modifikacije se odnose na rad sa listama susedstva.

Primer 2.6.1 *DFS i BFS drveta za graf G iz primera 2.4.1 su data na slici 2.10.*

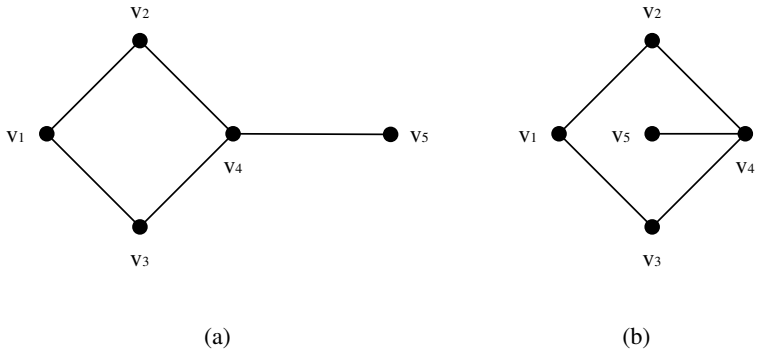
Razmotrimo dva grafa data na slici 2.11. Oba grafa imaju istu matricu susedstva, datu na slici 2.12. Mada ova dva grafa jesu identična u matematičkom smislu, očigledno je da ova dva crtanja nisu identična. Postavlja se pitanje da



Slika 2.10: (a) DFS drvo; (b) BFS drvo

li možemo razlikovati ova dva crtanja na osnovu liste susedstva. Lista susedstva grafa 2.11(a) je data na slici 2.13. Očigledno da ovom listom susedstva ne možemo napraviti razliku između ova dva crtanja. Kako listom susedstva možemo jednoznačno predstaviti planarni graf, preciznije konkretno utapanje planarnog grafa? Možemo primetiti da ukoliko sačuvamo poredak grana incidentnih temenu (u smeru kazaljke na satu) u ravanskom utapanju grafa da tada reprezentacija grafa zaista odgovara utapanju.

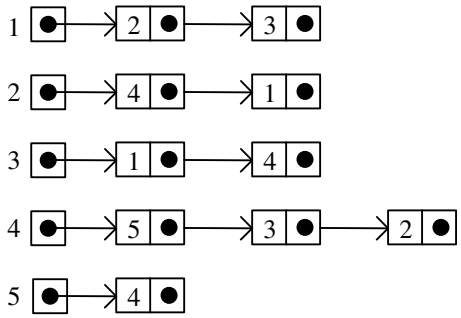
U mnogim algoritmima koji se koriste za crtanje planarnih grafova, često je potrebno obići sve strane ravanskog grafa efikasno. U tom cilju je potrebno imati mogućnost obilaska grana u smeru kazaljke na satu i u smeru suprotnom smeru kazaljke na satu (u kontekstu crteža koji se generiše). Takođe, potrebno je i da je moguće brzo se vratiti na početak liste. Trenutna struktura listi susedstva



Slika 2.11: 2 izomorfna grafa

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Slika 2.12: Matrica susedstva grafa sa slike 2.11

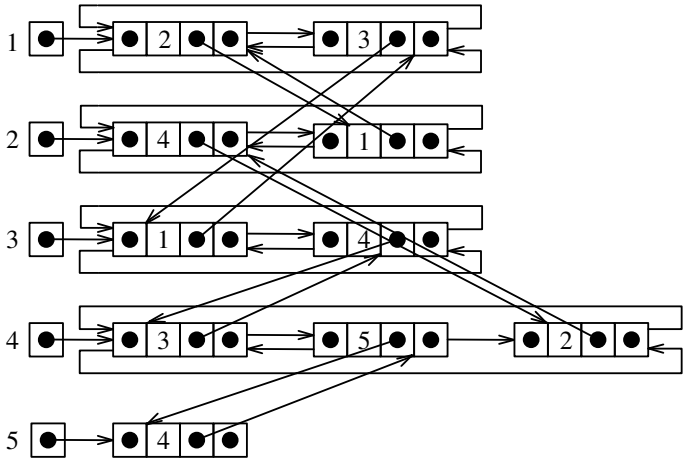


Slika 2.13: Lista susedstva grafa 2.11(a)

nam to ne omogućava. Iz tog razloga, uvodimo novu strukturu podataka, koja je ilustrovana na slici 2.14 i koja obuhvata i listu susedstva grafa sa slike 2.11(b). U ovoj strukturi podataka čuva se poredak grana u smeru kazaljke na satu i poredak grana suprotan smeru kazaljke na satu pomoću dvostruko povezane kružne liste suseda temena. Obilazak ove liste u jednom, odnosno drugom smeru rezultuje dobijanjem poretka grana u smeru kazaljke na satu, odnosno poretka grana suprotnom smeru kazaljke na satu. Na ovaj način je moguće obići sve strane grafa i u smeru kazaljke na satu i u smeru suprotnom od smera kazaljke na satu. Pored povezivanja svih elemenata koji treba da se povežu u dvostruko povezanoj ulančanoj listi, postoji dodatna veza između temena, koja omogućava direktni pristup.

Definicija 2.6.1 *Neka je $G = (V, E)$ planarni graf sa skupom temena V i skupom grana E. Graf možemo reprezentovati sa n listi susedstva, gde je $n = |V|$. Lista $Adj(v)$ za teme $v \in V$ sadrži sva susedna temena temena v. Za svako teme $v \in V$, konkretno crtanje planarnog grafa G određuje, do na cikličnu permutaciju, poredak suseda temena v.* Utapanje *planarnog grafa G predstavlja konstrukciju liste susedstva grafa G, tako da se u svakoj listi $Adj(v)$, svi susedi temena v pojavljuju u smeru kazaljke na satu. Takav skup Adj listi susedstva nazivamo* ravanskim utapanjem *(eng. planar embedding) grafa G.*

Ravansko utapanje je struktura podataka koja predstavlja listu susedstva, na takav način da su u svakoj listi, sve grane incidentne sa datim temenom uređene (navedene u odgovarajućem poretku u odnosu na crtež).



Slika 2.14: Lista susedstva grafa 2.11(b)

Glava 3

Crtanje opštih grafova

U prethodnoj glavi su definisani osnovni pojmovi potrebni za crtanje grafova. U ovoj glavi ćemo ćemo dati pregled različitih vrsta crtanja opštih neusmerenih grafova.

Geometrijska reprezentacija grafova je bila tema kojom su se matematičari bavili vekovima zbog intuitivnijeg predstavljanja raznih struktura. Knutov[1] [8] rad iz 1963. godine o crtanju dijagrama toka je verovatno prvi rad u kome je predstavljen neki algoritam za crtanje grafova.

Algoritmima za crtanje opštih grafova se mogu predstaviti sve klase grafova. Kao takve, ove metode imaju veoma značajnu ulogu u ovoj oblasti. U ovoj glavi ćemo predstaviti dve metode za crtanje opštih grafova: lučno-slojevitu metodu i baricentričnu metodu. Radi se o dve metode koje daju veoma različita crtanja. Lučno-slojevita metoda spada u grupu hijerarhijskih metoda, dok baricentrična metoda spada u grupu metoda usmeravanja silom. Obe metode se koriste za crtanje povezanih neusmerenih grafova.

Više detalja o raznim metodama za crtanje grafova se može naći u [2].

3.1 Stilovi crtanja

Postoje različiti grafički standardi za vizualno predstavljanje grafova (Definicija 2.1.1 na strani 11) u zavisnosti od primena samog crtanja. Najčešće se temena predstavljaju kao krugovi, kvadrati ili pravougaonici, dok se grane najčešće predstavljaju kao otvorene Žordanove krive, koje spajaju simbole koji predstavljaju odgovarajuća temena. Kao što je već rečeno, u zavisti od oblasti primene crtanja grafa, koriste se različite notacije. Na primer, u matematici se grafovi veoma često predstavljaju pravolinijskom crtanjima, jer su najintuitivnija, dok se ortogonalno crtanje koristi u radu sa bazama podataka ili pravljenju električnih šema. Kada se sve ovo uzme u obzir, jedan graf može imati mnogo vizuelno različitih crtanja.

Upotrebljivost crteža nekog grafa se ogleda prvenstveno u njegovoj ,,čitljivosti" za određeni kontekst (šta je ,,čitljivo" za jedan kontekst može biti krajnje nečitljivo za neki drugi kontekst). ,,Čitljivost" u određenom kontekstu se može definisati preko nekih estetskih kriterijuma bitnih za sam kontekst (kao što je

[1] Donald Knuth (1938-), američki matematičar i informatičar.

recimo minimalni broj preseka grana grafa, minimalna površina crteža, raspoređivanje temena u crtežu u odgovarajuće nivoe, simetrija, minimalni broj zakrivljenja na grani, itd).

Estetska svojstva koja se najčešće koriste su:

1. *Minimizacija ukupnog broja preseka između grana*;
2. *Minimizacija površine crteža* — površina crteža može formalno da se definiše na različite načine (na primer, možemo je definisati kao površinu najmanjeg konveksnog poligona koji pokriva crtež ili kao površinu najmanjeg pravougaonika sa vertikalnim i horizontalnim ivicama koji pokriva crtež). Pri tom, rastojanje između bilo koja dva temena ne sme biti veće od unapred definisane vrednosti (na ovaj način će biti izbegnuto da se dobije manja ili veća površina crteža skaliranjem).
3. *Minimizacija ukupne dužine svih ivica*;
4. *Minimizacija maksimalne dužine ivice*;
5. *Minimizacija ukupnog broja zakrivljenja (eng. bend)* — posebno važno za ortogonalno crtanje;
6. *Minimizacija maksimalnog broja zakrivljenja na ivici*;
7. *Maksimizacija najmanjeg mogućeg ugla između dve susedne grane (incidentne istom temenu)*;
8. *Minimizacija proporcije crteža (eng. aspect ratio)* — odnosa veće i manje stranice najmanjeg pravougaonika koji pokriva crtež;
9. *Simetrija* — prikaz simetrije grafa u crtanju.

Primena bilo kog od ovih svojstava je netrivijalan problem. Na primer, Geri[2] i Džonson[3] su 1983. godine pokazali da je minimizacija broja preseka NP-kompletan problem [6]. Kramer[4] i Leuven[5] su dokazali da je NP-kompletan problem testiranje da li je moguće graf utopiti u mrežu unapred određene veličine [7]. Garg[6] i Tamasia[7] su pokazali da je problem određivanja minimalnog broja zakrivljenja kod ortogonalnog crtanja takođe NP-kompletan problem [9].

Pored estetskih kriterijuma koje crtež treba da zadovolji i koja predstavaljaju opšta pravila i kriterijume koje se odnose na ceo graf i celo crtanje, ograničenja se odnose na podgraf, odnosno deo crteža. Najčešće korišćena ograničenja prilikom crtanja grafova su:

- *centralno* (*eng. center*) — pozicioniranje datog temena u središte crteža;
- *spoljašnje* (*eng. external*) — pozicioniranje datog temena na spoljašnju granicu crteža;

[2] M. R. Garey, informatičar.
[3] D. S. Johnson, informatičar.
[4] M. R. Kramer, informatičar
[5] J. van Leeuwen, informatičar.
[6] A. Garg, informatičar.
[7] Roberto Tamassia, informatičar.

- *grupno* (*eng. cluster*) — pozicioniranje datog skupa temena tako da se nalaze blizu jedna drugima;
- *levo-desna (gore-dole) sekvenca* (*eng. left-right (top-bottom) sequence*) — crtanje date putanje horizontalno poravnate sa leva na desno, odnosno od gore na dole;
- *oblik* (*eng. shape*) — crtanje datog podgrafa odgovarajućim oblikom.

Pored estetskih kriterijuma i ograničenja, kao parametara koji se koriste za crtanje grafova, bitan parametar predstavlja i efikasnost crtanja (pogotovo za grafove sa velikim brojem temena).

Većina metodologija za crtanje grafova je bazirana na sledeća dva pravila:

- estetska svojstva su veoma često u konfliktu, pa su prema tome kompromisi neophodni;
- čak i ukoliko nema konflikta među estetskim svojstvima koja treba da budu zadovoljena, najčešće je algoritamski veoma teško zadovoljiti sva u isto vreme.

Iz ovoga sledi da većina metodologija za crtanje grafova definiše prioritet između različitih estetskih kriterijuma. Razni prioriteti su pogodni za neke primene crtanja, dok su za druge primene manje pogodni.

3.2 Lučno-slojevito crtanje

Lučno-slojevito crtanje (*eng. arc-layered drawing*) je crtanje koje spada u klasu hijerarhijskih crtanja. Ova vrsta crtanja se koristi za crtanje opštih grafova. Najčešće se koristi u situacijama kada je potrebno prikazati neki zavisni odnos između entiteta. U takve situacije spadaju PERT dijagrami, grafovi koji ilustruju pozive funkcije, itd. Osnovna ideja u ovom pristupu jeste da se sva temena rasporede u *slojeve* (*eng. layers*), koji su u nekom odnosu. Odnos između slojeva definiše kontekst crtanja.

Kao što je već rečeno, osnovna ideja kod ovog crtanja jeste da se sva temena grafa $G = (V, E)$ raspodele u slojeve $L_i, i = 1, 2, \ldots, s, s \leq n$, za koje važi:

- $1 \leq |L_i| \leq n, i = 1, 2, \ldots, s$
- $\forall (u, v) \in E \Rightarrow (u \in L_i \wedge v \in L_{i+1}) \vee (u \in L_i \wedge v \in L_i)$

Drugim rečima, sva temena su raspoređena u slojeve takve da se temena koja su susedna nalaze ili u istom sloju ili u susednim slojevima. Ovakvo raspoređivanje temena u slojeve je uvek moguće. Grane između susednih temena koja se nalaze u susednim slojevima se crtaju pravim linijama, dok se grane između susednih temena koja se nalaze u istom sloju, crtaju ili pravim linijama (u slučaju da se radi o temenima koja su (geometrijski) susedna) ili lukovima (u ostalim slučajevima).

Tokom crtanja grafa ovom metodom, graf se obilazi po širini, odnosno sledeći sloj se popunjava svim direktnim potomcima svih temena prethodnog sloja.

Činjenica da se susedna temena nalaze ili u istom sloju ili u susednom sloju ovo crtanje čini veoma intuitivnim i strukturu koju graf opisuje veoma čitljivom (veze između temena se veoma lako uočavaju).

Definišimo preciznije termine koji su potrebni u nastavku ovog poglavlja.

Definicija 3.2.1 *Neka je $G = (V, E)$ neusmereni graf.* Podela u nivoe *(eng. layering) je particionisanje skupa V u podskupove $L_1, L_2, \ldots, L_h$, takve da ukoliko je $(u, v) \in E$, gde je $u \in L_i$ i $v \in L_j$, tada $j = i + 1$. Graf G zovemo* slojevitim grafom.

Definicija 3.2.2 *Neka je $G = (V, E)$ neusmereni graf, koji je podeljen u nivoe $L_1, L_2, \ldots, L_h$.* Visina *slojevitog grafa G je broj nivoa h. Još kažemo da je G h-*slojevit graf.

Definicija 3.2.3 Širina *grafa je broj temena u najvećem sloju, to jest:* $\max_{1 \leq i \leq h} |L_i|$.

Najčešće se slojevi crtaju menjanjem y koordinata, odnosno slojevi se nalaze „jedan iznad drugog", međutim, to je stvar konvencije.

U prvom koraku potrebno je rasporediti sva temena u ogovarajuće slojeve. Zatim, prilikom iteriranja kroz slojeve, postavljati odgovarajuće x i y koordinate temena (na jednom sloju sva temena imaju istu y koordiantu, dok su razlika između x koordinata dva susedna (u sloju) temena jednaka. Grane koje spajaju temena iz susednih slojeva se crtaju pravom linijom, dok se grane koje spajaju temena iz istog sloja crtaju pravom linijom, ukoliko se radi o temenima koja su susedna (u tom sloju), odnosno lûkom ukoliko se radi o temenima koja nisu susedna (u tom sloju).

Algoritam 3.2.1 (Algoritam za lučno-slojevito crtanje) *Ovaj algoritam je dat kroz nekoliko manjih algoritama rad lakšeg razumevanja.*

```
procedure ArcLayeredDrawing;
begin
     napraviSlojeve();
     Postavi sirinu najsireg sloja u najsiraSirinaSloja;
     horizontalniPomeraj=0;
     Postavi kooridnate svih temena na 0;
     Postavi trenutnaYKoordinata na 0;
     for each nivoSloja in slojevi
     begin
          hozirontalniPomeraj=nadjiNajsiruLukGranu(nivoSloja);
          postaviXKoordinatu(nivoSloja, najsiraSirinaSloja);
          postaviYKoordinatu(nivoSloja, trenutnaYKoordinata);
          if (hozirontalniPomeraj>1)
               trenutnaYKoordinata=trenutnaYKoordinata+
                  (hozirontalniPomeraj+1);
          else
               trenutnaYKoordinata=2;
     end;
     upisiCrtanjeUFajl();
end;

procedure napraviSlojeve;
begin
     Postavi sva temena kao neistrazena temena;
     Postavi prvi sloj;
```

```
     i=0;
     while postoje neistrazena temena
     begin
          Nadji potomke svih temena (i-1)-og sloja;
          Postavi potomke u i-ti nivo sloja;
          i=i+1;
     end;
end;

procedure nadjiNajsiruLukGranu;
begin
     Postavi najsiraLukGrana na 0;
     for i=0 to velicinaSloja-1
     begin
          for j=i+1 to velicinaSloja
          begin
               if (graf sadrzi granu koja spaja
                  slojevi[nivoSloja][i] i slojevi[nivoSloja][j])
                    if j-i>najsiraLukGrana
                         najsiraLukGrana=j-1;
          end;
     end;
return najsiraLukGrana;
end;

procedure postaviXKoordinatu;
begin
     Postavi pocetnaPozicija to
        najsiraSirinaSloja-size(sloj[nivoSloja])
     for i=0 to velicinaSloja
          Postavi x koordinatu za slojevi[nivoSloja][i] na
             pocetnaPozicija+2*i;
end;


procedure postaviYKoordinatu;
begin
     for i=0 to velicinaSloja
          Postavi y koordinatu za slojevi[nivoSloja][i] na
             procedure argument trenutnaYKoordinata;
end;
```

Napomenimo da prvi nivo ne mora da ima samo jedno teme. U različitim varijacijama metode je moguće postaviti da se u prvom sloju nalazi različit broj temena.

Preostalo je još implementacija samog crtanja, odnosno crtanje grafa na osnovu nivoa i koordinata temena u njima.

Algoritam 3.2.2 *Algoritam za crtanje grafa lučno-slojevitom metodom na osnovu novoa i koordinata temena u njima. Ova procedura se sastoji od definisanja*

temena i njihovog označavanja na crtežu i od crtanja grana. S obzirom da je prvi deo algoritma trivijalan, ovde će biti predstavljeno samo crtanje grana grafa.

```
procedure upisiCrtanjeUFajl;
begin
     definisiTemena;
     napraviVeze;
end;

procedure napraviVeze;
begin
     for i=0 to size(slojevi)-1
     begin
          spojiSusedneSlojeve(i);
          spojiSloj(i);
     end;
     spojiSloj(size(slojevi));
end;

procedure spojiSusedneSlojeve;
begin
     for i=0 to size(slojevi[indeksSloja])
     begin
          for j=0 to size(slojevi[indeksSloja+1])
          begin
               if (graf sadrzi granu koja spaja
                  slojevi[indeksSloja][i] i slojevi[indeksSloja][j])
                       nacrtaj liniju koja spaja slojevi[indeksSloja][i]
                          i slojevi[indeksSloja][j];
          end;
     end;
end;

procedure spojiSloj;
begin
     for i=0 to size(slojevi[indeksSloja])-1
     begin
          for j=i+1 to size(slojevi[indeksSloja])
          begin
               if (graf sadrzi granu koja spaja
                  slojevi[indeksSloja][i] i slojevi[indeksSloja][j])
               begin
                    if i+1=j
                         nacrtaj liniju koja spaja slojevi[indeksSloja][i]
                            i slojevi[indeksSloja][j]
                    else
                         nacrtaj luk koji spaja slojevi[indeksSloja][i]
                            i slojevi[indeksSloja][j]
               end;
          end;
```

```
    end;
end;
```

U primeru 5.3.1 i na slikama 8.1, 8.2, 8.3 i 8.4 se mogu videti primeri crtanja lučno-slojevitom metodom.

3.3 Baricentrična metoda

Baricentrična metoda (*eng. barycenter method*) je metoda koja spada u grupu metoda iz pristupa usmeravanja silom. Algoritmi iz ovog pristupa su metode za generisanje pravolinijskih crtanja neusmerenih grafova. Osnovna ideja kod ovog pristupa jeste simulacija sistema sila koje su definisane na ulaznom grafu. Izlaz je konfiguracija minimalne energije. Postoje dve glavne komponente kod ovog pristupa:

- *model sile* (*eng. force model*). Na primer, možemo dodeliti oprugu „prirodne dužine" $l_{u,v}$ svakom paru (u, v) temena. Dalje, možemo odabrati da $l_{u,v}$ bude broj grana na najkraćoj putanji između u i v. Opruge se ponašaju po *Hook*[8]*-ovom zakonu*[9], to jest, indukuju silu magnitude $d_{u,v} - l_{u,v}$ na u, gde je $d_{u,v}$ proporcionalno euklidskom rastojanju između u i v
- *tehnike za nalaženje lokalnog minimuma konfiguracije energije*. Ovakve tehnike su najčešće proizvod numeričke analize pre nego kombinatornih algoritama.

Kao što je već rečeno, osnovna ideja kod ovih algoritama je da koriste analogiju sa fizikom za crtanje grafova. Graf razmatramo kao sistem tela između kojih postoje sile. Algoritam traži konfiguraciju tela sa lokalno minimalnom energijum, to jest, poziciju svakog tela takvu da je suma svih sila na svako telo jednaka nuli. Takva pozicija se naziva *stanje ravnoteže* (*eng. equilibrium configuration*). Ovakvu konfiguraciju možemo da interpretiramo pravolinijskim crtanjem.

Naravno, model može da bude definisan na neki drugi način, a ne samo kao sistem sila. Na primer, model bi mogao biti definisan kao sistem energija. U tom slučaju, algoritam bi moglao da se posmatra kao tehnika za pronalaženje konfiguracija sa lokalno minimalnom energijom. U ovu svrhu bi mogao da se koristi model opruga (u tom slučaju bi bile korisšćen opruge i električna energija).

Metode usmeravanja silom su veoma popularne, između ostalog i zbog toga što fizička interpretacija predstavlja veoma intiutivnu analogiju sa samim crtanjem, kao i zbog činjenice da su rezultati vizuelno veoma dobri.

Ove metode su korišćene i ranije u druge svrhe osim u vizualizaciji grafova. Između ostalog, korišćene su u matematici, dizajnu štampanih ploča, itd. Jedna od najpoznatijih metoda usmeravanja silom jeste baricentrična metoda koju je razvio Tut[10].

Pre nego što predstavimo samu baricentričnu metodu, predstavićemo jedan model koji se bazira na sistemu opruga i električnih sila. Grane modelujemo kao

[8] Robert Hook (1635 - 1703) — engleski naučnik.

[9] Hook-ov zakon elastičnosti u mehanici i fizici, je aproksimacija koja tvrdi da je količina istezanja materijala u linearnom odnosu sa silom koja isteže taj materijal

[10] William Thomas Tutte (1917 - 2002) — britanski matematičar.

opruge, a temena predstavljaju jednako naelektrisane čestice koje se međusobno odbijaju. Preciznije, sila koja deluje na teme v je:

$$F(v) = \sum_{(u,v)\in E} f_{uv} + \sum_{(u,v)\in V\times V} g_{uv} \tag{3.1}$$

gde f_{uv} predstavlja silu koja deluje usled opruge između u i v, dok g_{uv} predstavlja električno odbijanje koje deluje na v od strane temena u. Sila f_{uv} je sila koja se ponaša po *Hukovom zakonu*, to jeste f_{uv} je sila proporcionalna razlici između rastojanja između u i v i dužine opruge koja ima eneriju jednaku nuli. Električna sila g_{uv} se dobija po inverznom kvadratnom zakonu.

Označimo sa $g(p,q)$ euklidsko rastojanje između tačaka p i q i pretpostavimo da je pozicija temena v označena sa $p_v = (x_v, y_v)$. Prema tome, na osnovu jednačine 3.1, x komponenta sile $F(v)$ je na teme v je:

$$\sum_{(u,v)\in E} k_{uv}^{(1)}(d(p_u,p_v)) - l_{uv})\frac{x_v - x_u}{d(p_u,p_v)} + \sum_{(u,v)\in V\times V} \frac{k_{uv}^{(2)}}{(d(p(u,p_v))^2}\frac{x_v - x_u}{d(p_u,p_v)} \tag{3.2}$$

y komponenta sile $F(v)$ se računa na sličan način. Parametri l_{uv}, k_{uv}^1 i $k_{uv}^{(2)}$ su nezavisni u odnosu na poziciju temena i mogu se interpretirati na sledeći način:

- prirodna dužina (nula energija) opruge između u i v je l_{uv}. Ukoliko je dužina opruge l_{uv} (to jest, $d(p_u,p_v) = l_{uv}$), tada nikakva sila nije oslobođena od strane (u,v);
- krutost opruge između u i v je izražena sa $k_{uv}^{(1)}$. Što je veća vrednost $k_{uv}^{(1)}$, to je veća tendencija da rastojanje između u i v bude blisko l_{uv};
- snaga električne odbojnosti između u i v zavisi od $k_{uv}^{(2)}$.

Ovaj model teži da zadovolji dva estetska kriterijuma:

- Sila opruga između susednih temena treba da osigura da rastojanje između susednih temena u i v bude približno jednako l_{uv};
- električno naelektrisanje treba da osigura da temena ne treba da budu blizu jedno drugome.

Praksa je pokazala da su crtanje dobijena ovim modelom veoma „lepa". Takođe, pod određenim pretpostavkama crtanja su simetrična.

Moguće je da se umesto opruga koje se ponašaju po Hukovom zakonu koriste logaritamske opruge. U tom slučaju x komponenta od f_{uv} u jednačini 3.1 postaje:

$$k_{uv}^{(1)} \log\left(\frac{d(p_u,p_v)}{l_{uv}}\right)\frac{x_v - x_u}{d(p_u,p_v)}$$

Međutim, pokazuje se da je teško opravdati dodatna izračunavanja kvalitetom rezultujućeg crtanja.

U zavistnosti od konteksta crtanja, parametri $l_{uv}, k_{uv}^{(1)}$ i $k_{uv}^{(2)}$ se mogu odabrati tako da se dodatno utiče na crtanje ili da se dodatno naglase neke semantičke

osobine samoga grafa. Na primer, l_{uv} predstavlja poželjno rastojanje između u i v. Ukoliko je odnos izražen stranicom između u i v jak, tada bi l_{uv} trebalo da bude malo. Prema tome, rastojanje između u i v će biti manje za jače odnose temena.

Algoritam traži stanje ravnoteže za ove sile, to jest crtanje u kome je ukupna sila $F(v)$ za svako teme v jednaka nuli. Ekvivalentno, algoritam traži crtanje u kome je energija lokalno minimalna u odnosu na pozicije temena.

Baricentrična metoda je jedna od najpopularnijih metoda usmeravanja silom. Kao što je već rečeno, radi se o metodi koju je razvio Tut i može se opisati kao varijacija metode koja je opisana u dosadašnjem delu poglavlja.

Tutov model koristi opruge za koje važi $l_{uv} = 0$, odnosno da je sila koja deluje na v od strane grane (u, v) proporcionalna $d(p_u, p_v)$. Parametar krutosti $k_{uv}^{(1)}$ je postavljen na jedan za svaku granu (u, v), a električne sile ne postoje. Prema tome, $F(v)$ možemo izraziti na sledeći način:

$$F(v) = \sum_{(u,v)\in E} (p_u - p_v) \tag{3.3}$$

Očigledno je da se stanje ekvilibriuma za skup sila datih jednačinom (3.3) može postići za trivijalno rešenje $p_v = 0$ za svako teme v. Naravno, ovo crtanje nije „lepo" crtanje. Da bi izbegli trivijalno rešenje, skup temena V, možemo podeliti u dva skupa:

- skup od najmanje tri *fiksna temena* (*eng. fixed vertices*). Ovo su temena koja su fiksirana i čije se koordinate ne menjaju posle inicijalnog postavljanja. Prema tome, sile opruga ne utiču na položaj ovih temena. Pozicije temena ovog skupa treba da budu odabrane tako da čine konveksni poligon.
- skup *slobodnih temena* (*eng. free vertices*). Temena koja su ostala u skupu V posle definisanja skupa fiksnih temena. Ova temena su temena čija se pozicija ažurira tokom rada algoritma. Odnosno, to su temena na koja utiču sile opruga.

Pozicije za slobodna temena biramo tako da važi jednakost (3.3). To jest, biramo p_v, tako da važi $F(v) = 0$ za svako slobodno teme v. Prema tome, važi:

$$\sum_{(u,v)\in E} (x_u - x_v) = 0 \tag{3.4}$$

odnosno:

$$\sum_{(u,v)\in E} (y_u - y_v) = 0 \tag{3.5}$$

gde je $p_v = (x_v, y_v)$ za svako teme v. Označimo sa $N_0(v)$ skup fiksiranih temena, a sa $N_1(v)$ skup slobodnih temena. Tada jednačine (3.4) i (3.5), možemo zapisati na sledeći način:

$$\deg(v)x_v - \sum_{u\in N_1(v)} x_u = \sum_{w\in N_0(v)} x_w^* \tag{3.6}$$

i:

$$\deg(v)y_v - \sum_{u \in N_1(v)} y_u = \sum_{w \in N_0(v)} y_w^* \tag{3.7}$$

gde je pozicija fiksnog temena w (x_w^*, y_w^*), a stepen temena v je označen sa $\deg(v)$.

Primetimo da su jednačine (3.6) i (3.7) linearne jednačine i da je broj jednačina i broj nepoznatih jednak broju slobodnih temena. Rešavanje ovih jednačina se svodi na postavljanje svakog slobodnog temena u baricentar njegovih suseda, te se iz ovog razloga ova metoda i zove baricentrična metoda.

Finalne jednačine po kojima se računaju koordinate su:

$$x_v = \frac{1}{deg(v)} \sum_{(u,v) \in E} x_u \tag{3.8}$$

i:

$$y_v = \frac{1}{deg(v)} \sum_{(u,v) \in E} y_u \tag{3.9}$$

Koordinate se računaju iterativno koristeći navedene formule sve dok x_v i y_v ne konvergiraju za svako slobodno teme v. U knjizi [2], u kojoj se može naći opis rada baricentrične metode ne dokazuje se konvergencija ove metode. Ipak, u praksi ova metoda konvergira i verovatno se može dokazati konvergencija u opštem slučaju.

Algoritam 3.3.1 *Algoritam za crtanje opšteg grafa baricentričnom metodom.*

```
procedure BarycenterDrawing
begin
     Postavi svako fiksirano teme na njegove koordinate;
     Postavi svako fiksirano teme u koordinatni pocetak;
     Postavi da za sva slobodna temena treba raditi X i Y azuriranje;
     trenutnaIteracija=0;
     do
     begin
          for each vertex v in V do
          begin
               stepenTemena=nadjiStepenTemena(v);
               Nadji susede temena v;
               if v zahteva X azuriranje
               begin
                    novo_x_v=1/stepenTemena*suma_{(u,v)\in E}(x_u);
                    if |novo_x_v-x_v|<tacnost
                         Postavi da za teme v vise ne treba raditi
                            X azuriranje;
                    x_v=novo_x_v;
               end;
               if v zahteva Y azuriranje
               begin
                    novo_y_v=1/stepenTemena*suma_{(u,v)\in E} (y_u);
                    if |novo_y_v-y_v|<tacnost
```

```
                        Postavi da za teme v vise ne treba raditi
                           Y azuriranje;
                    y_v=novo_y_v;
               end;
          end;
          trenutnaIteracija=trenutnaIteracija+1;
     end;
     while(nastaviSaRadom());
end;

procedure nastaviSaRadom;
begin
     if trenutnaIteracija>=maksimalni broj iteracija
          return false;
     for each vertex v in V do
     begin
          if v zahteva X azuriranje
               return true;
          if v zahteva Y azuriranje
               return true;
     end;
     return false;
end;
```

Ono što je posebno interesantno kod baricentrične metode je činjenica da se za različita fiksirana temena, dobijaju različiti crteži. To znači da se odabir fiksiranih temena, može razmatrati kao postavljanje parametara koji utiču na samo crtanje grafa, što omogućava da se ovom metodom, u zavisnosti od odabira fiksiranih temena, dobiju različite koordinate temena, odnosno stanja ekvilibrijuma se postižu za različite vrednosti. Ova činjenica baricentričnu metodu čini posebno interesantnom.

U primeru 5.3.2, i na slikama 8.5 i 8.6 se mogu videti primeri crtanja baricentričnom metodom.

Glava 4

Pravolinijsko crtanje planarnih grafova

Kao što je već rečeno, pravolinijsko crtanje je crtanje koje omogućava reprezentaciju grafa na ,,najintuitivniji" način. Zbog činjenice da je svaka grana predstavljena pravom linijom, veoma je lako uočiti odnose među temenima (praćenje grane od temena do temena je u ovom crtanju najjednostavnije). Iz ovog razloga, ova vrsta crtanja je veoma popularna. Međutim, tek od nedavno postoje algoritmi koji daju pravolinijsko crtanje proizvoljnog planarnog grafa. Mada je odavno dokazano da svaki planarni graf ima pravolinijsko crtanje, algoritmi za pravolinijsko crtanje su se pojavili mnogo kasnije. Dugo su se matematičari bavili pitanjem postojanja pravolinijskog crtanja, a ne toliko samim crtanjem. Iz tog razloga su svi algoritmi razvijeni pre 1988. godine imali veoma lošu rezoluciju, odnosno, temena su bila raspoređena na takav način da je crtanje bilo veoma ,,nečitko", što je ovo crtanje često činilo praktično neupotrebljivim. Detaljnije se o crtanju planarnih grafova može naći u [1].

4.1 Stilovi crtanja planarnih grafova

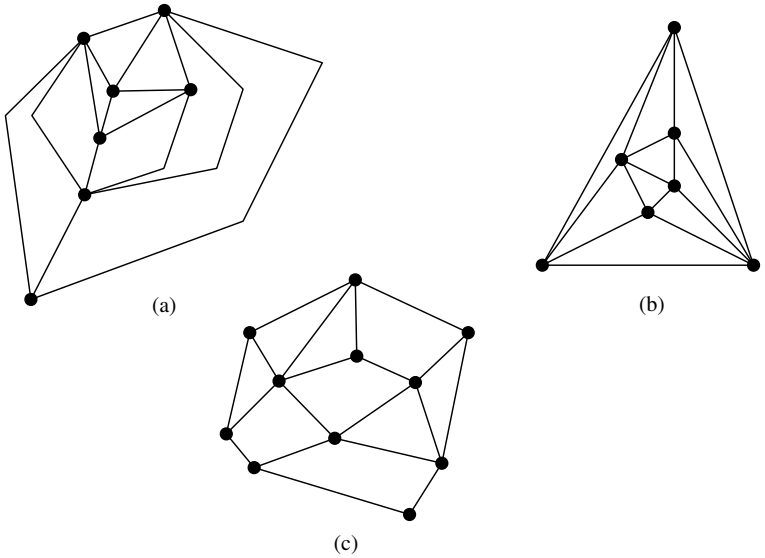
Posebna grupa grafova su planarni grafovi (Definicija 2.3.1 na strani 16). Jedan planaran graf može imati veliki broj utapanja. Crtanje planarnih grafova je znatno teži problem od crtanja opštih grafova. U okviru ove klase grafova postoje različite vrste crtanja:

1. *poligono crtanje (eng. polyline drawing)* — svaka grana je predstavljena poligonom linijom (slika 4.1(a));

2. *pravolinijsko crtanje (eng. straight line drawing)* — svaka grana je predstavljena delom prave linije (slika 4.1(b));

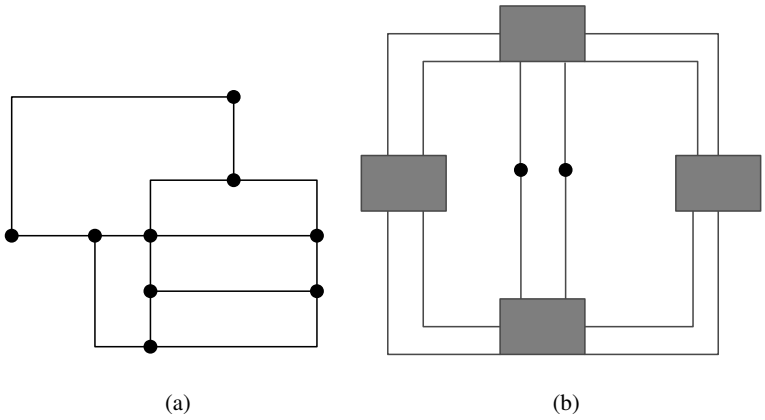
3. *konveksno crtanje (eng. convex drawing)* — pravolinijsko crtanje planarnog grafa kod koga su sve strane (*eng. faces*) grafa konveksni poligoni (slika 4.1(c));

4. *ortogonalno crtanje (eng. orthogonal drawing)* — svako zakrivljenje grane je pod pravim uglom (slika 4.2(a));



Slika 4.1: (a) poligono crtanje; (b) pravolinijsko crtanje; (c) konveksno crtanje

5. *box-ortogonalno crtanje (eng. box-orthogonal drawing)* — teme je standardno prikazano kao tačka. Očigledno, graf koji ima teme stepena većeg ili jednokog od pet nema ortogonalno crtanje, jer najviše četiri grane mogu biti incidentne temenu u ortogonalnom crtanju. Kod box-ortogonalnog crtanja teme prikazano kao (potencijalno degenerisani) pravougaonik, koji se naziva box, a svaka ivica je prikazana kao naizmeničan niz horizontalnih i vertikalnih duži (slika 4.2(b)). Svaki ravanski graf ima box-ortogonalno crtanje.

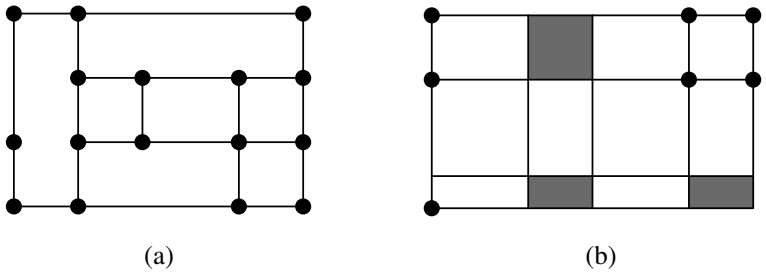


Slika 4.2: (a) ortogonalno crtanje; (b) box-ortogonalno crtanje

6. *pravougaono crtanje (eng. rectangular drawing)* — svako teme je prikazano kao tačka, a svaka grana je prikazana ili kao horizontalna ili vertikalna lin-

ija (bez preseka sa drugim linijama). Pri tome, svaka strana je pravouganik (slika 4.3(a));

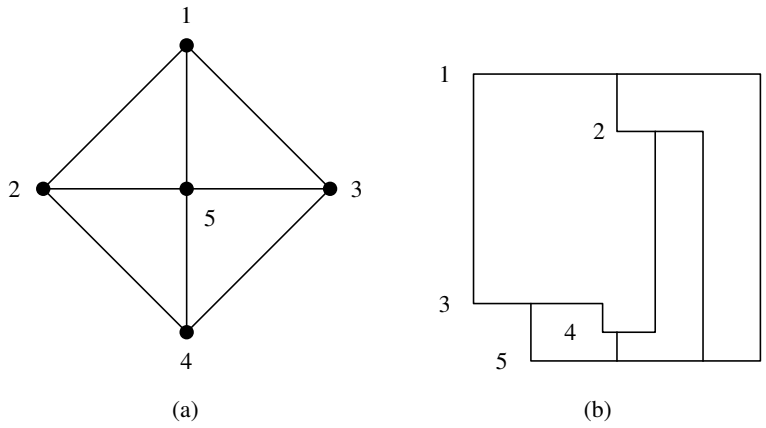
7. *box-pravougaono crtanje (eng. box-rectangular drawing)* — svako teme je nacrtano kao (potencijalno degenerisani) pravougaonik, a pri tome je kontura svake strane pravougaonik (slika 4.3(b)). Ukoliko graf G ima višestruke grane ili teme stepena većeg ili jednakog od pet, tada G nema pravougaono crtanje, ali možda ima box-pravougaono crtanje. Međutim, nema svaki ravanski graf box-pravougaono crtanje.



Slika 4.3: (a) pravougaono crtanje; (b) box-pravougaono crtanje

8. *mrežno crtanje (eng. grid drawing)* — temena i zakrivljenja su na tačkama preseka celobrojne mreže;

9. *vidljivo crtanje (eng. visibility drawing)* — svako teme je prikazano kao horizontalna duž, dok je svaka grana nacrtana kao vertikalna duž. Vertikalna duž, koja predstavlja ivicu mora da povezuje tačke na horizontalnim dužima, koje predstavljaju temena koja treba spojiti. Postoji generalizacija vidljivog crtanja koja se naziva *2-vidljivo crtanje (eng. 2-visibility drawing)*, u kome su temena prikazana kao pravougaonici, a ivice su prikazane kao horizontalne ili vertikalne duži.



Slika 4.4: (a) graf G sa 5 temena; (b) vidljivo crtanje grafa G

4.2 Osnovne ideje za pravolinijsko crtanje planarnih grafova

Definicija 4.2.1 Pravolinijsko crtanje *planarnog grafa je crtanje u kome je svaka grana nacrtana kao duž i pri tome se grane ne seku.*

Vagner[1] [10], Fari[2] [11] i Štajn[3] [12] su nezavisno dokazali da svaki planarni graf ima pravolinijsko crtanje.

Teorema 4.2.1 (Fari) *Svaki prost (Definicija 2.1.1 na strani 11) planarni graf se može nacrtati tako da mu se grane ne seku i da su predstavljene kao duži.*

Iz dokaza ove teoreme neposredno sledi algoritam koji u polinomijalnom vremenu (po broju temena) može da nađe pravolinijsko crtanje datog planarnog grafa. Međutim, površina pravougaonika koji pokriva crtanje na celobrojnoj mreži, nije polinomijalno ograničena. Zapravo, mnogo godina je bilo otvoreno pitanje naći površinu pravougaonika koji pokriva crtanje, a koja je polinomijalno ograničena. Godine 1990., De Frejsi [4] [13] i Šnider[5] [14] su dvema nezavisnim metodama dokazali da svaki planarni graf sa $n \geq 3$ temena ima pravolinijsko crtanje na celobrojnoj mreži veličine $(2n-4)\times(n-2)$, odnosno $(n-2)\times(n-2)$. Ova dva metoda su poznata kao metoda pomeraja (*eng. shift method*) i metoda realizer (*eng. realizer method*).

U ovoj glavi će biti predstavljen konstruktivni dokaz teoreme de Frejsia da svaki ravanski graf G sa $n \geq 3$ temena ima pravolinijsko crtanje na mreži veličine $(2n-4)\times(n-2)$. Ovaj algoritam kao ulaz uzima triangulirani graf. Ukoliko graf G nije triangulirani, triangulirani graf G' se od njega može dodati dodavanjem lažnih temena (*eng. dummy vertices*). Zatim se nalazi pravolinijsko crtanje grafa G', a neposredno iz pravolinijsko crtanja grafa G' sledi pravolinijsko crtanje grafa G, koje se dobija brisanjem lažnih temena. Prema tome, dovoljno je dokazati da triangulirani ravanski graf ima pravolinijsko crtanje na mreži veličine $(2n-4)\times(n-2)$.

4.3 Kanonsko uređenje

U cilju pravolinijskog crtanja, de Frejsi je uveo uređenje, poredak, koje se naziva *kanonsko uređenje* (*eng. canonical ordering*). Ugrađivanje temena u crtanje se radi teme po teme, prema kanonskom uređenju.

Definicija 4.3.1 *Neka je C ciklus u grafu G. Ivica koja spaja dva neuzastopna temena u C se naziva* tetiva *(eng. chord).*

Definicija 4.3.2 *U 2-povezanom ravanskom grafu G, sa $C_0(G)$ označavamo spoljašnji ciklus od G, odnosno ivicu spoljašnje strane od G. Teme $v \in C_0(G)$ nazivamo* spoljašnjim temenom *(eng. outer vertex), a ivicu $e \in C_0(G)$ nazivamo* spoljašnjom ivicom.

[1] K. Wagner, informatičar.
[2] István Fáry (1922 - 1984) — mađarski matematičar.
[3] K. S. Stein, informatičar.
[4] H. de Fraysseix, informatičar.
[5] W. Schnyder, informatičar.

Definicija 4.3.3 *Za graf kažemo da je* unutrašnje trianguliran *(eng. internally triangulated), ukoliko je svaka unutrašnja strana trougao.*

Definicija 4.3.4 *Neka je $G = (V, E)$ trianguliran ravanski graf sa $n \geq 3$ temena. Kako je G trianguliran, na $C_0(G)$ se nalaze tačno tri temena. Označimo ta temena sa v_1, v_2 i v_n i pretpostavimo da se na $C_0(G)$ nalaze u smeru suprotnom od kazaljke na satu. Neka je dalje, $\pi = (v_1, v_2, \ldots, v_n)$ uređenje svih temena grafa G. Za svaki prirodan broj $3 \leq k \leq n$, sa G_k označavamo ravanski podgraf od G indukovan temenima $(v_1, v_2, \ldots, v_k)$. Važi $G_n = G$. π nazivamo* kanonskim uređenjem *(eng. canonical ordering), ako sledeći uslovi važe za svako k, $3 \leq k \leq n$:*

(a) G_k je 2-povezan i unutrašnje triangulirani;

(b) (v_1, v_2) je spoljašnja ivica G_k;

(c) ako je $k + 1 \leq n$, tada se teme v_{k+1} nalazi na spoljašnjoj strani G_k i uz to svi susedi temena v_{k+1} se na $C_0(G_k)$ pojavljuju uzastupno.

Primer 4.3.1 *Kanonsko uređenje grafa G iz primera 2.4.1 je:*

$$1, 2, 3, 4, 5, 6, 7, 8$$

Teorema 4.3.1 *Svaki triangulirani ravanski graf G ima kanonsko uređenje.*

Dokaz: Graf G očigledno ima kanonsko uređenje za $n = 3$. Pretpostavimo da je $n \geq 4$. S obzirom da je $G = G_n$, uslovi (a), (b) i (c) važe za $k = n$. Odaberimo $n - 3$ unutrašnja temena $v_{n-1}, v_{n-2}, \ldots, v_3$ u ovom redosledu i pokažimo da uslovi (a), (b) i (c) važe za $k = n - 1, n - 2, \ldots, 3$.

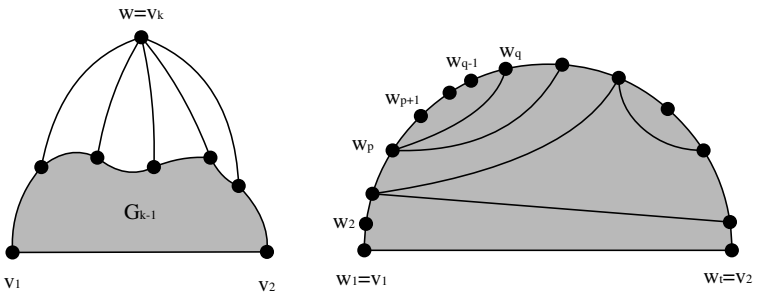
Pretpostavimo (induktivna pretpostavka) da su temena $v_n, v_{n-1}, \ldots, v_{k+1}$, $k + 1 \geq 4$ valjano odabrana i da uslovi (a), (b) i (c) važe za k. Ukoliko za v_k možemo izabrati teme $w \neq v_1, v_2$ na ciklusu $C_0(G_k)$ koje nije kraj tetive na $C_0(G_k)$, kao što je ilustrovano na slici 4.5(a), tada uslovi (a), (b) i (c) važe za $k - 1$, jer je $G_{k-1} = G_k - v_k$. Prema tome, dovoljno je da pokažemo da takvo teme w postoji.

Neka je $C_0(G_k) = w_1, w_2, \ldots, w_t$ i neka je pri tome $w_1 = v_1$ i $w_t = v_2$. Ukoliko $C_0(G_k)$ nema tetiva, tada se za w može odabrati bilo koje od temena $w_2, w_3, \ldots, w_{t-1}$. Pretpostavimo da $C_0(G_k)$ ima tetive. Tada G_k ima minimalnu tetivu (w_p, w_q), $p + 2 \leq q$, takvu da nijedno od temena $w_{p+1}, w_{p+2}, \ldots, w_{q-1}$ nije kraj tetive, kao što je ilustrovano na slici 4.5(b), gde su tetive nacrtane neisprekidanim linijama. Tada se bilo koje od temena $w_{p+1}, w_{p+2}, \ldots, w_{q-1}$ može odabrati za teme w. □

Sada možemo predstaviti algoritam koji računa kanonsko uređenje za dati triangulirani ravanski graf G. U opisu algoritma se koriste primitive:

- `mark(v)=true`, ako je `v` dodat u kanonsko uređenje;
- `out(v)=true`, ako je `v` spoljašnje teme ravansko grafa G_k u k-toj iteraciji;
- `chords(v)=s`, gde je `s` broj tetiva spoljašnjeg ciklusa čije je krajnje teme `v`.

Algoritam 4.3.1 *Algoritam za određivanje kanonskog uređenja*



Slika 4.5: (a) G_k; (b) tetive

```
procedure CanonicalOrdering(G);
begin
     Neka su v_1, v_2, v_n temena na spoljasnjem ciklusu
       u smeru kazaljke na satu;
     for each x in V do
     begin
          chords(x)=0;
          out(x)=false;
          mark(x)=false;
     end;
out(v_1)=true;
out(v_2)=true;
out(v_n)=true;
for k=n down to 3 do
begin
     Odaberi proizvoljno x za koje vazi mark(x)=false,
        out(x)=true, chords(x)=0 i x!=v_1, v_2;
     v_k=x;
     mark(x)=true;
     Neka je C0(G_{k-1})=w_1, w_2,..., w_t, gde je w_1=v1
        i w_t=v_2;
     Neka su w_p, w_{p+1},..., w_q susedi temena v_k
        za koje vazi mark(w_i)=false;
     for each w_i, p<i<q
     begin
          out(w_i)=true;
          azuriraj_tetive_i_susede(w_i);
     end;
end;
```

Funkcija `azuriraj_tetive_i_susede(w_i)` će biti detaljno predstavljena u dokazu teoreme 4.3.2.

Teorema 4.3.2 *Algoritam* `CanonicalOrdering(G)` *računa kanonsko uređenje trianguliranog ravanskog grafa G u linearnom vremenu $O(n)$.*

Dokaz: Na osnovu teoreme 4.3.1, uvek postoji teme koje zadovoljava uslov:

```
choose any x such that mark(x)=false, out(x)=true,
   chords(x)=0 and x!=v_1, v_2;
```

te iz ovoga sledi da je algoritam 4.3.1 korektan.

Da bismo našli teme x iz gornjeg uslova u vremenu $O(1)$, ažuriramo listu koja sadrži sva temena koja zadovoljavaju dati uslov.

Funkciju `azuriraj_tetive_i_susede(w_i)` implementiramo na sledeći način:

Ukoliko je $q = p + 1$, tada `chords(w_p)` i `chords(w_q)` smanjujemo za jedan, jer je grana (w_p, w_q) na trenutno spoljašnjem ciklusu $C_0(G_{k-1})$ bila tetiva (w_p, w_q) na prethodnom spoljašnjem ciklusu $C_0(G_k)$.

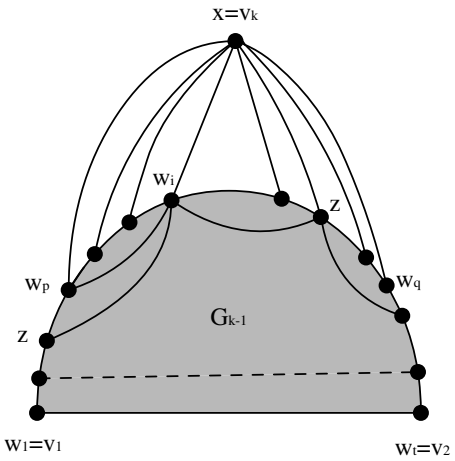
Ukoliko je $q > p + 1$, tada ažuriramo promenljivu `chords` za svako teme w_i, $p < i < q$ i za svaki sused z temena w_i. Za svako teme w_i, $p < i < q$, proveravamo njegove susede z. Ukoliko je `out(z)=true` i $z \neq w_{i-1}, w_{i+1}$, to znači da je (w_i, z) tetiva od $C_0(G_{k-1})$, pa prema tome `chords(w_i)` povećavamo za jedan. Ukoliko je `out(z)=true`, $z \neq w_{i-1}, w_{i+1}$ i $z \neq w_{p+1}, w_{p+2}, \ldots, w_{q-1}$, tada vrednost `chords(z)` povećavamo za jedan.

Ažuriranje temena w_i i njegovih suseda z se može uraditi u vremenu $O(d(w_i))$. S obzirom da se ovo radi samo jednom za svako teme w_i u grafu G, ukupno vreme izvršavanja `azuriraj_tetive_i_susede(w_i)` je je $O(n)$.

Primetimo da, na osnovu teoreme 2.3.3, važi:

$$\sum_{w_i \in V} d(w_i) = 2m \leq 2 \cdot 3n = O(n)$$

Očigledno, sve ostale komande algoritma mogu biti izvršene u vremenu $O(n)$, pa je algoritam 4.3.1 složenosti $O(n)$. □



Slika 4.6: Nove tetive na $C_0(G_{k-1})$ su crtane punom linijom, dok su tetive na $C_0(G_k)$ crtane isprekidanom linijom

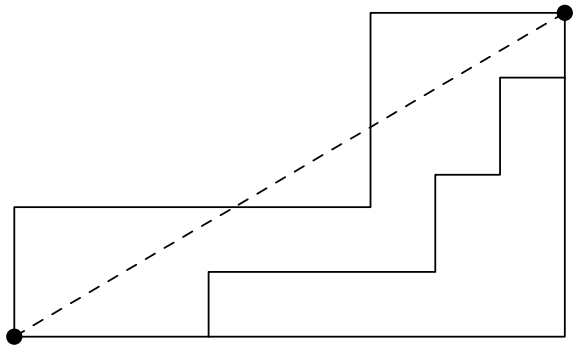
4.4 Metoda pomeraja

U ovom poglavlju će biti predstavljen algoritam koji računa samo crtanje na osnovu prethodno izračunatog kanonskog uređenja. Ovaj algoritam je konstruisao d Frajsi [13]. Algoritam radi tako što u crtež ugrađuje teme po teme, prema kanonskom uređenju $\pi = (v_1, v_2, \ldots, v_n)$. Takođe, u svakoj iteraciju potrebno je ažurirati parcijalno utapanje. Odnosno, u svakoj iteraciji i je potrebno ažurirati skup temena koja treba da budu pomerena svaki put kada se pozicija temena v_i podesi. Ovaj skup temena označavamo sa $L(v_i)$. Primetimo da je $v_i \in L(v_i)$. Napomenimo još da je rezultat primene ovog algoritma crtanje grafa na celobrojnoj mreži.

Definicija 4.4.1 Taksi metrika *(eng. taxicab metrix), ili Menhetn rastojanje (eng. Manhattan distance), je metrika euklidske ravni definisana sa*

$$g((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

za sve tačke $P_1(x_1, y_1)$ i $P_2(x_2, y_2)$. Ovo rastojanje je jednako dužini svih puteva koje spajaju P_1 i P_2 duž horizontalnih i vertikalnih segmenata, bez vraćanja unatrag.



Slika 4.7: Menhetn rastojanje je nactano punom linijom, dok je euklidsko rastojanje nacrtano isprekidanom linijom

Taksi metrika je metrika koju je definisao Herman Minkovski[6]. Menhetn rastojanje je ime koje aludira na ostrvo Menhetn u Njujorku (SAD), gde je većina ulica pod pravim uglom. Ova metrika je još poznata pod imenom rektilinearno rastojanje (*eng. rectilinear distance*), L_1 rastojanje (*eng. L_1 distance*), (*eng. city block distance*) ili Menhetn širina (*eng. Manhattan length*).

U nastavku će biti dat opis algoritma metode pomeraja.

Označimo trenutnu poziciju temena v sa $P(v) = (x(v), y(v))$. Ako su $P_1 = (x_1, y_1)$ i $P_2 = (x_2, y_2)$ dve tačke čije je Menhetn rastojanje parno, tada se prava sa koeficijentom +1 koja sadrži tačku P_1 i prava sa koeficijentom -1 koja sadrži tačku P_2 seku u tački na celobrojnoj mreži. Tu tačku preseka označavamo sa $\mu(P_1, P_2)$. Važi:

$$\mu(P_1, P_2) = \left(\frac{1}{2}(x_1 - y_1 + x_2 + y_2), \frac{1}{2}(-x_1 + y_1 + x_2 + y_2)\right)$$

[6]Hermann Minkowski (1864 - 1909) — nemački matematičar.

Algoritam metode pomeraja se realizuje na sledeći način.

Inicijalno crtamo podgraf G_3. Postavljamo $P(v_1) = (0,0)$, $P(v_2) = (2,0)$ i $P(v_3) = (1,1)$. Takođe, postavljamo $L(v_i) = \{v_i\}$, za $i = 1,2,3$.

Pretpostavimo da je $k-1 \geq 3$ i da je graf G_{k-1} već nacrtan tako da važi:

(a) $P(v_1) = (0,0)$ i $P(v_2) = (2k-6, 0)$;

(b) $x(w_1) \leq x(w_2) \leq \ldots \leq x(w_t)$, gde je $C_0(G) = w_1, w_2, \ldots, w_t$ i pri tome $w_1 = v_1$ i $w_t = v_2$;

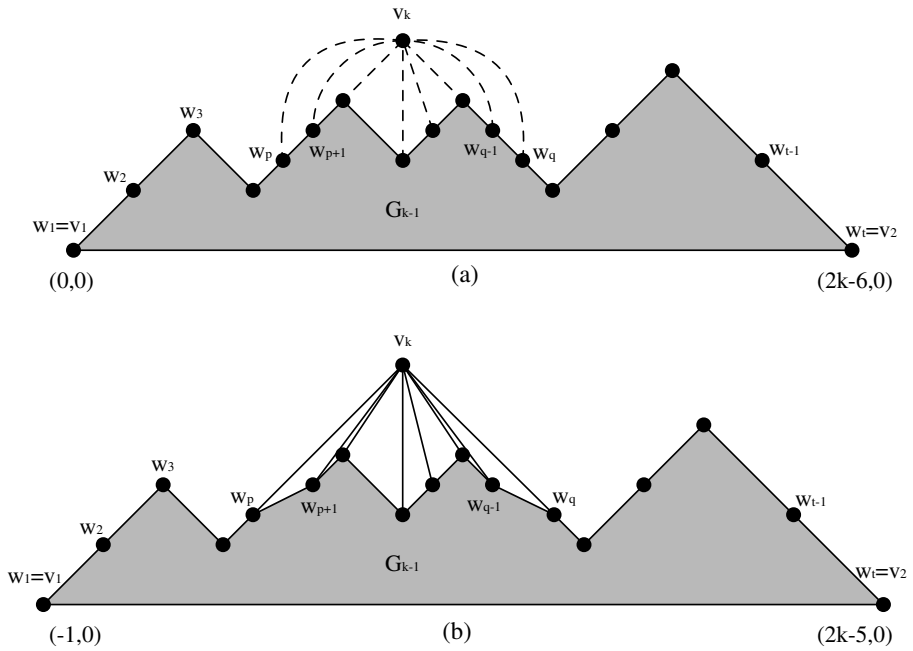
(c) svaka grana (w_i, w_{i+1}) na $C_0(G_{k-1})$ se crta kao prava linija koja ima koeficijent ili -1 ili 1.

Utapanje v_k u već postojeći crtež grafa G_{k-1} se izvršava na sledeći način. Neka su $w_p, w_{p+1}, \ldots, w_q$ susedi temena v_k na $C_0(G_{k-1})$. Kažemo da v_k *pokriva* $w_p, w_{p+1}, \ldots, w_q$. Na osnovu uslova (c), Menhetn rastojanje između w_p i w_q je parno, pa sledi da je $\mu(w_p, w_q)$ na mreži. Međutim, v_k ne možemo nacrtati na $\mu(w_p, w_q)$, jer može da se desi da se linija $w_p v_k$ poklapa sa $w_p w_{p+1}$, zato što $w_p w_{p+1}$ može da ima koeficijent +1, što naravno mora da se izbegne. Iz tog razloga, temena $w_1(= v_1), w_2, \ldots, w_p$ zajedno sa još nekim unutrašnjim temenima pomeramo (*eng. shift*) u levo, dok temena $w_q, w_{q+q}, \ldots, w_t(= v_2)$ zajedno sa još nekim unutrašnjim temenima pomeramo u desno. Tek tada možemo nacrtati teme v_k u tački $\mu(w_p, w_q)$, ali tek kada smo uradili ovaj pomeraj. Ovaj postupak možemo formalno zapisati na sledeći način:

1. $\forall v \in \bigcup_{i=1}^{p} L(w_i)$ do $x(v) = x(v) - 1$
2. $\forall v \in \bigcup_{i=q}^{t} L(w_i)$ do $x(v) = x(v) + 1$
3. $P(v_k) = \mu(w_p, w_q)$
4. $L(v_k) = \{v_k\} \cup \left(\bigcup_{i=p+1}^{q-1} L(w_i) \right)$

S obzirom da se w_p pomera za 1 ulevo, a w_q za 1 udesno, Menhetn rastojanje između ova dva temena će ostati parno, odnosno $\mu(w_p, w_q)$ će se nalaziti na mreži. Takođe, s obzirom da temena $w_1, \ldots, w_p$ pomeramo ulevo, temena $w_q, \ldots, w_t$ pomeramo udesno, a temena $w_{p+1}, \ldots, w_{q-1}$ ostaju nepromenjena, to znači da grane $(w_{p+1}, w_{p+2}), (w_{p+2}, w_{p+3}), \ldots, (w_{q-2}, w_{q-1})$ imaju koeficijent čija je apsolutna vrednost 1, dok ivice (w_p, w_{p+1}) i (w_{q-1}, w_q) imaju koeficijent čija je apsolutna vrednost manja od 1. Iz svega ovoga sledi da su sva temena koja pokriva teme v_k vidljiva iz tačke $\mu(w_p, w_q)$, odnosno ivice $(v_k, w_p), (v_k, w_{p+1}), \ldots, (v_k, w_q)$ se mogu nacrtati tako da se međusobno ne seku.

U opisanom algoritmu postoji jedan problem. Naime, kao uslov (c) za crtanje G_k je navedeno da važi $P(v_1) = (0,0)$ i $P(v_2) = (2k-6, 0)$. Očigledno da ovaj uslov neće važiti posle pomeraja koji budu izvršeni (već posle prve iteracije će biti $P(v_1) = (-1, 0)$. Ovaj problem može se lako rešiti modifikacijom koraka 1 i 2. Naime, umesto tih koraka, definišemo korake 1' i 2':



Slika 4.8: Primer rada metode pomeraja

1' $\forall v \in \bigcup_{i=p+1}^{q-1} L(w_i)$ do $x(v) = x(v) + 1$

2' $\forall v \in \bigcup_{i=q}^{t} L(w_i)$ do $x(v) = x(v) + 2$

Na ovaj način uslovi (a), (b) i (c) važe za G_k za svako k.

Algoritam 4.4.1 (Metoda pomeraja) *Algoritam za pravolinijsko crtanje planarnih grafova*

```
procedure shift;
begin
     Postavi P(v1)=(0,0), P(v2)=(2,0), P(v3)=(1,1) i
        L(vi)={vi} za i=1,2,3;
     Postavi kanonsko uredjenje u costruct;
     for k=3 to k<n do
     begin
          {Korak 1'}
          obradiwpq();
          {Korak 2'}
          obradiwqt();
          {Korak 3}
          izracunajNovoTeme();
```

```
          {Korak 4}
          izracunajNovoL();
     end;
end;

procedure obradiwpq;
begin
     if (size(wpq)<=2) {ako je p+1==q}
          return;
     else              {ako je p+1<q}
     begin
          for i=1 to size(wpq) do
          begin
               for j=0 to size(L[wpq[i]]) do
               begin
                    if L[wpq[i][j]] nije u UnijaLi
                         Dodaj L[wpq[i][j]] u UnijaLi;
               end;
          end;
     end;
     for i=0 to size(UnijaLi) do
          UnijaLi[i].x=UnijaLi[i].x+1;
end;

procedure obradiwqt;
begin
     Postavi qJeDostignuto na false;
     for i=0 to size(w1t) do
     begin
          if w1t[i]==q
               qJeDostignuto=true;
          if qJeDostignuto
          begin
               for j=0 to size(L[w1t[i]]) do
               begin
                    if L[w1t[i][j] nije u UnijaLi
                         Dodaj L(w1t[i][j]) u UnijaLi;
               end;
          end;
     end;
     for i=0 to size(UnijaLi) do
          UnijaLi[i].x=UnijaLi[i].x+2;
end;

procedure izracunajNovoTeme;
begin
     x1=wp.x;
     y1=wp.y;
     x2=wq.x;
     y2=wq.y;
```

```
     vk.x=(x1-y1+x2+y2)/2;
     vk.y=(-x1+y1+x2+y2)/2;
end;

procedure izracunajNovoL;
begin
     if size(wpq)<=2  {ako je p+1==q}
     begin
          Dodaj vk u L[vk];
          return;
     end;
     else             {ako je p+1<q}
     begin
          Dodaj vk u novoL;
          {za svaki element L(wi), i=p+1, ..., q-1}
          for i=1 to size(wpq)-1 do
          begin
               {dodaj elemente iz L(wi) u novo novoL}
               for j=0 to size(L[wpq[i]]) do
               begin
                    if L[wpq[i]][j] nije u novoL
                         Dodaj L[wpq[i]][j] u novoL;
               end;
          end;
     L[vk]=novoL;
     end;
end;
```

Posle primene ovog algorima dobijamo pravolinjsko crtanje grafa $G = G_n$, tako da važi $P(v_1) = (0, 0)$ i $P(v_2) = (2n - 4, 0)$. Na osnovu uslova (c), sledi da je $P(v_n) = (n - 2, n - 2)$. Prema tome, graf G je nacrtan na mreži veličine $(2n - 4) \times (n - 2)$. Ovaj algoritam se može implementirati u vremenu $O(n^2)$.

Primeri pravolinijskog crtanja planarnih grafova metodom pomeraja se mogu videti na slikama 8.7 i 8.8.

Glava 5

Implementacija algoritama za crtanje grafova

U ovoj glavi će biti predstavljeni algoritmi za crtanje grafova sa implementacione strane. Biće predstavljene razne tehnike i tehnologije koje su korišćene. Takođe, u nastavku će biti prikazana i objektna dekompozicija problema.

5.1 Korišćeni alati i tehnologije

Za implementacija algoritama za crtanje grafova predstavljenih u ovom radu korišćen je programski jezik `C++`. Imajući u vidu vrstu problema koja je rešavana, objektno-orijentisani pristup je praktično bio idealan. Tokom razvijanja programa, posebna pažnja je obraćana na mogućnost kasnijeg portovanja ovog programa na različiti operativne sisteme. Mada je program originalno razvijan pod `Linux`-om, korišćenjem `gcc`-a[1], on se može kompajlirati pod `Windows` operativnim sistemom većinom `C++` kompajlera za ovaj operativni sistem[2].

Pored portabilnosti, vođeno je računa o tome da se koristi što manji broj spoljašnjih biblioteka. Iz tog razloga jedina spoljašnja biblioteka koja je korišćena je bio `STL`[3] iz koje su korišćene liste, vektori i mape.

Za automatsko generisanje dokumentacije je korišćen `Doxygen`. Na kraju ovog rada, kao dodatak je priložen i dokument koji predstavlja dokumentaciju za upotrebu ove biblioteke, generisanu sistemom `Doxygen`.

5.2 Objektna dekompozicija problema

U cilju predstavljanja objektnog modela koji je kreiran u ovoj biblioteci, na samom početku će biti predstavljen način reprezentovanja grafa. Tačnije, pre samo reprezentacije grafa će biti predstavljen još osnovniji pojam — teme grafa. Teme grafa je reprezentovano klasom `GraphNode` i karakterišu ga broj temena (*eng. node number*) i naziv (opis) temena (*eng. node label*). Osnovni

[1] Korišćena je verzija 4.1.2 `gcc`-a.

[2] `Visual Studio 2008` i ranije verzije je korišćen za kompajliranje ovog programa pod `Windows`-om.

[3] Standard Template Library

razlog zbog kojeg je teme grafa predstavljeno sa ova dva podatka, a ne samo sa brojem temena, leži u činjenici da svakako jedno od mesta za dalje razvijanje ove biblioteke predstavlja i mogućnost definisanja temena različitih oblika sa različitim opisima istih (više o ovome u glavi 7). U ovoj klasi se nalaze neke osnovne metode za postavljanje i dobijanje informacija o broju temena grafa i nazivu temena grafa, kao i funkcija `updateNodeLabel()`, koja se koristi za generisanje naziva temena grafa na osnovu njegovog rednog broja (ukoliko naziv temena grafa nije prosleđen od strane korisnika). Takođe su implementirani i operatori: $==, != , =, <$, dok su operatori $<<$ i $>>$ su implementirani kao prijateljske funkcije ove klase.

Graf je predstavljen klasom `Graph`. U poglavlju 2.4 je objašnjeno da se graf najčešće predstavlja matricom susedstva ili listom susedstva. U klasi `Graph`, graf se čuva preko liste susedstva, kao vektor listi temena grafova:

```
vector< list<GraphNode> > adjacencyList;
```

U poglavlju 2.4 su objašnjeni razlozi zbog kojih se ova struktura podataka češće koristi, kao i njene mane i prednosti u odnosu na matricu susedstva.

Ova klasa je praktično osnovna klasa za rad sa grafovima. U njoj je preko javnih funkcija definisano mnoštvo metoda koje se koriste za rad sa grafovima, počev od metoda za modifikaciju grafa (metode za dodavanje novih temena, brisanje postojećih, dodavanje novih ivica, brisanje postojećih ivica). U ovoj klasi se takođe nalaze i metode koje se koriste za ,,izvršavanje upita" nad grafovima (informacije o ukupnom broju temena grafa, dobijanje spiska svih temena, dobijanje indeksa nekog temena grafa u listi susedstva, dobijanje temena grafa koje se nalazi na odgovarajućoj poziciji u listi susedstva, provere da li neka grana postoji u grafu, da li neko teme postoji u grafu, dobijanja vektora svih suseda nekog temena, provere da li je graf povezan, 2-povezan, neusmeren i mnogih drugih). U ovoj klasi se nalazi i dve metode za obilazak grafa — BFS i DFS obilazak grafova. Mada je lista susedstva način reprezentacije grafa koji se koristi u ovoj klasi, omogućeno je dobijenje i druge reprezentacije grafa — matrice susedstva. U tu svrhu se koristi metoda `getAdjacencyMatrix()`. Kao i kod klase za rad sa temenima grafova `GraphNode` i u ovoj klasi su definisani mnogi operatori.

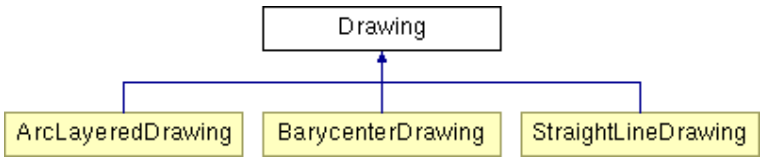
Ove metode predstavljaju samo deo metoda koje se koriste za rad sa grafovima. Detaljni pregled svih metoda se nalazi u dodatku ovog rada (dokumentacija za korišćenje ove biblioteke).

Pored javnih funkcija, postoje i privatne funkcija koje su sakrivene od ,,klijentskog" koda, a koje se koriste iz javnih metoda ove klase. U ovu grupu funkcija spadaju funkcije koje vode računa o radu sa memorijom, kao i neke ,,uslužne" funkcije koje koriste javne funkcije ove klase.

Klasama `GraphNode` i `Graph` je predstavljen graf. Ove dve klase omogućavaju rad sa grafovima. Samo crtanje grafa je implementirano preko klase `Drawing` i iz nje izvedenih klasa.

Klasa `Drawing` je apstraktna klasa koja se koristi kao bazna klasa za sva crtanja. Iz ove klase su nasleđene redom klase:

- `ArcLayeredDrawing` — klasa koja se koristi za lučno-slojevito crtanje grafova;
- `BarycenterDrawing` — klasa koja se koristi za baricentrično crtanje grafova;



Slika 5.1: Klasa `Drawing` i klase izvedene iz nje: `ArcLayeredDrawing`, `BarycenterDrawing` i `StraightLineDrawing`

- `StraightLineDrawing` — klasa koja se koristi za pravolinijsko crtanje planarnih grafova.

Napomenimo, pre nego što detaljnije objasnimo upotrebu ovih klasa, da je usled ove organizacije crtanja veoma lako dodati novo crtanje. Sve što je potrebno uraditi jeste pravljenje nove klase koja će biti nasleđena od klase `Drawing` i koja će reprezentovati to novo crtanje.

Klasa Drawing u sebi ima enkapsulirane osnovne elemente koji su potrebni za crtanje. Svaka od izvedenih klasa eventualno ima neka svojstva karakteristična za dato crtenje.

Funkcija:

```
map<int, struct Point> getCoordinates();
```

je funkcija pomoću koje je moguće dobiti koordinate temena grafa. Samo računanje koordinata se postiže čisto virtualnom zaštićenom funkcijom:

```
virtual void computeCoordinates()=0;
```

Funkcija članica ove klase je i funkcija:

```
bool draw();
```

Pomoću ove funkcije se realizuje samo crtanje grafa, odnosno upisivanje crtanja u fajl u `GCLC` formatu.

U zavisnosti od vrste crtanja, ono može biti konfigurabilno, odnosno, moguće je napraviti varijacije na jedno crtanje i time uticati na vizuelni izgled nacrtanog grafa. Ovi parametri koji utiču na vizuelni izgled crtanja se prosleđuju kroz objekat klase `Settings`. Ovom klasom su predstavljena generička podršavanja svakog od crtanja. Ovom klasom se određuje tip crtanja koje je potrebno napraviti, pravac crtanja grafa (za neke od crtanja je ovo moguće podesiti), format crtanja. Takođe, objektom ove klase je moguće postaviti i faktore skaliranja i transliranja, kao i tačnost i maksimalni broj iteracija (poslednja dva parametra se mogu postaviti za baricentrično crtanje). Mada je moguće objektom ove klase bitno uticati na crtanje na višestruke načine, radi što jednostavnije upotrebe ove biblioteke, data je mogućnost prosleđivanja `Settings` objekta u podrazumevanom stanju, u kom slučaju se generiše podrazumevano crtanje. Pored postavljanja parametara crtanja klase `Settings` preko konstruktora, ova klasa ima i funkcije koje se mogu koristi za naknadno postavljanje ovih parametara, kao i njihovo čitanje.

Prilikom dizajniranja klase `Drawing` i klasa izvedenih iz nje, cilj je bio omogućiti ,,klijentskom" kodu konzican i jednostavan interfejs, kojim bi bilo moguće crtanje grafova na što jednostavniji način. Iz ovog razloga veliki broj funkcija je privatan ili zaštićen i na taj način sakriven od ,,klijentskog" koda. Te zaštićene i privatne funkcije se uglavnom koriste za definisanje i crtanje temena i grana u različitim stilovima i formatima.

Kao što je već ranije napomenuto, konkretna crtanja podrazumevaju definisanje novih parametara potrebnih za to crtanje. Klasa `ArcLayeredDrawing` definiše i nivoe potrebne za ovo crtanje (u poglavlju 3.2 na strani 29) je definisano ovo crtanje. Ovi nivoi se čuvaju kao vektori celobrojnih vektora:

```
vector<vector <int> > layers;
```

U ovoj klasi su definisane i dodatne (mahom privatne) funkcije potrebne za ovo konkretno crtanje. Te funkcije uključuju funkcije za određivanje x i y koordinata crtanja, dobijanja najšire ,,lučne" grane, određivanje potomaka određenog nivoa, generisanje svih nivoa potrebnih za crtanje i tako dalje.

Klasa `BarycenterDrawing` definiše svoje (opet mahom privatne) funkcije potrebne za ovo crtanje. Uz to, ovom crtanju je potrebno proslediti i skup fiksiranih temena. Taj skup je realizovane preko mape:

```
map<int, struct Point> fixedVertices;
```

`struct Point` je struktura koja se koristi na više mesta u kodu i predstavlja tačku u ravni. Definisana je sa x i y koordinatom koje su tipa `float`.

Tokom rada algoritma za računanje baricentričnog crtanja potrebno je voditi računa o skupu temena za koje više ne treba računati nove koordinate (njihove koordinate su već dostigle tačku konvergencije). Ovi skupovi temena su realizovani preko mapa:

```
map<int, bool> needsXUpdating, needsYUpdating;
```

Kao i kod klase `ArcLayeredDrawing` i u ovoj klasi je većina novih funkcija privatna ili zaštićena. Neke od funkcija potrebnih za ovo crtanje su funkcija za postavljanje fiksiranih koordinata, funkcija koja određuje da li je potrebno nastaviti iterativni algoritam za određivanje koordinata temena, sama funkcija koja iterativnim algoritmom određuje koordinate temena i tako dalje.

Za pravolinijska crtanja planarnih grafova se koristi klasa `StraightLineDrawing`. Za ovo crtanje je potrebna struktura kanonskog uređenja (definicija ovog uređenja i algoritam za njegovo određivanje se nalazi u poglavlju 4.3 na strani 42). Kanonsko uređenje je definisano preko strukture `struct CanonicalOrder`, koja se sastoji od članova:

```
//! Canonical ordering
vector<int> canonicalOrdering;
//! Map of outer cycles of Gk
map<int, vector<int> > c0Gk;
//! Map of neighbours of vertex vk
map<int, vector<int> > vk_neighbours;
```

U biblioteku su uključene još dve klase, a koje se koriste u pomoćne svrhe. To su klase `GraphUtil` i `TestUtil`. Sve javne funkcije obe ove klase su statične. Kao što je već napomenuto funkcije ove klase se mogu koristiti u pomoćne svhe uglavnom za vreme debagovanja. Neke od funkcija koje se nalaze u ovim klasama uključuju funkcije za proveru da li se dati element nalazi u prosleđenom vektoru, ispisivanje vektora, ispisivanje nivoa u lučno-slojevitom crtanju crtanju, ispisivanje stanja objekta klase `Settings`, alociranje i dealociranje memorije za rad sa matricom susedstva i tako dalje. Imajući u vidu ,,servisni" karakter ove dve klase, one ovde neće biti detaljnije objašnjene.

U ovom poglavlju je dat kratak prikaz klasa, enumeracija u struktura koje su korišćene u ovoj biblioteci. Za detaljniju dokumentaciju treba koristi dodatak.

U originalnom obliku baricentrične metode, postoji nekoliko situacija u kojima se ne dobijaju vizuelno prihvatljivi crteži. Mogući problemi su poklapanje koordinata dva temena grafa (ovaj problem može da se javi kod temena stepena jedan ili kada u grafu postoji više temena koja imaju iste susede) ili delimična preklapanja grana. Kako bi ovi problemi bili izbegnuti, u biblioteci koja prati ovaj rad je napravljno nekoliko modifikacija originalne baricentrične metode.

5.3 Integracija sa programom GCLC

U prethodnom poglavlju je opisana razvijena bibioteka za crtanje grafova. Ova biblioteka je integrisana u program `GCLC` ([3], [4] i [5]) i u okviru njega je sada moguće crtati opšte grafove metodama koje su implementirane u biblioteci. U skladu sa tim, proširen je jezik `GCLC`-a, a na nižem nivou kôd `GCLC`-a koristi nekoliko ključnih metoda iz javnog interfejsa klasa biblioteke. Grafovi se u `GCLC`-u mogu koristiti slično kao ostali objekti, pa ih je moguće transformisati, menjati u okviru animacije, itd.

Lučno-slojevito crtanje se dobija komandom `drawgraph_a` u `GCLC`-u. Sintaksa ove komande je:

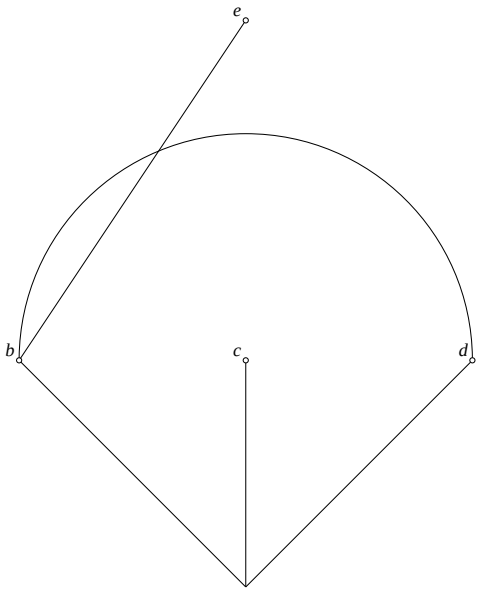
```
drawgraph_a <p_id> <n1> <n2> <list_of_nodes> <list_of_edges>
```

gde je `<p_id>` centar crteža grafa, `<n1>` je širina crteža grafa, dok je `<n2>` ugao rotacije sa centrom u `<p_id>`. `<list_of_nodes>` i `<list_of_edges>` redom predstavljaju listu temena i listu grana.

Primer 5.3.1 *Primer crtanja grafa lučno-slojevitom metodom u programu GCLC.*

```
drawgraph_a P 0 0
{ _a b c d e }
{
  _a b
  _a c
  _a d
  b  d
  b  e
}
```

Crtanje baricentričnom metodom u programu `GCLC` se postiže komandom `drawgraph_b`. Sintaksa ove komande je:



Slika 5.2: Lučno-slojevito crtanje grafa iz primera 5.3.1

`drawgraph_b <id> <list_of_nodes> <list_of_edges>`

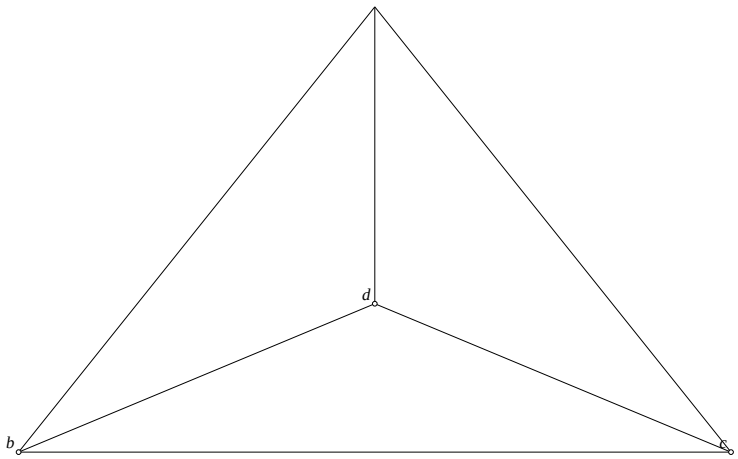
`<id>` se koristi kao identifikator grafa, preko koga se kasnije eventualno može pristupati njegovim temenima. `<list_of_nodes>` i `<list_of_edges>` redom predstavljaju listu temena i listu grana.

Zadavanjem parametara u `GCLC` komandama `drawgraph_a`[4] i `drawgraph_b` crtež grafa je jedinstveno određen.

Primer 5.3.2 *Primer crtanja grafa baricentričnom metodom u programu `GCLC`.*

```
drawgraph_b G
{ _a A
  b  B
  c  C
  d  _
}
{
  _a b
  _a c
  _a d
  b  d
  b  c
  c  d
}
```

[4] U slučaju komande `drawgraph_a` crtež grafa je jedinstveno određen do na poredak temena i grana.



Slika 5.3: Baricentrično crtanje grafa iz primera 5.3.2

Glava 6

Pregled literature i postojećih alata

U ovoj glavi će biti dat pregled literature i postojećih alata. U prvom poglavlju će biti dat pregled literature koja je korišćena prilikom pisanja ovog rada, dok će u drugom poglavlju biti dat pregled postojećih alata koji se koriste za crtanje grafova, kao i biblioteka i korisnih linkova.

6.1 Pregled literature

Knjiga *,,Graph Drawing - Algorithms for the Visualization of Graphs"*, (autori: Giuseppe Di Battista, Peter Eades, Roberto Tamassia, Ioannis G. Tollis) [2] predstavlja verovatno najbolji izbor u oblasti crtanja grafova. Knjiga predstavlja presek najznačajnijih algoritama za crtanje opštih grafova, ali i specijalnih grafova. Autori polaze od osnovnih pojmova, kao i opšteg predstavljanja metodologija koje se koriste za crtanje grafova. Kroz glave koja slede se predstavljaju algoritmi koji se koriste u tim metodologijama. Autori polaze od osnovnih algoritama za crtanje stabala, preko raznih algoritama koji se koriste za crtanja planarnih grafova. Neke od predstavljanih metodologija su: mrežnog toka (*eng. network flow*), inkrementalne tehnike, tehnike za konstruisanje ortogonalnih crteža neplanarnih grafova na celobrojnoj mreži, tehnike za crtanje grafova hijerarhijskim pristupom, tehnike za crtanje grafa koršćenjem metoda iz metodologije usmeravanja silom i druge.

Kao preduslov za korišćenje ove knjige, autori od čitaoca očekuju poznavanje osnovnih algoritama i struktura podataka, a knjigu namenjuju kao literaturu za kurseve sa postdiplomskih studija, kao i za istraživače koji se bave crtanjem grafova. Većina glava se završava vežbama i problemima, koji su posvećeni utrvrđivanju tehnika koje su predstavljene u toj glavi, dok su neke kompleksnije i predviđene su za korišćenje na naprednijim kursevima.

Oblast crtanja planarnih grafova je znatno uža nego oblast crtanja opštih grafova, pa je samim tim i izbor literature manji. Svakako jedna od najboljih knjiga u ovoj oblasti je *,,Planar Graph Drawing"*, (autori: Takao Nishizeki i Saidur Rahman) [1]. Radi se o knjizi u kojoj je predstavljena teorija i algoritmi za crtanje planarnih grafova. Knjiga polazi od nekih osnovnih uvoda u crtanje planarnih grafova, da bi dalje kroz knjigu bili predstavljene osnovne vrste

crtanja planarnih grafova, kao i primene crtanja planarnih grafova. Uvodni deo knjige je posvećen i nekim osnovnim definicijama i teoremama iz polja crtanja planarnih grafova, kao i nekim algoritamskim osnovama uopšte. Ostatak knjige je posvećen raznim vrstama crtanja različitih tipova planarnih grafova: pravolinijsko crtanje planarnih grafova na celobrojnoj mreži, konveksno crtanje planarnih grafova, pravougaono crtanje planarnih grafova, box-pravougaono crtanje planarnih grafova, ortogonalno crtanje planarnih grafova, itd. Autori su knjigu zamislili kao literaturu koja se može koristiti na postdiplomskim kursevima algoritmike, teorije grafova, crtanja grafova, vizualizacije informacija i drugim.

Generalno, druga knjiga [1] ima znatno više teorijski pristup polju crtanja (planarnih) grafova nego prva knjiga [2], i materijal u njoj je znatno formalnije izložen. Za razliku od nje [2] je znatno praktičnija sa većim brojem raznovrsnijih algoritama, i njena upotreba je svakako praktičnija.

6.2 Pregled postojećih alata

- **Graphviz — Graph Visualization Software**

  (`http://www.graphviz.org/`) — radi se o open source alatu koji se koristi za vizuelizaciju grafova. Ovaj program ima jednostavan jezik koji omogućava zadavanje grafa u tekstualnom obliku. Tekstualni fajl u kome se nalazi opis grafa se zatim kompajlira u željeni grafički format. U okviru samog programa je implementirano više različitih metoda za vizuelizaciju grafova. Program je vrlo konfigurabilan i omogućava crtanje grafova sa različitim oblikom i bojom temena kao i različitom vrstom grana. Moguće je definisanje i podgrafova, koji se dalje mogu integrisati u crtež. Postoje verzije programa za `Linux`, `Windows` i `Mac`. Na sajtu je moguće naći i veoma detaljnu dokumentaciju.

- **uDraw(Graph)**

  (`http://www.informatik.uni-bremen.de/uDrawGraph/en/index.html`) — alat za generisanje crteža grafova. Ovaj alat je ranije bio poznat kao daVinci, međutim, 2005. godine je ime programa promenjeno. Program se koristi za automatsko generisanje crteža grafova, odnosno dijagrama, vizualizacija raznih struktura, hijerarhija itd. Program dolazi u verzijama za različite operativne sisteme, između ostalog za `Linux`, `FreeBSD`, `Windows` i `Mac`.

- **Tom Sawyer Software** (`http://www.tomsawyer.com/home/index.php`) — komercijalni alat za vizualizaciju i analizu grafova. Omogućava hijerarhijska, kružna, simetrična, ortogonalna i mnoga druga crtanja.

- **GDToolkit — Graph Drawing Toolkit**

  (`http://www.dia.uniroma3.it/~gdt/gdt4/index.php`) — `C++` bilbioteka za rad sa grafovima i crtanje grafova. Pomoću ove biblioteke je moguće raditi sa nekoliko različitih vrsta grafova. Vizuelizaciju je moguće napraviti u skladu sa različitim vrstama estetskih kriterijuma i ograničenja.

- **The Open Source Java Graph Library** (`http://www.jgraph.com`) — open source komponenta za crtanje grafova za Java-u.

- **Walrus — Graph Visualization Tool**

  (`http://www.caida.org/tools/visualization/walrus/` — alat za interaktivnu vizualizaciju velikih usmerenih grafova u trodimenzionalnom prostoru. Radi se o projektu koji je dostupan pod GNU GPL licencom.

- **LEDA** (`http://www.algorithmic-solutions.com/leda/index.htm`) — `C++` biblioteka, čiji je jedan deo je posvećen radu sa grafovima.

- **VGJ (Visualizing Graphs with Java)**

  (`http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html`) — alat za crtanje grafova.

- **VCG** (`http://rw4.cs.uni-sb.de/~sander/html/gsvcg1.html`) — alat koji na osnovu tekstualne specifikacije grafa crta graf.

- **http://graphdrawing.org/** — sajt posvećen crtanju grafova.

Glava 7

Zaključci i dalji rad

Oblast crtanja grafova je izuzetno široka i ima mnogo praktičnih primena, u najrazličitijim oblastima (o primenama crtanja grafova je bilo reči u poglavlju 1.2 na strani 9). Mogućnosti za nastavak ovog rada su veoma velike.

Prilikom dizajniranja biblioteke za crtanje grafova vođeno je računa o tome da dodavanje novih crtanja bude jednostavno. Implementiranje nove metode za crtanje grafova i njena integracija u ovu biblioteku se postiže nasleđivanjem klase `Drawing`. Nova crtanja mogu biti dodata u cilju vizuelizacije neke posebne klase grafova, gde je potrebno podatke prikazati na specifičan način, ali crtanja mogu biti dodavana i radi novih dodatnih vizualizacija opštih grafova. Među specijalnim vrstama grafova najzanimljiviji su planarni grafovi (Definicija 2.3.1 na strani 16). U ovom radu je predstavljena jedna metoda za pravolinjsko crtanje planarnih grafova — metoda pomeraja (poglavlje 4.4 na strani 46). Pored ove metode, postoji još veoma mnogo različitih metoda koje se mogu koristi za razne vrste crtanja planarnih grafova. Detaljni spisak svih metoda za crtanje planarnih grafova se može naći u poglavlju 4.1 na strani 39.

U ovom trenutku biblioteka koja prati ovaj tekst podržava crtanje grafova u kojima su temena predstavljena krugovima, dok su strane predstavljene dužima, odnosno polukrugovima (u zavisnosti od vrste crtanja). Bilo bi interesantno proširiti biblioteku dodavanjem novih stilova za crtanje grafova u okviru već postojećih (i eventualno novih metoda). To proširenje stilova bi moglo da uključuje mogućnost predstavljanja temena različitim geometrijskim figurama, koje bi mogle biti obojene različitim bojama, dok bi grane mogle biti predstavljene različitim vrstama linija (različite debljine linija, različiti dizajni linija, itd).

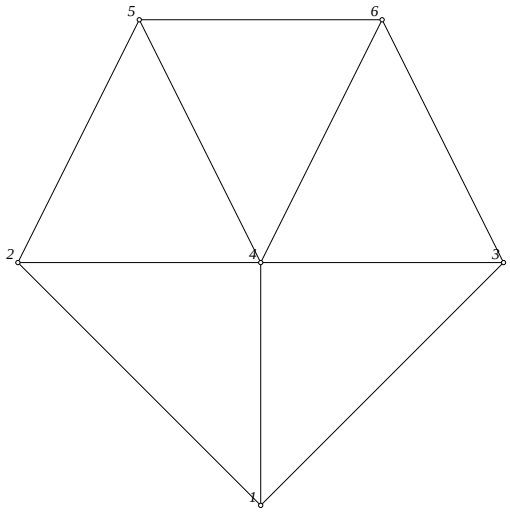
Biblioteka (tačnije deo bilbioteke koji se odnosi na samo crtanje grafova) je uglavnom koncentrisana na crtanje grafova u `GCLC`-u. Međutim, veoma lako bi bilo moguće proširiti ovaj izlaz na različite druge formate. Ti formati bi mogli da uključuju `bmp, png, jpg, eps, swf` itd, ali i u mnoge druge različite sisteme (recimo `OpenGL`).

Bilo bi interesantno dodati mogućnost animiranja grafova u biblioteku. Animiranim grafovima bi bilo moguće prikazati neku vrstu dinamike odnosa među entitetima. Pored ove činjenice, bilo bi zanimljivo animirati i same algoritme koji su korišćeni za crtanje grafova (recimo, konvergencija koordinata ka finalnom crtežu kod baricentrične metode).

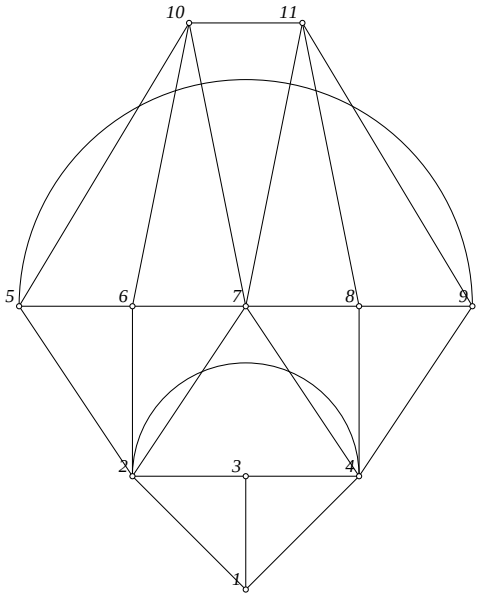
Glava 8

Primeri

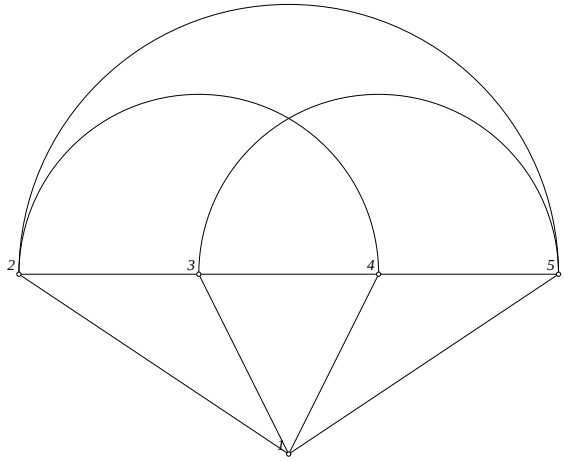
U ovoj glavi će biti dato nekoliko primera crtanja planarnih grafova koja su implementirana u okviru ovog rada. Sami crteži su napravljeni korišćenjem programa `GCLC`.



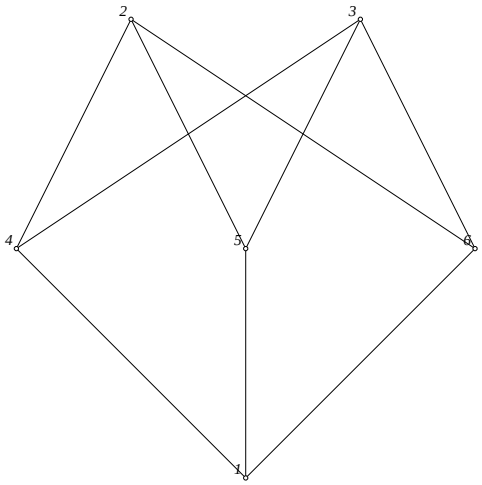
Slika 8.1: Lučno-slojevito crtanje grafa sa 6 temena. Na slici 8.5 je nacrtan isti graf baricentričnom metodom.



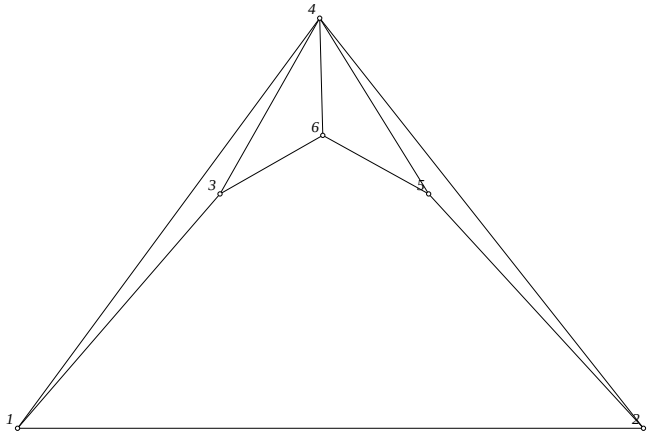
Slika 8.2: Lučno-slojevito crtanje grafa sa 11 temena. Na slici 8.6 je nacrtan isti graf samo baricentričnom metodom.



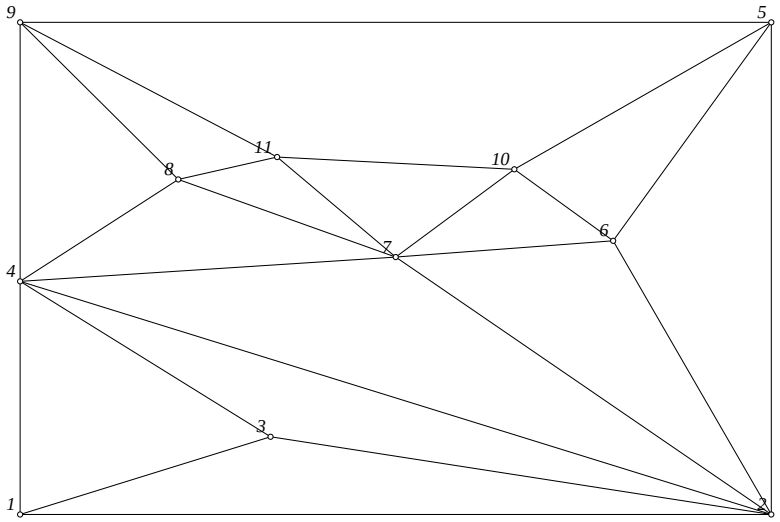
Slika 8.3: Lučno-slojevito crtanje grafa K_5



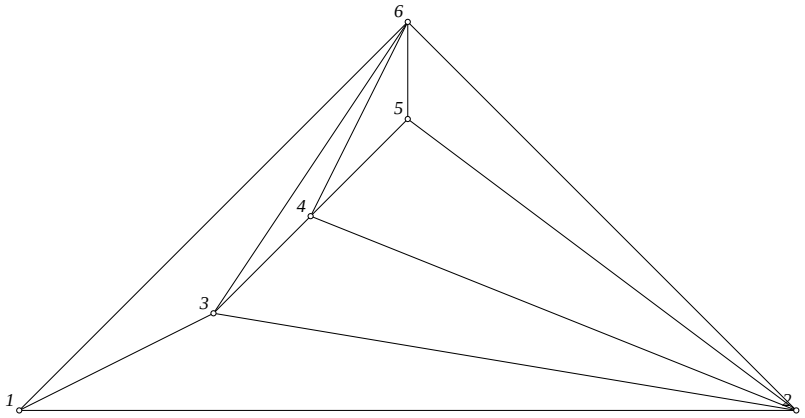
Slika 8.4: Lučno-slojevito crtanje grafa $K_{3,3}$



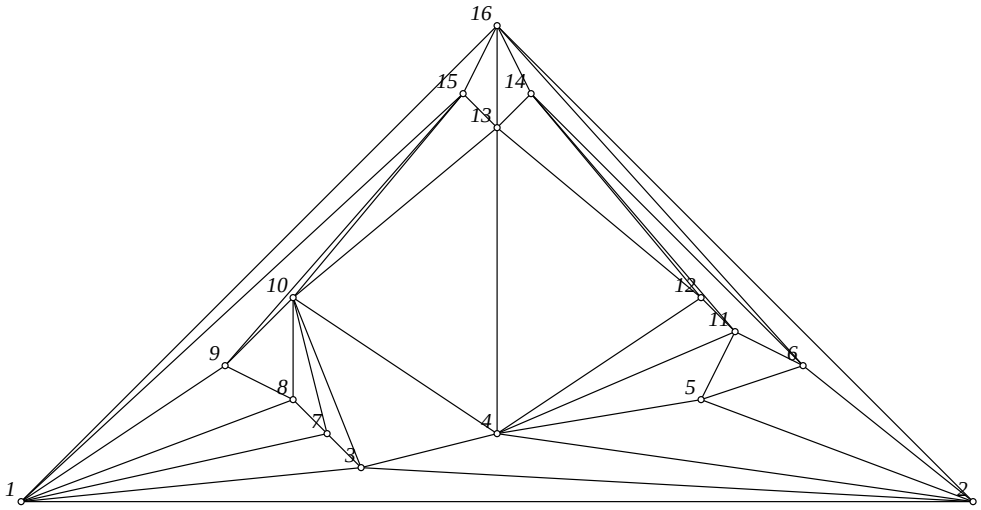
Slika 8.5: Baricentrično crtanje grafa sa 6 temena. Fiksirana temena 1, 2 i 3. Odabirom nekih drugih fiksiranih temena se dobijaju potpuno drugačija crtanja.



Slika 8.6: Baricentrično crtanje grafa sa 11 temena. Za fiksirana temena su temena 1, 2, 5, 9 i 4. Odabirom nekih drugih fiksiranih temena se dobijaju potpuno drugačija crtanja.



Slika 8.7: Pravolinijsko crtanje planarnog grafa sa 6 temena metodom pomeraja



Slika 8.8: Pravolinijsko crtanje planarnog grafa sa 16 temena metodom pomeraja

Sažetak / Summary

Polje crtanja grafova je veoma bogato. U prvom delu ovog rada je predstavljen uvod u rad sa grafovima kao i teorijske osnove rada sa grafovima, a u drugom delu su predstavljene tri metode: lučno-slojevita metoda, baricentricna metoda i metoda pomeraja. Lučno-slojevita i baricentrična metoda se koriste za crtanje opštih grafova, dok se metoda pomeraja koristi za pravolinijsko crtanje planarnih grafova. U sklopu ovog rada je implementirana i bilbioteka za crtanje grafova, koja je integrisana u program `GCLC` i kojom je omogućeno crtanje grafova u `GCLC`-u (iz tog razloga, sintaksa `GCLC`-a je proširena). Na kraju rada je dat pregled implementacije sa programom `GCLC`, pregled postojećih alata, kao i zaključci i dalji rad.

The field of graph drawing is very wide and has many applications. In the first part of this work, introduction to graph drawing was presented as well as some basic theory required for graph drawing. In the second part of the work, three methods were presented: the arc-layered method, the baricentric method and the shift method. The arc-layered method and the baricentric method are methods that can be used in drawing of general graphs, while the shift method is used in straightline drawing of planar graphs. Along side with this work, API for graph drawing was implemented as well. This API is integrated with the program `GCLC`. Thanks to this API, it is now possible to draw graphs in `GCLC` (for this purpuse its syntax is extended). At the very end of this thesis, there are chapters about implementation of these methods, overview of existing tools and literature for graph drawing, and the final conclusions and the directions for future work.

Literatura

[1] Takao Nishizeki, Md. Saidur Rahman, *Planar Graph Drawing*, Lecture Notes Series on Computing - Vol. 12, World Scientific Publishing, 2004

[2] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, Ioannis G. Tollis, *Graph Drawing - Algorithms for the Visualization of Graphs*, Prentice Hall, 1999

[3] `http://www.matf.bg.ac.yu/~janicic/gclc/`, Internet adresa GCLC-a

[4] Predrag Janicic and Ivan Trajkovic, *WinGCLC - a Workbench for Formally Describing Figures. In Proceedings of the 18th spring conference on Computer graphics (SCCG 2003), pages 251–256, Budmerice, Slovakia, April, 24-26 2003.* ACM Press, New York, USA.

[5] P. Janicic. GCLC - *A Tool for Constructive Euclidean Geometry and More than That.* In N. Takayama, A. Iglesias, and J. Gutierrez, editors, Proceedings of International Congress of Mathematical Software (ICMS 2006), volume 4151 of Lecture Notes in Computer Science, pages 58-73. Springer-Verlag, 2006.

[6] M.R. Garey and D.S. Johnson, *Crossing number is NP-complete*, SIAM, J. Alg. Disc. Methods, 4(3), pp. 312-316, 1983

[7] M.R. Kramer and J. van Leeuwen, *The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits*, (Eds.) F.P. Preparata, Advances in Cmputing Research, vol. 2: VLSI Theory, JAI Press, Reading, MA, pp. 129-146, 1984

[8] Knuth, D. E. (1963), *Computer-Drawn Flowchats*, Communications of the ACM, 6(9):555-563. 81

[9] A. Garg and R. Tamassia, *On the computational complexity of upward and rectilinear planarity testing*, SIAM, J. Comput., 31(2), pp. 601-625, 2001

[10] K. Wagner, *Bemerkungen zum vierfarbenproblem*, Jahresber. Deutsch. Math-Verien., 46, pp. 26-32, 1936

[11] I. Fáry, *On straight line representations of planar graphs*, Acta Sci. Math. Szeged, 11, pp. 229-233, 1948

[12] K. S. Stein, *Convex maps*, Proc. Amer Math. Soc., 2, pp. 464-466, 1951

[13] H. de Fraysseix, J. Pach and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10, pp. 41-51, 1990

[14] W. Schnyder, *Embedding planar graphs on the grid*, Proc. First ACM-SIAM Symp. On Descrete Algorithms, San Francisco, pp. 138-148, 1990