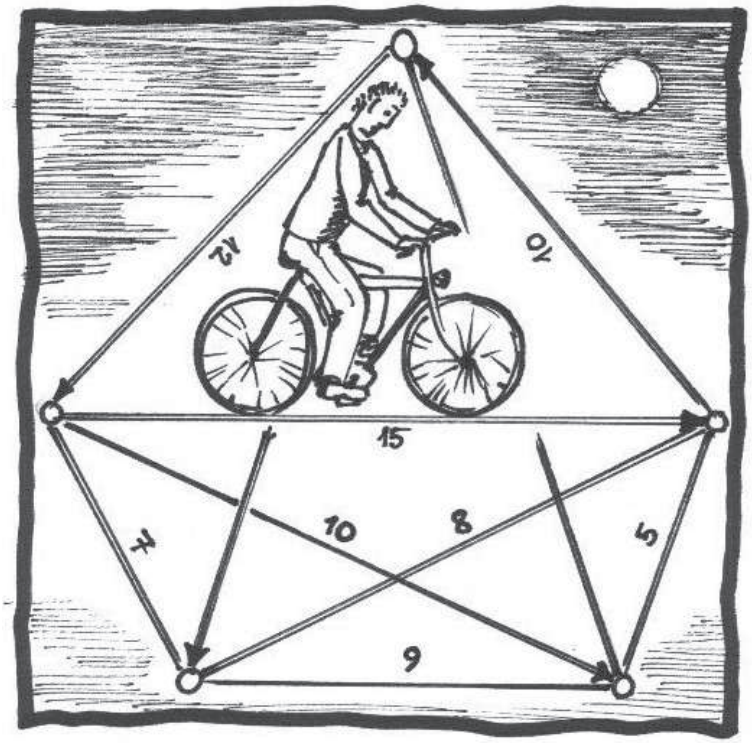
Poglavje 10

PETERSEN - odločanje na podlagi teorije grafov



Teorija grafov
Minimalno vpeto drevo
Primer: iskanje najkrajših poti z izbranimi algoritmi

10.1 Teoretično ozadje

10.1.1 Teorija grafov

Kadar govorimo o teoriji grafov, govorimo o veji matematike in računalništva, ki raziskuje lastnosti grafov. Začetke razvoja beležimo vse od druge polovice 19. stoletja. Razvila se je iz potrebe po reševanju konkretnih primerov v znanosti in tehnologiji. Graf definiramo kot množico objektov in reči, imenovane točke (vozlišča, vozil...), ki so povezane s povezavami (robovi, veje...). S teorijo grafov rešujemo številne primere v praksi. Grafe lahko razširimo z vpeljavo uteži, ko so pozitivna števila, prirejena vsaki povezavi. Če npr. graf predstavlja mrežo cest, lahko uteži predstavljajo dolžino vsake ceste. Če grafu dodamo utežene povezave, govorimo o mreži. Grafe uporabljamo predvsem pri metodi mrežnega planiranja.

Rečemo lahko, da graf G sestavljata neprazna množica elementov, ki jih imenujemo točke ali vozlišča grafa in seznam (neurejenih) parov teh elementov, ki jih imenujemo povezave grafa. Grafe si enostavno predstavljamo z njihovimi diagrami. Točke (vozlišča) ponazorimo s krogci, sosednji točki pa povežemo s črto.

Minimalno vpeto drevo

Minimalno vpeto drevo je je pomembno za reševanje problemov načrtovanja optimalnih (najpogosteje najcenejših) prometnih in drugih omrežij. Prometni problem lahko rešujemo s pomočjo neusmerjenega grafa. Vozlišča v grafu predstavljajo mesta, ki jih želimo povezati, povezave pa so označene z razdaljami, časom oz. cenami povezave med dvema krajema. Ker za povezavo vseh mest zadošča, da v grafu obstaja ena pot med vsakim krajem, problem definiramo kot iskanje minimalnega (najcenejšega) podgrafa, ki izpolnjuje ta pogoj. Takšen podgraf imenujemo minimalno vpeto drevo. Iskanje minimalnega vpetega drevesa v povezanem neusmerjenem grafu izvedemo s pomočjo dveh znanih algoritmov - Kruskalov in Primov.

Kruskalov algoritem

Kruskalov algoritem je model, ki se uporablja za iskanje minimalnega vpetega drevesa v omrežju. Drevo začnemo graditi iz najkrajše povezave, ki je že sama zase vpeto drevo in nato naprej iščemo najkrajše povezave, dokler nimamo celega vpetega drevesa (pri tem smo pozorni, da ne pride do cikla). V izbranem načinu gradimo drevo tako, da na vsakem koraku v rešitev dodamo povezavo z minimalno vrednostjo, ki ima največ eno izmed točk, ki so v enem delu rešitve. V

primeru, ko je prva točka povezave v enem delu rešitve in druga točka povezave v drugem delu rešitve, ta dva dela rešitve združimo.

Primov algoritem

Tudi Primov algoritem se uporablja za iskanje minimalnega vpetega drevesa v omrežju. Drevo začnemo graditi iz poljubnega vozlišča, ki je že samo zase vpeto drevo in nato dodajamo najkrajše povezave, dokler ne dobimo celega vpetega drevesa. V grafu poiščemo povezave, s katerimi povežemo vse točke, ki imajo najmanjšo vsoto povezav. Postopek računanja poteka tako, da v celotnem grafu izberemo najkrajšo povezavo med vozlišči v doslej zgrajenemu drevesu in vozlišči, ki še niso del drevesa in z njo ne naredimo cikla, kar je osnoven korak. Če sta dve povezavi z enakimi vrednostmi, izberemo tisto, ki ne naredi cikla. Če nobena povezava ne naredi cikla, izberemo eno izmed njiju. To povezavo dodamo in postopek ponavljamo, dokler obstajajo prosta vozlišča.

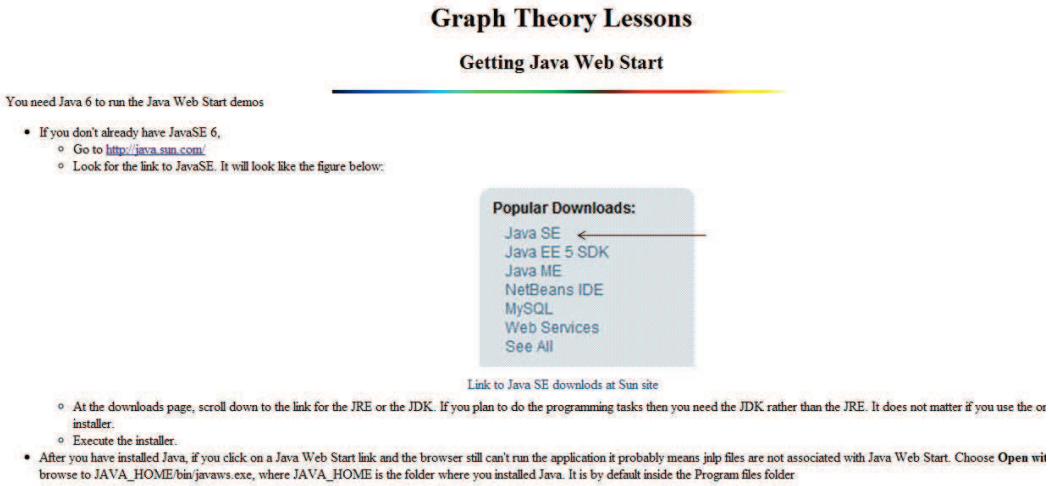
10.2 O programskem orodju

Petersen programsko orodje je nastalo pod okriljem avtorja Chrisa Mawata. Izdelujemo, urejamo in manipuliramo enostavne grafe ter preučujemo njihove lastnosti. Prikazujemo podatke o grafu (število točk in njihove stopnje), matriko sosednosti in število sestavnih delov; preverimo, če je graf dvostranski, če sta dva grafa izomorfna ali če je graf podgraf drugega. Petersen dokazuje Euler in Hamilton vezja, algoritme za iskanje minimalnih vpetih dreves ipd.

Pri pisanju programskega orodja je avtor poskušal rešiti čim več izobraževalnih in učnih problemov. Eden izmed razlogov nastanka je prikaz velikih in zapletenih grafov. Za pripravo takšnih grafov je potreben bistveno daljši čas, saj je že samo pridobivanje informacij iz velikih grafov mučno in dolgotrajno. Algoritmi v programu niso novi. Edina razlika med Petersen programskim orodjem ter poslovnimi in raziskovalnimi aplikacijami je v tem, da je njegov glavni cilj dobiti rezultate v najkrajšem možnem času. V izobraževalnih ustanovah pa je primarni cilj, da študent spozna in vidi grafičen prikaz nekega problema.

Prenos in namestitev

Programsko orodje Petersen je dostopno na spletnem naslovu Mathcove [33]. Deluje preko spletne aplikacije Java. V kolikor Jave nimamo že predhodno nameščene, na spletni strani *(Getting Java Web Start)* izberemo možnost *go to* Java [22] (glej Sliko 10.1).



Slika 10.1: Prenos Jave

Brezplačno Petersen programsko orodje prenesemo tako, da v razdelku *Getting Petersen*, iz spletne strani Mathcove [33] (glej Sliko 10.2), v 2. poglavju izberemo možnost *download petersen-3.2.1.jar* (glej Sliko 10.2).

Programsko okno

Pred pričetkom uporabe programskega orodja se informiramo o funkcijah in orodjih, ki so na voljo. Ob zagonu programskega orodja se v začetnem oknu pojavi menijska vrstica, katere funkcije razložimo v nadaljevanju (glej Sliko 10.3).

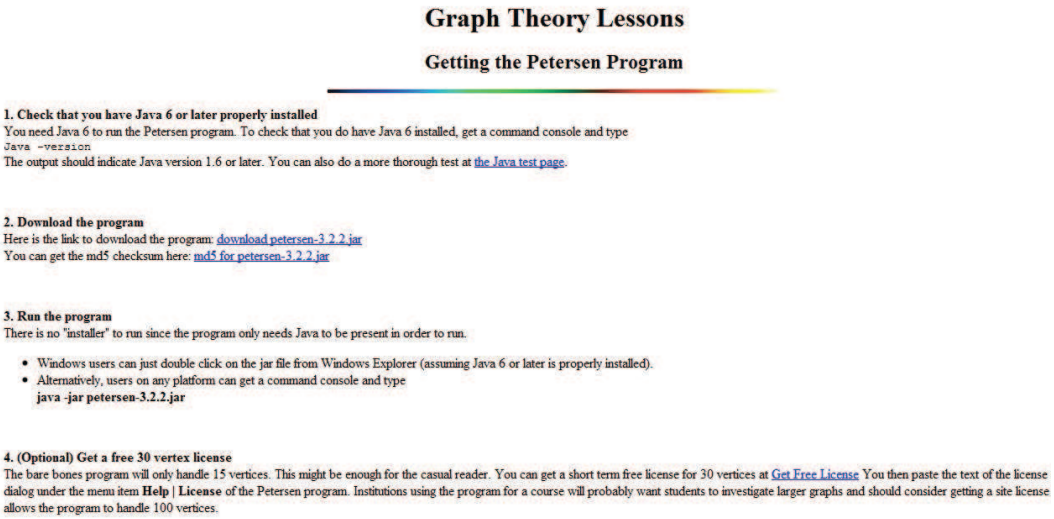
Menijska vrstica *Graph* omogoča številne možnosti urejanja in oblikovanja grafov. V primeru, ko izberemo možnost *Named Graph* in v nadaljevanju *Null Graph* se prikaže možnost določitve točk *(How many vertices?)*. V okence vpišemo število točk, ki bodo vključene v graf (glej Sliko 10.4).

V razdelku menijske vrstice *Graph - Named Graph* v *Complete* izbiramo med tremi razdelki. Izberemo možnost izrisa grafa z določitvijo števila povezav, ki bodo povezane z že narisanimi točkami (glej Sliko 10.5).

Ob izbiri razdelka *Complete Bipartite Graph* zapišemo število točk na levi in desni strani (glej Sliko 10.6).

V razdelku *Complete Tripartite Graph* izbiramo in zapišemo 3 vrste števil: število točk v prvi skupini, število točk v drugi skupini in število točk v tretji skupini (glej Sliko 10.7).

Razdelek *Circuit Graph* izriše graf, ki je povezan v cikel. Izberemo zgolj možnost števila točk, za katere želimo, da bodo vsebovane v določenem



Graph Theory Lessons

Getting the Petersen Program

1. Check that you have Java 6 or later properly installed
You need Java 6 to run the Petersen program. To check that you do have Java 6 installed, get a command console and type
`Java -version`
The output should indicate Java version 1.6 or later. You can also do a more thorough test at the Java test page.

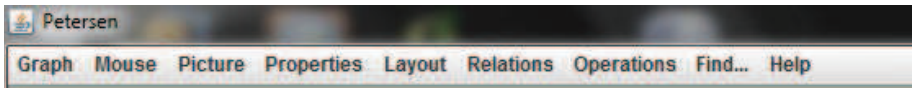
2. Download the program
Here is the link to download the program: download petersen-3.2.2.jar
You can get the md5 checksum here: md5 for petersen-3.2.2.jar

3. Run the program
There is no "installer" to run since the program only needs Java to be present in order to run.

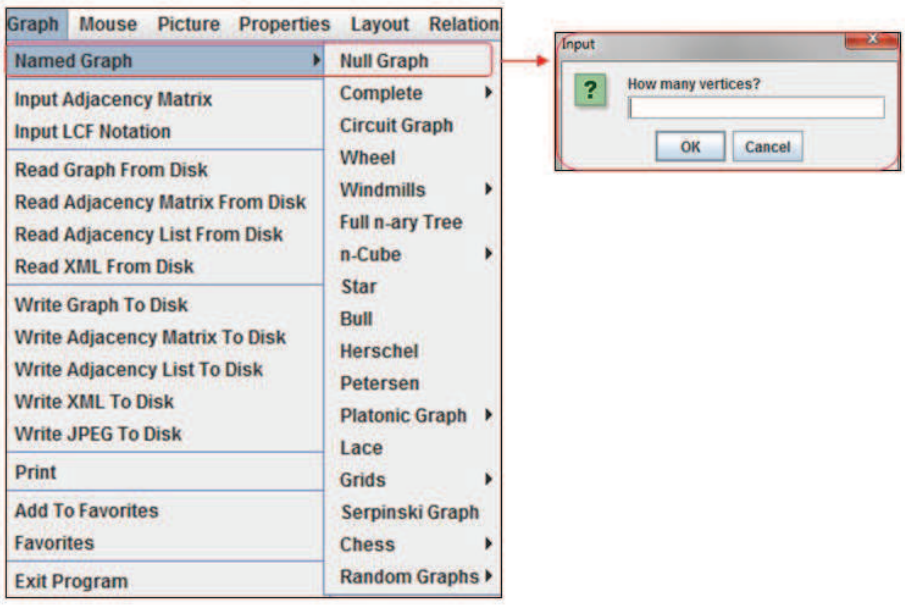
- Windows users can just double click on the jar file from Windows Explorer (assuming Java 6 or later is properly installed).
- Alternatively, users on any platform can get a command console and type
  java -jar petersen-3.2.2.jar

4. (Optional) Get a free 30 vertex license
The bare bones program will only handle 15 vertices. This might be enough for the casual reader. You can get a short term free license for 30 vertices at Get Free License You then paste the text of the license dialog under the menu item **Help | License** of the Petersen program. Institutions using the program for a course will probably want students to investigate larger graphs and should consider getting a site license allows the program to handle 100 vertices.

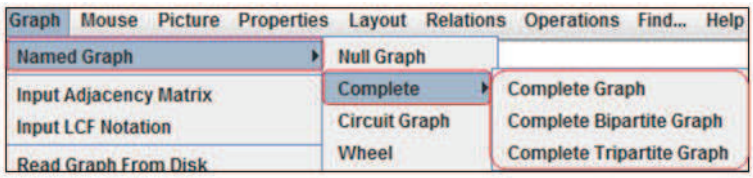
Slika 10.2: Prenos Petersen-ovega programskega orodja



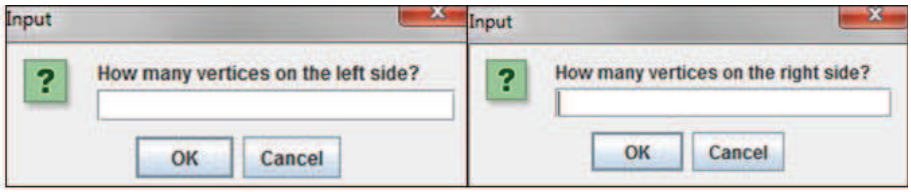
Slika 10.3: Menijska vrstica



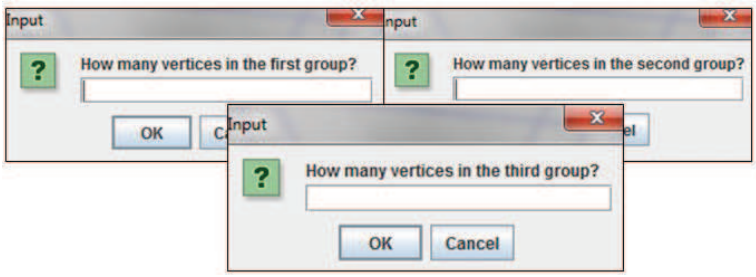
Slika 10.4: Določitev števila točk – Null Graph



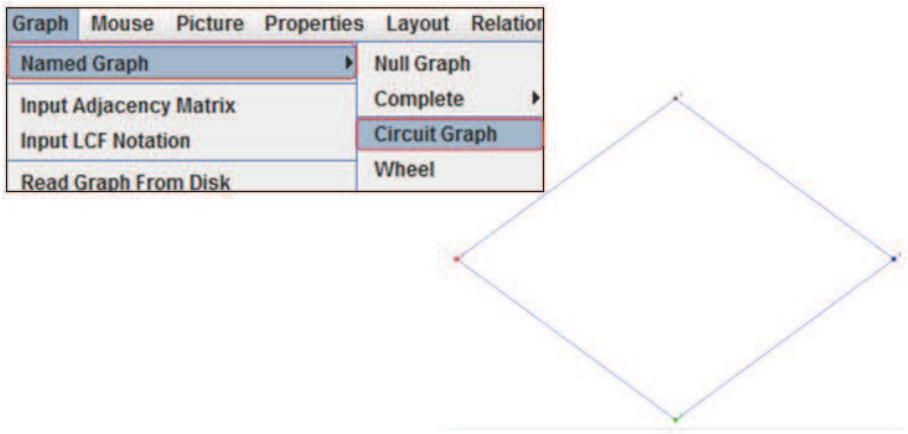
Slika 10.5: Določitev števila točk – Complete



Slika 10.6: Complete – Complete Bipartite Graph



Slika 10.7: Complete-Complete Tripartite Graph



Slika 10.8: Circuit Graph

cikličnem grafu (glej Sliko 10.8).

Ob izbiri razdelka *Wheel* se na grafu (npr. 4 povezane točke) izriše določeno število točk (npr. 5 točk). Točke se izrišejo na način, da je ena točka povezana z vsemi ostalimi na grafu (glej Sliko 10.9).

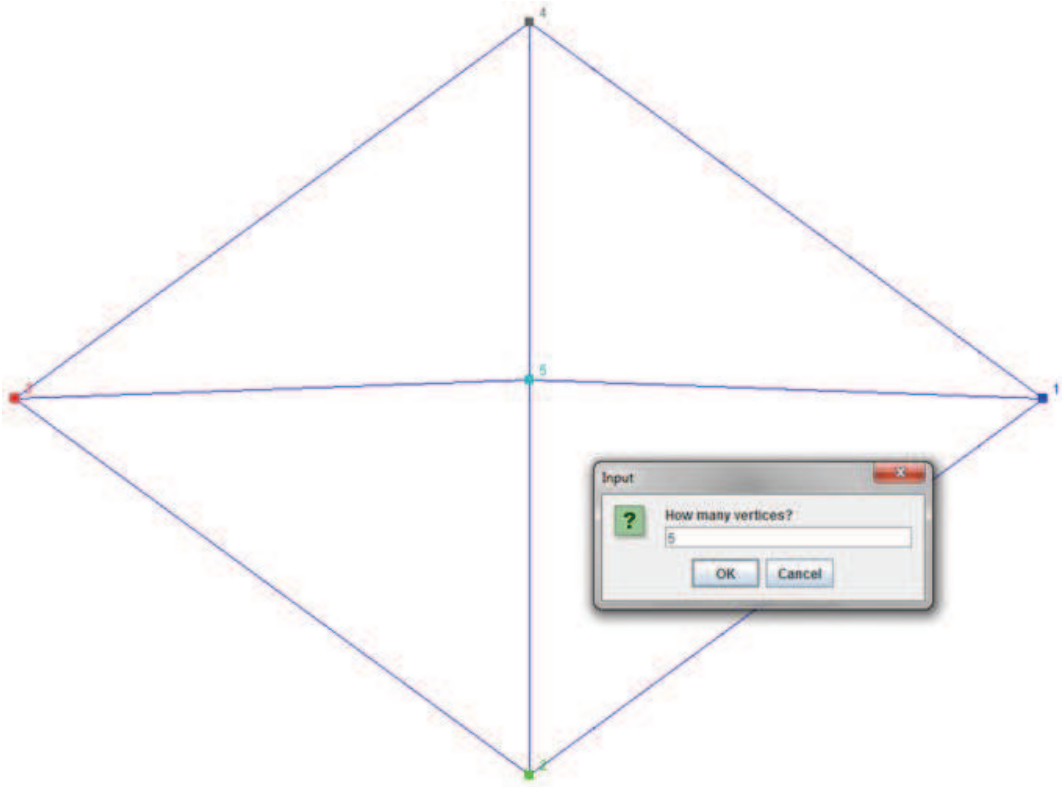
Izbiramo lahko med posameznimi oblikami grafov *(Windmills)*. Ti se lahko izrišejo v obliki metuljčka *(Butterfly)* ipd. (glej Sliko 10.10).

Full n-ary Tree prikazuje možnost izrisa razcepljenega grafa. Izberemo možnosti razcepov s po 2 točkama (glej Sliko 10.11).

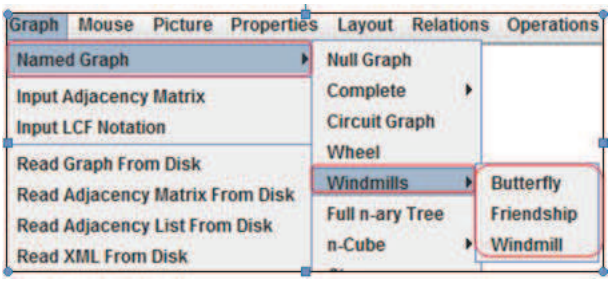
Izbiramo lahko tudi med izrisom različnih vrst grafov, kot so: *n-Cube, Star* (graf v obliki zvezde), *Bull, Herschel in Petersen graf* (glej Sliko 10.12).

Podana je še možnost izrisa *(Platonic Graph)* v obliki tetraedra, oktaedra ipd. (glej Sliko 10.13).

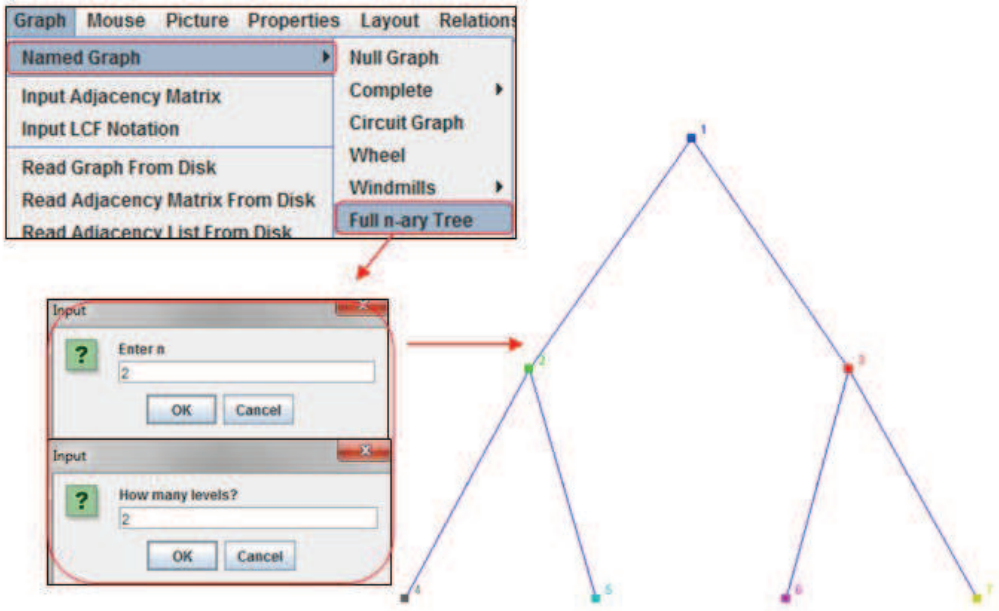
V menijski vrstici *Graph* so na voljo tudi druge možnosti: vnos podatkov za vhodno matriko sosednosti *(Input Adjancency Matrix)* in LCF zapis *(Input LCF*



Slika 10.9: Wheel



Slika 10.10: Windmills



Slika 10.11: Full n-ary Tree

Notation). V programu lahko odpremo že shranjen dokument na disku (graf - *Read Graph From Disk*, matrika sosednosti – *Read Adjacency Matrix From Disk* ipd.). V danem razdelku izberemo tudi možnost tiskanja grafa *(Print)* in izhoda iz programskega orodja *(Exit Program)* (glej Sliko 10.14).

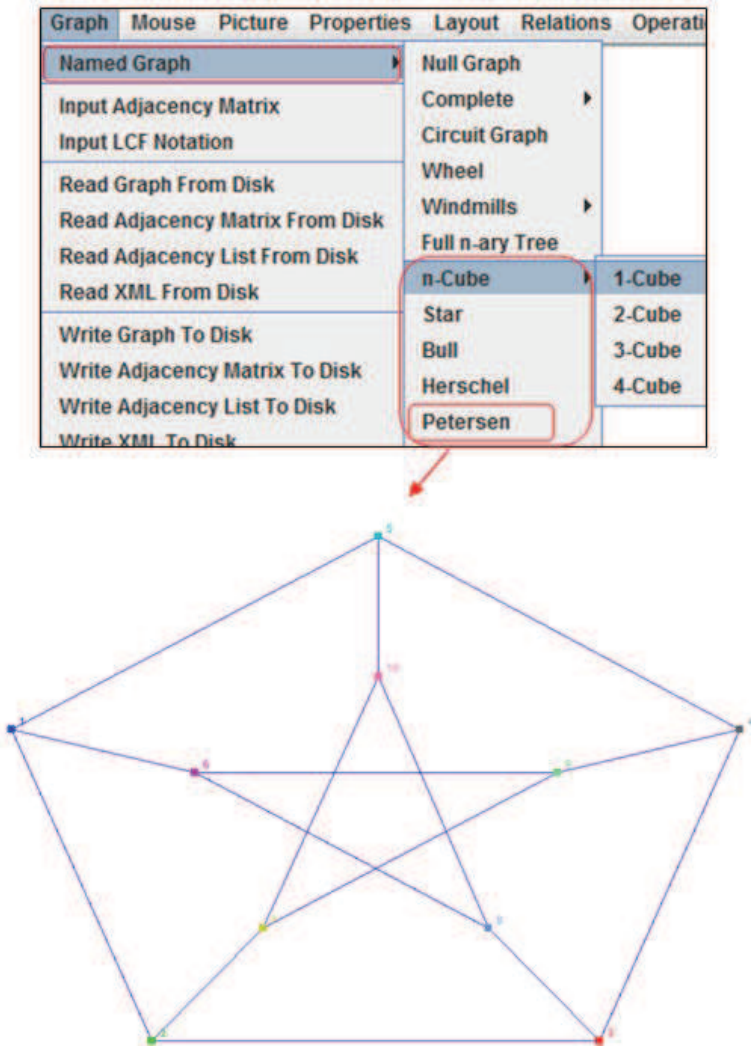
Na koncu menijske vrstice se nahaja razdelek *Pomoč oz. Help* (Slika 10.15). V razdelku *About* je naslov spletne strani, kjer so na voljo informacije za uporabo programskega orodja.

Na spletni strani najdemo še informacije iz posameznega področja (npr. klik na dano spletno stran: Mathcove [33], prikaže meni s številnimi podatki).

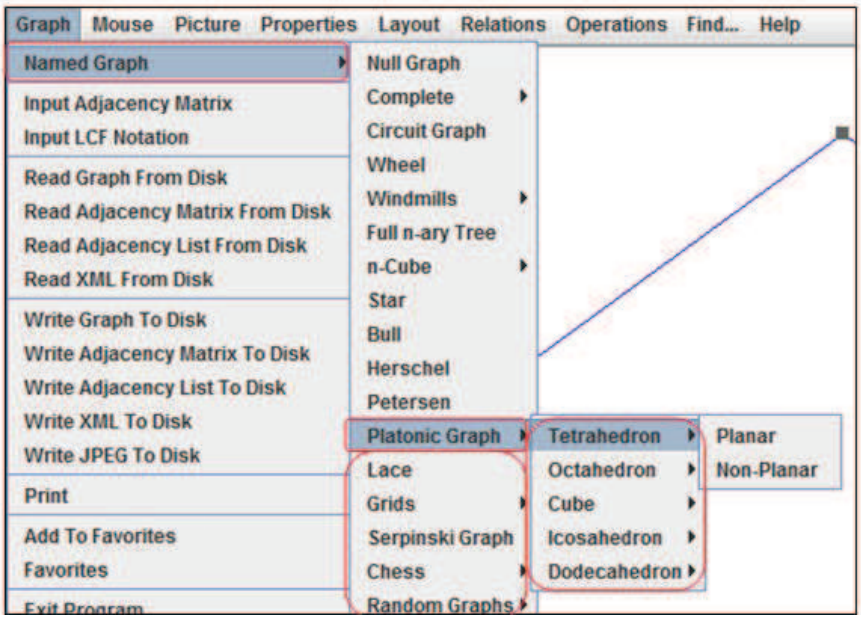
Pri poglobljenem proučevanju določenega področja je smiselno prebrati besedilo, zapisano pod posamezno lekcijo. Na koncu vsake lekcije je povezava, ki direktno preusmeri na področje, ki nas zanima. Na Sliki 10.16 je prikazana teorija o minimalno vpetem drevesu.

S klikom na povezavo (vijoličaste barve) se prikaže okno programskega orodja (glej Sliko 10.17).

Izriše se prazno modro okno, kjer imamo na voljo povsem lastno izbiro dodajanja ali brisanja povezav in linij (glej Sliko 10.18).

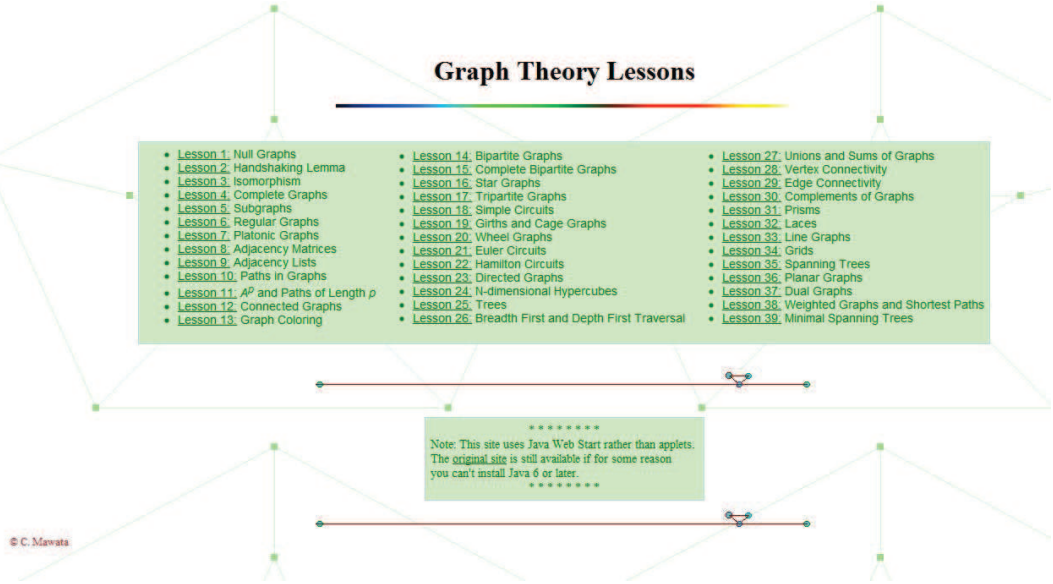


Slika 10.12: Različne vrste grafov – 1. del

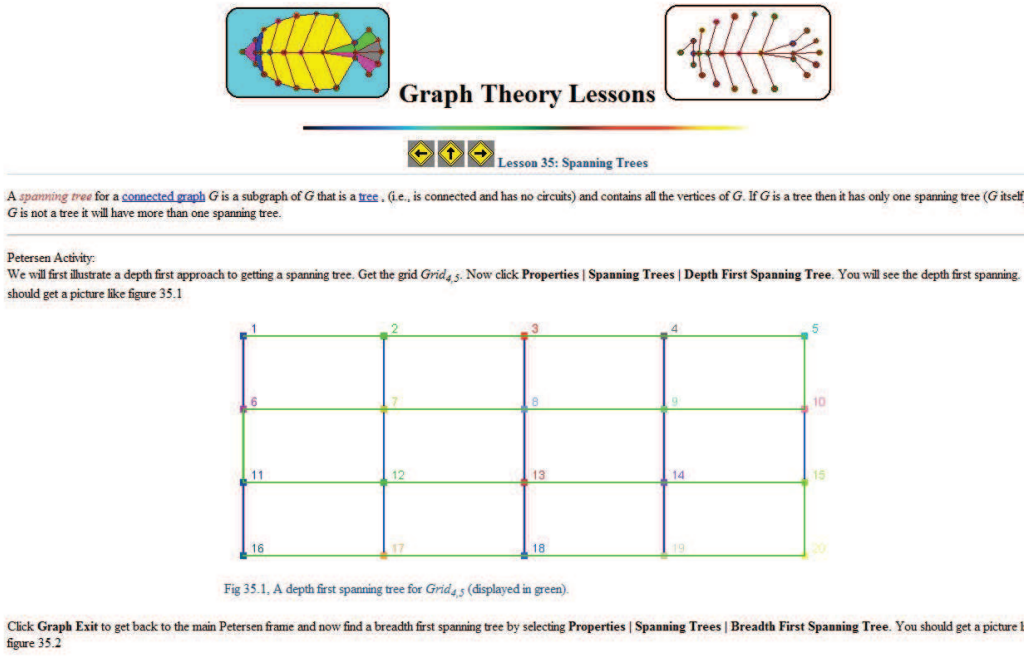


Slika 10.13: Različne vrste grafov – 2. del

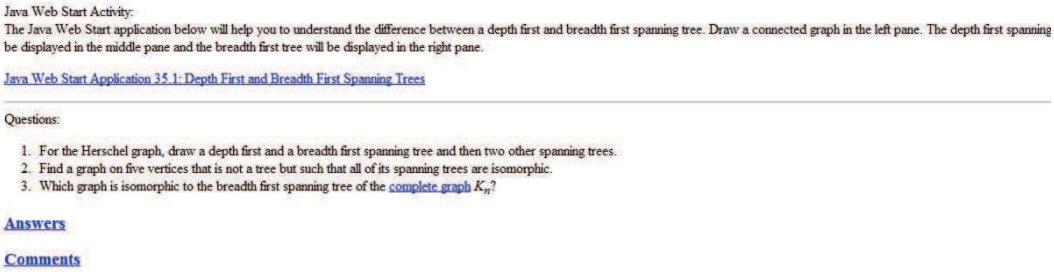
Slika 10.14: Menijska vrstica in razdelek Graph



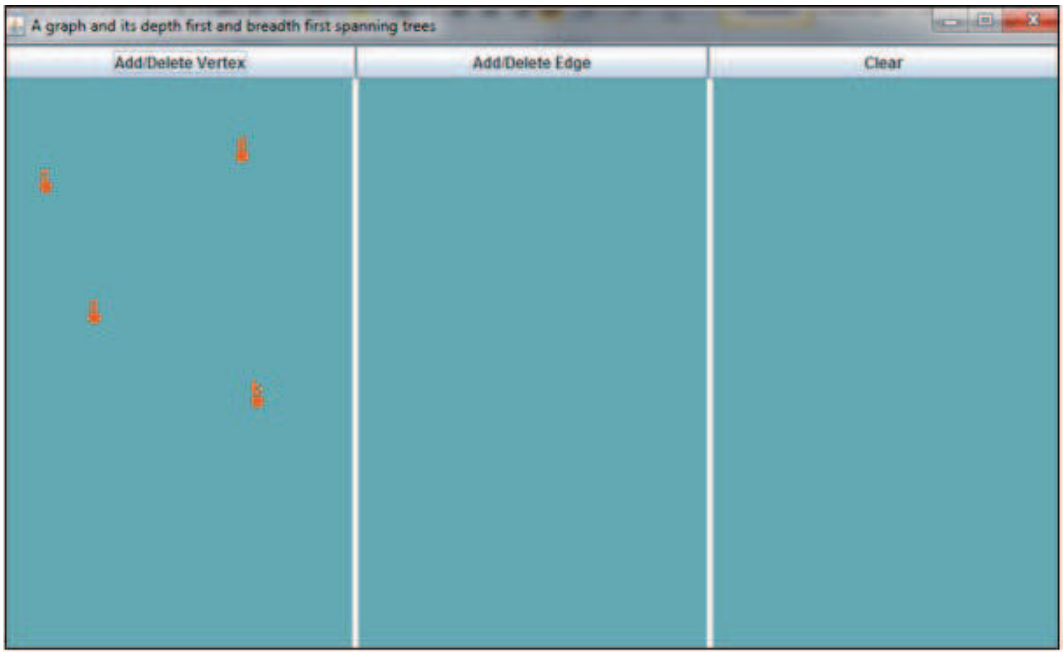
Slika 10.15: Dodatna pomoč pri uporabi programa



Slika 10.16: Dodatna literatura na temo minimalnega vpetega drevesa



Slika 10.17: Klik na direktno povezavo



Slika 10.18: Posamezna programska okna

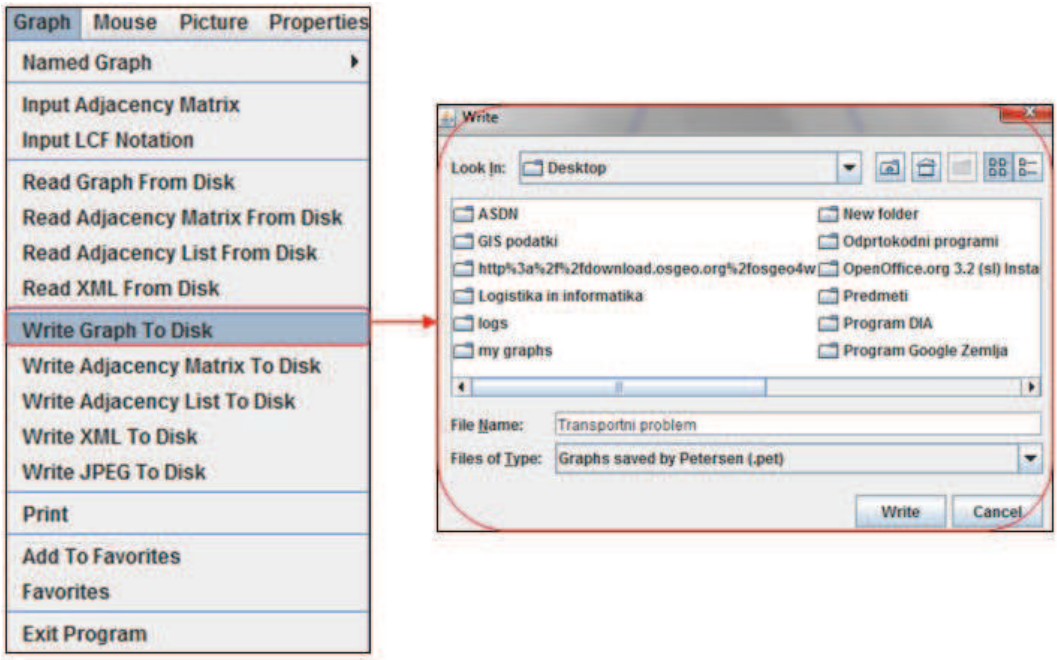
Problem

Z danim programskim orodjem v nadaljevanju prikažemo problem, opažen v resnični situaciji. V namišljenem podjetju OpenStorage moramo poskrbeti za transport komponent, namenjenih za izdelavo avtomobilov znamke X. Komponente (natančneje platišča) se v podjetje dostavljajo s cestnim transportom (prevoz s tovornjaki), pri čemer so mnogokrat izpostavljene oviram na poti, zato je potrebno izbrati pot, kjer ovire obidemo z minimalni stroški.
V ta namen s Petersen programskim orodjem prikažemo problem dostave platišč od dobavitelja do podjetja v Bohinju. Izračunamo najkrajšo pot po kateri dobavitelj pripelje platišča na dostavno mesto, preizkusimo pa še nekaj izmed drugih možnosti, ki jih programsko orodje omogoča.

10.3 Uporaba

V menijski vrstici *Graph* izberemo razdelek *Write Graph To Disk,* kjer narisan graf shranimo na poljubno mesto (glej Sliko 10.19).

Ob ponovnem odpiranju shranjenega dokumenta odpremo Petersen



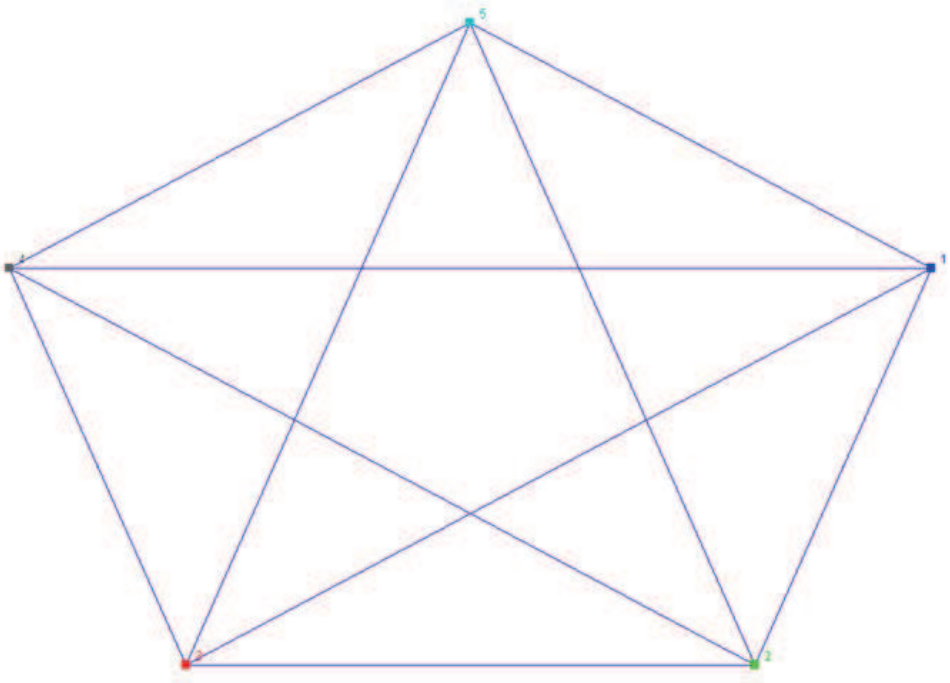
Slika 10.19: Shranjevanje datoteke

programsko orodje v Javi in znotraj njega izberemo možnost *(Read Graph From Disk)*. Za prikaz problema izbranim točkam določimo razdalje (uteži na povezavah). Določimo 5 točk, ki jih med sabo povežemo. Risanje grafa pričnemo postopoma, tako da najprej zapišemo in povežemo 5 točk. Te narišemo tako, da v razdelku *Named Graph – Null Graph* izberemo možnost 5 točk. Nato jih med sabo, z ukazom v menijski vrstici *Complete* razdelek *Complete Graph* povežemo (ponovno izberemo možnost 5) (glej Sliko 10.20).

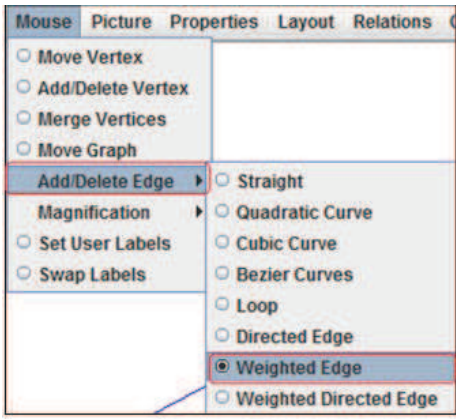
Razdelek *Mouse* omogoča, da z miško spreminjamo točke in povezave, jih dodajamo ali brišemo *(Move Vertex, Add/Delete Vertex, Merge Vertices)*. Podana je možnost premikanja grafa *(Mode Graph)*, prav tako pa lahko grafom dodajamo določene uteži. Kadar želimo uveljaviti spremembe na grafu, enostavno kliknemo na želeno možnost, npr. dodajanje uteži *(Weighted Edge)* (glej Sliko 10.21).

Najprej dodamo utež med točkama 4 in 5. Z miško se postavimo na točko 4 in se premikamo do točke 5. Program nas vpraša po vrednosti uteži, ki jo vpišemo v prazno okence. V danem primeru točki 4 in 5 pomenita vozlišče med dvema krajema z razdaljo 4 kilometrov. Enak postopek izvedemo še za določitev ostalih povezav (glej Sliko 10.22).

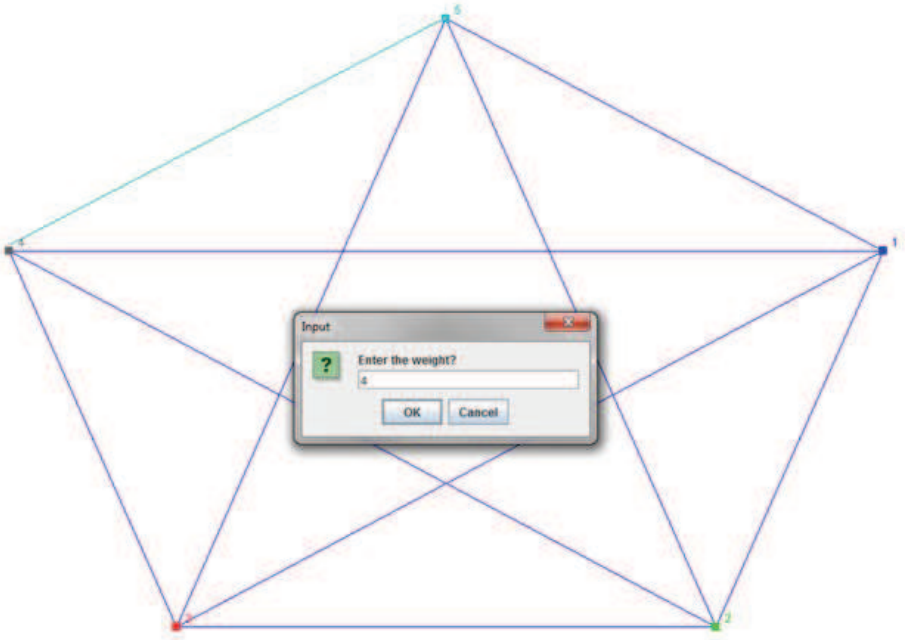
V primeru omejitev izberemo drugo vrsto uteži - *Weighted Directed Edge*.



Slika 10.20: Izris grafa s petimi točkami in povezavami



Slika 10.21: Dodajanje uteži



Slika 10.22: Zapis uteži

Postopek je popolnoma enak, kakor je predhodno zapisano, le da so povezave zapisane od ene do druge točke. Če želimo, da bo povezava obojestranska se z miško postavimo iz ene na drugo točko (npr. 3 →4) in nato še v obratni smeri (npr. 4 →3). V izbranem problemu uporabimo obojestranske povezave.

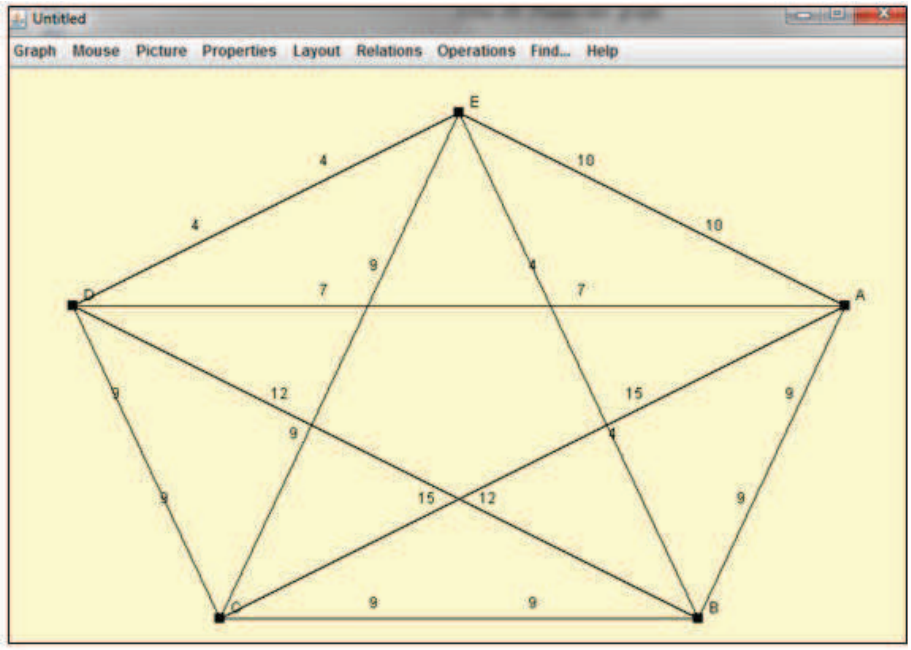
Slika 10.23 prikazuje vrednosti dvosmernih povezav. Da ne bi prišlo do zapletov, vozlišča označimo s črkami, saj so črke bolje razvidne.

Ob zapisu povezav in razdalj med posameznimi točkami izbiramo med možnostmi, ki jih ponuja programsko orodje. Razdelek *Change title* omogoča spreminjanje naslova dokumenta. Razdelek *Background Color* omogoča spreminjanje ozadja grafa - na voljo imamo najrazličnejše barvne sheme *(Color Sheme)* in linije *(Edges)* (glej Sliko 10.24).

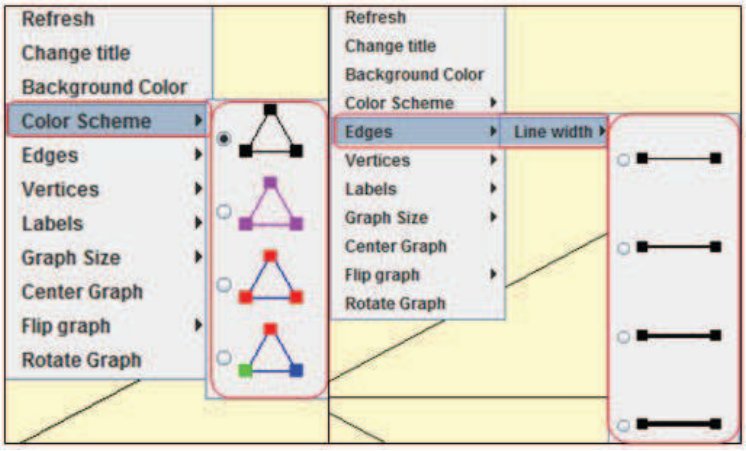
V razdelku *Labels* določimo oznake vozlišč: brez oznak *(No labels)*, s številkami *(Numbers)*, s črkami *(Letters)* ali z že uporabljenimi oznakami *(User's Labels)*. Menijska vrstica *Graph Size* ponuja možnost prikaza grafa (glej Sliko 10.25).

Razdelek *Flip graph* ponuja možnost horizontalnega, vertikalnega in diagonalnega premika. V razdelku *Picture* se nahaja razdelek *Rotate Graph*, ki omogoča poljubno rotacijo grafa (glej Sliko 10.26).

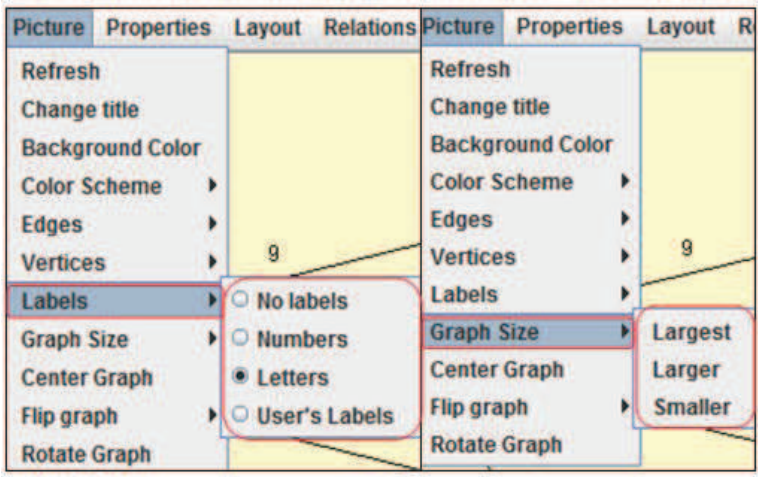
V menijski vrstici se nahaja še meni *Properties*. Razdelek *Statistics* prikazuje



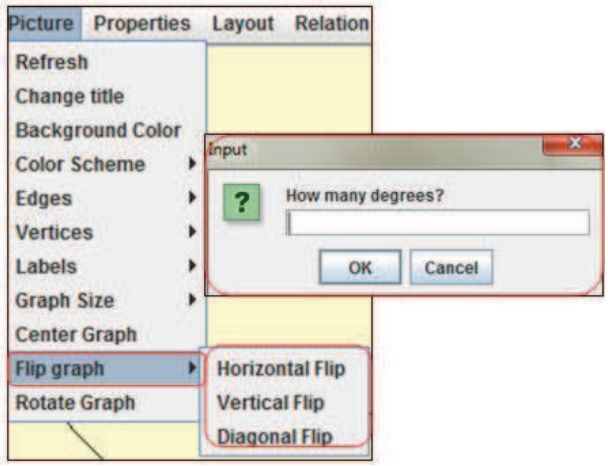
Slika 10.23: Vrednosti povezav



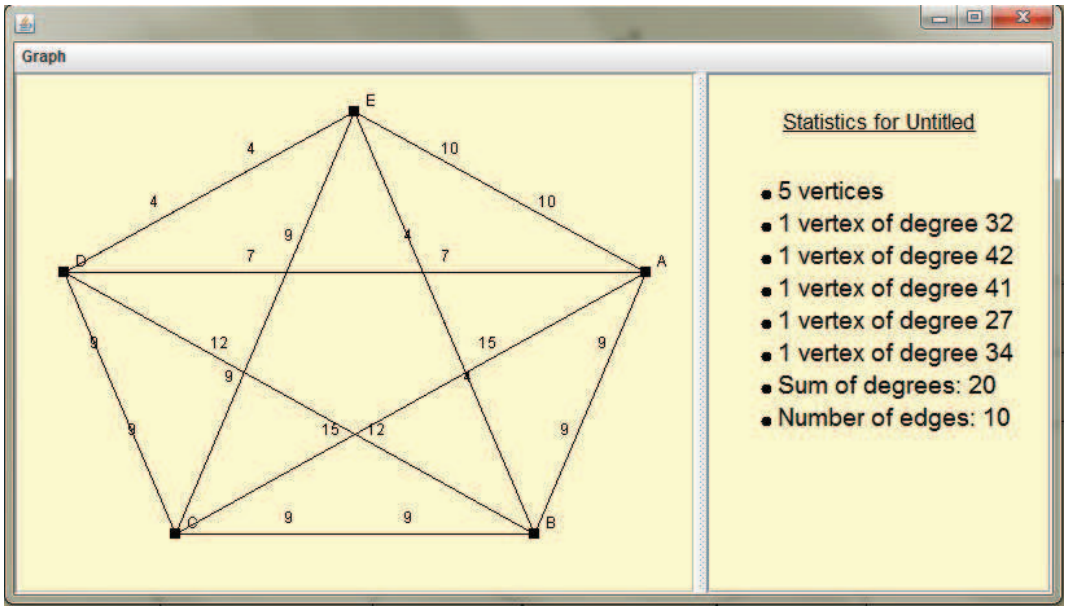
Slika 10.24: Možnosti grafa – 1. del



Slika 10.25: Možnosti grafa – 2. del



Slika 10.26: Možnosti grafa – 3. del



Slika 10.27: Properties- Statistics

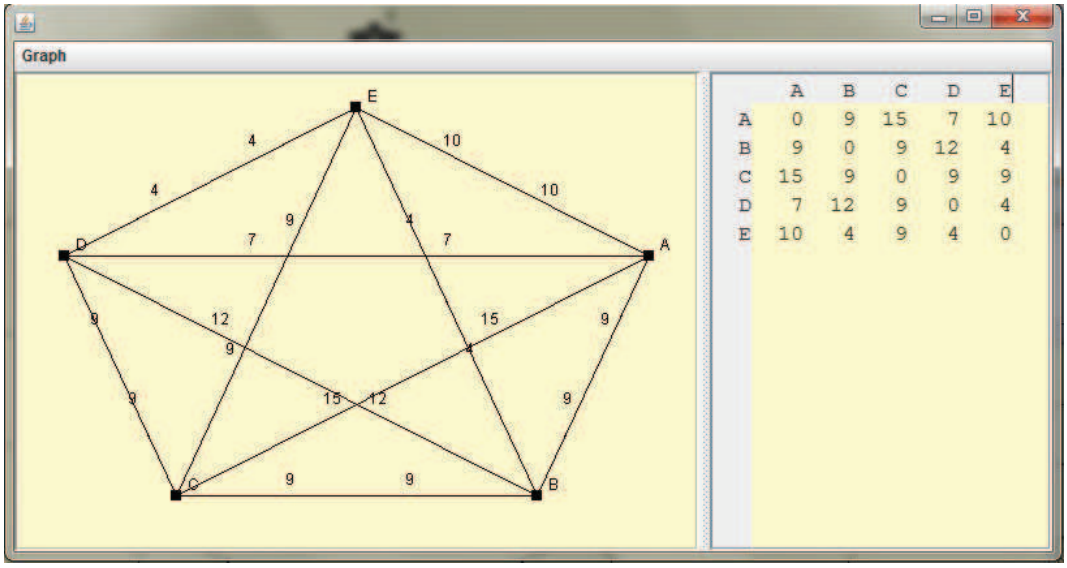
statistične podatke, ki se nanašajo na dani transportni problem (glej Sliko 10.27).

Sledi prikaz matrike sosednosti glede na razdalje. S klikom na dano možnost *(Adjacency Matrix)* se izriše tabela, prikazana na Sliki 10.28.

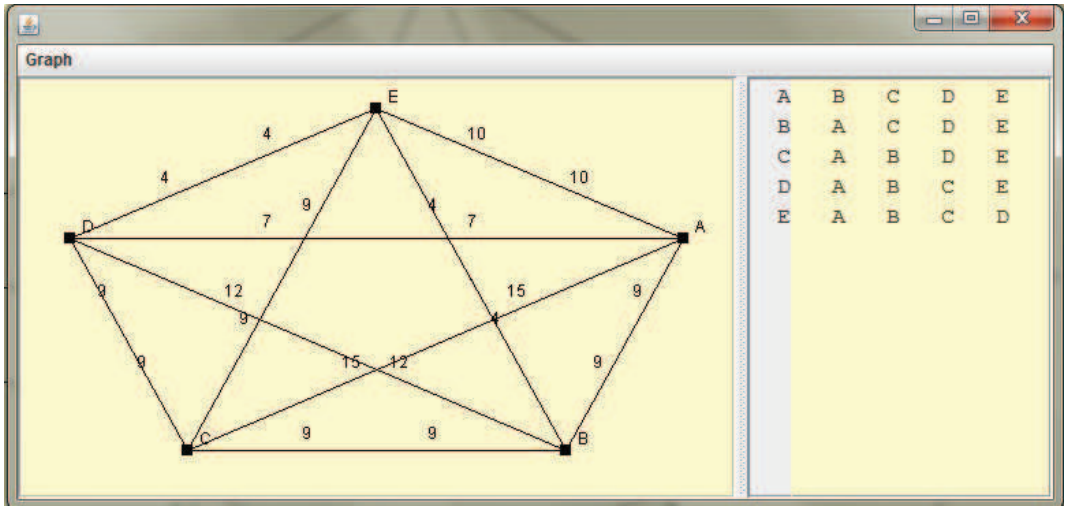
Slika 10.29 prikazuje izpis matrike sosednosti glede na povezave *(Adjacency List)*.

Z danim grafom ustvarjamo in izpišemo rezultate, glede na številne možnosti, ki jih ponuja programsko orodje. Ena izmed njih je izpis kromatičnega števila *(Chromatic Number)* (glej Sliko 10.30).

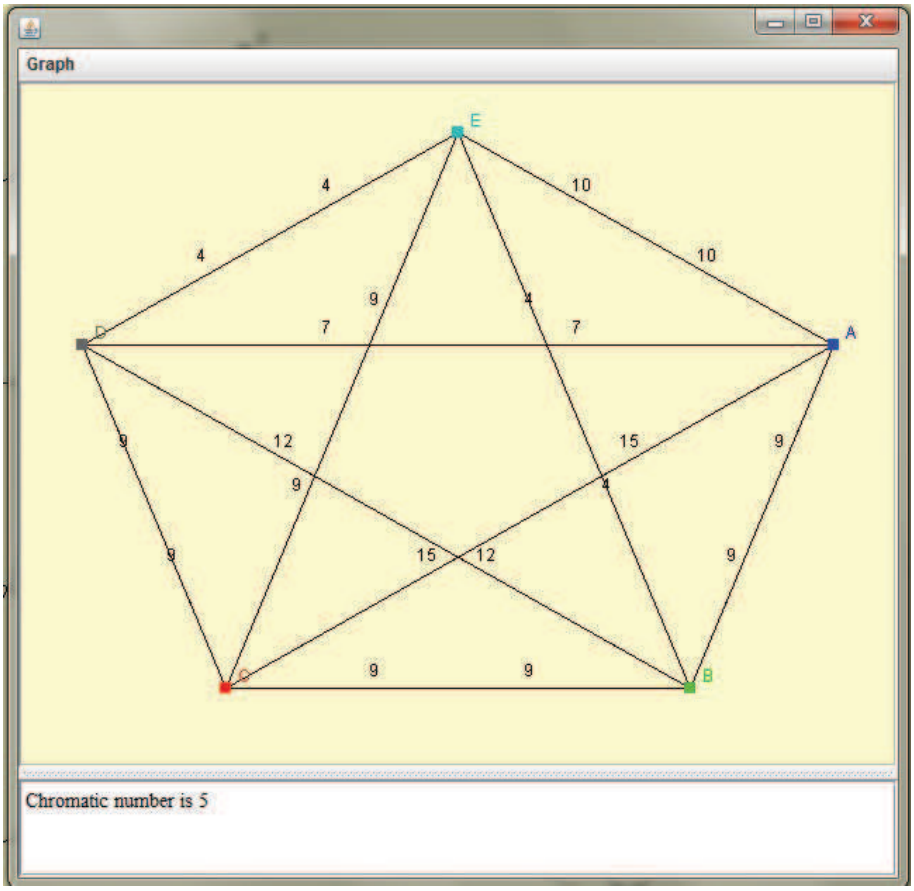
Seveda programsko orodje ponuja še številne druge možnosti, izhajajoče iz teorije grafov, npr.: Hamiltonov cikel, Eulerjev graf. Hamiltonova pot je po teoriji grafov pot v neusmerjenem grafu, ki gre skozi vsako točko na grafu natanko enkrat. Če sta začetna in končna točka poti enaki, jo imenujemo Hamiltonov cikel. Ime je dobila po irskem matematiku William Rowan Hamilton. V danem primeru se le ta ne izriše saj je graf usmerjen. Teorija Eulerjevega grafa temelji na dejstvu, da gremo po vsaki točki natanko enkrat in zaključimo na isti točki. Obhod je torej Eulerjev, če vsebuje vsako povezavo grafa natanko enkrat (in se zaključi v začetni točki).



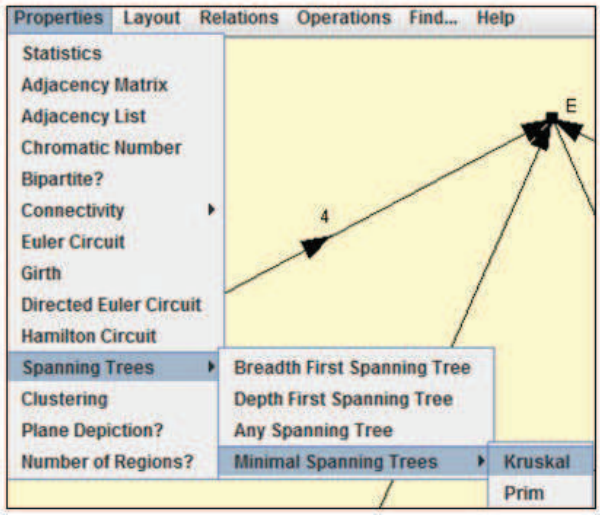
Slika 10.28: Matrika sosednosti glede na vrednosti



Slika 10.29: Matrika sosednosti glede na povezave



Slika 10.30: Kromatično število



Slika 10.31: Minimalno vpeto drevo

Iskanje najkrajše poti

V nadaljevanju se osredotočimo na problem iskanja najkrajše poti. V praksi poznamo dva algoritma, ki jih podrobneje opišemo v samem uvodu. Problem voznika, ki v podjetje dostavlja platišča je lahko povsem preprost. Prevoz opravi s tovorim vozilom, pri čemer upošteva razdalje med lokacijami.

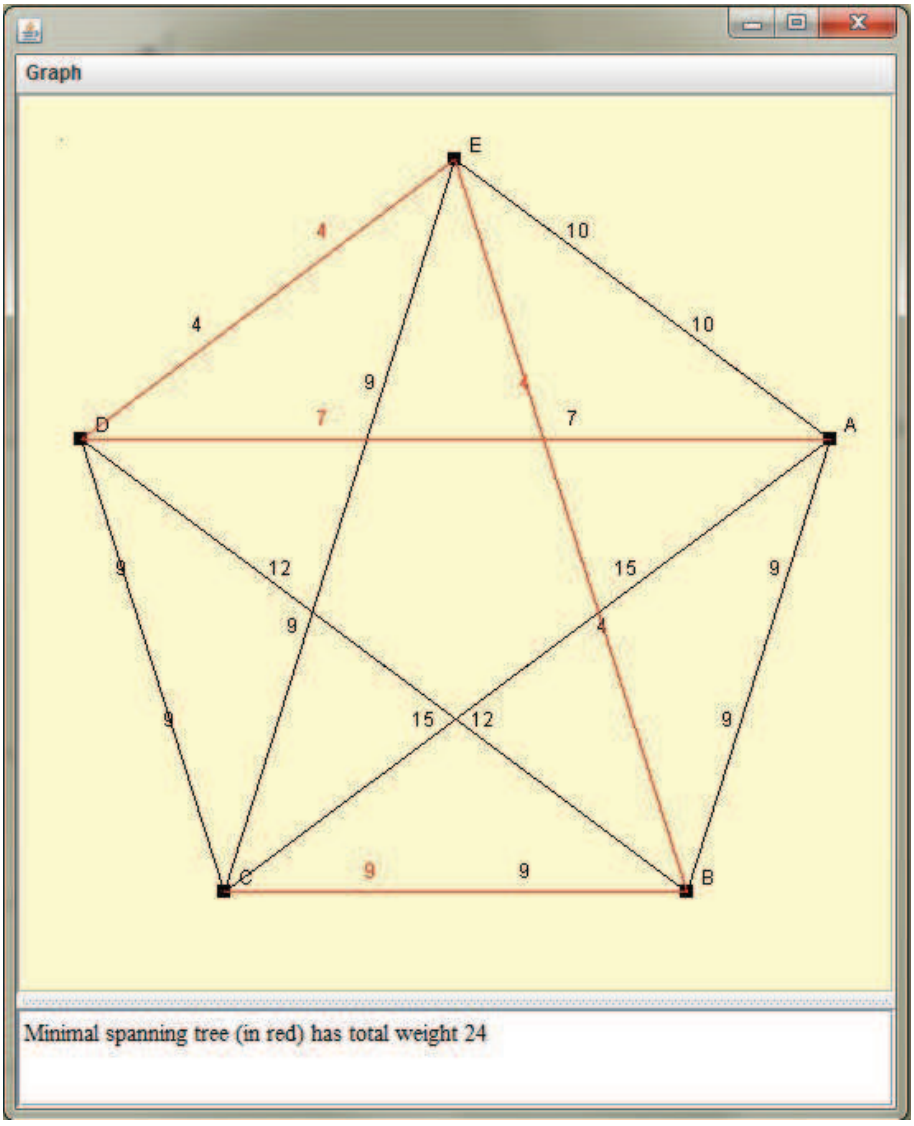
V menijski vrstici *Properties* izberemo razdelek *Spanning Trees*, v nadaljevanju pa *Minimal Spanning Trees*. Odločimo se med dvema možnostima - reševanje s Kruskalovim ali Primovim algoritmom (glej Sliko 10.31).

Na Sliki 10.32 prikazujemo izračun minimalnega vpetega drevesa s Kruskalovim algoritmom. Programsko orodje glede na izbrane lokaciije prikaže, da je minimalno vpeto drevo oz. minimalno število km, ki jih voznik tovornjaka opravi 24. Minimalno vpeto drevo lahko poiščemo tudi s Primovim algoritmom, ki v danem primeru izriše enako pot. Običajno poti nista enaki. V danem primeru imamo na voljo manjše število povezav, kar pomeni, da je možnosti drugih poti bistveno manjša.

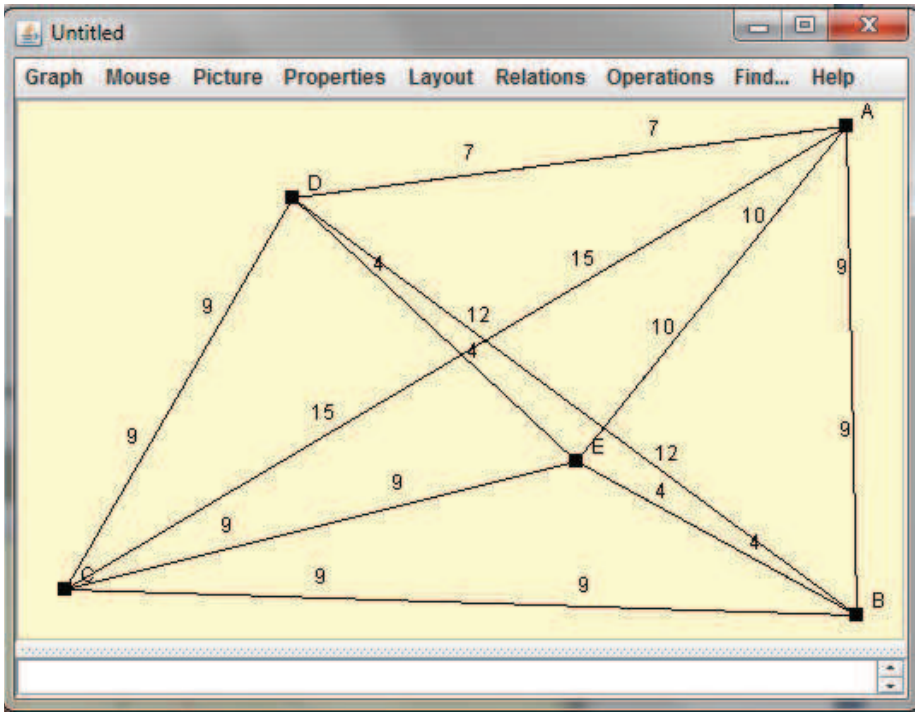
Druge možnosti programa

Menijska vrstica vsebuje še meni različnih postavitev programa *(Layout)*, kar je prikazano na Sliki 10.33.

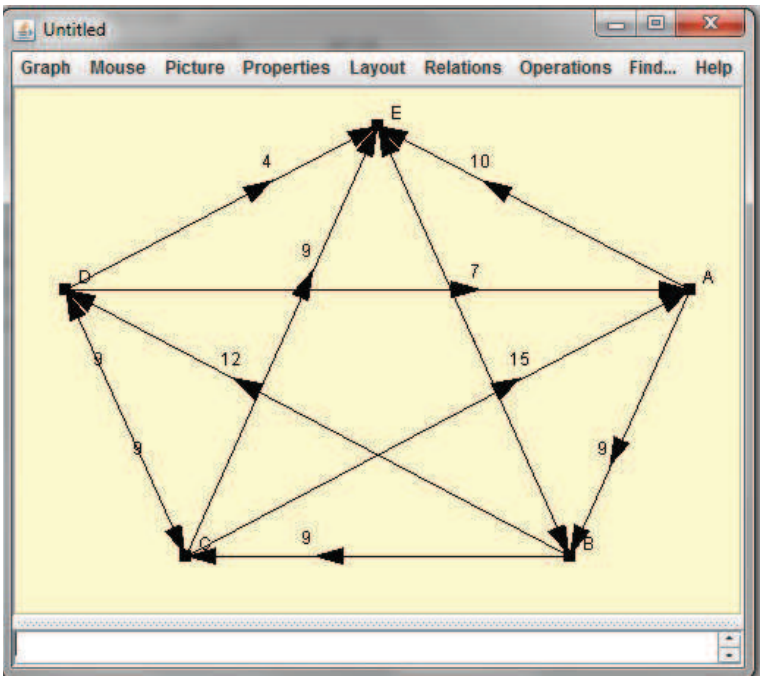
Menijska vrstica *Relations* prikaže razmerja med grafi. Če imamo več grafov (npr. 2), lahko ta med sabo primerjamo. Pogledamo ali sta



Slika 10.32: Minimalno vpeto drevo – Kruskalov algoritem



Slika 10.33: Postavitev grafa



Slika 10.34: Graf z omejitvami

izomorfna *(Isomorphism)*) in ugotovimo ali sta subgrafa *(Subgraph)*. Menijska vrstica *Operation* omogoča možnost prikaza komplementa narisanega grafa *(Complement)*), linijskega grafa med dvema točkama *(Line Graph)*, prizme *(Prism)*) ipd.

Menijska vrstica *Find* z razdelkom *Search* omogoča pregled grafa v globino in širino. Pomen iskanja točk na takšen način bo zagotovo znan znanstveniku, ki se podrobneje ukvarja s teorijo grafov. Pregled grafa v širino *(Breadth First)* je algoritem, ki začne v izbrani točki, katero pregleda. Vse "sosede" da v FIFO podatkovno strukturo in si zapomni h komu spada posamezna točka. Nato vzame prvega iz FIFO strukture in ponovi postopek. Postopek se konča, ko je FIFO struktura prazna. Pregled grafa v globino *(Depth First)* je algoritem, ki začne v izbrani točki in jo pregleda. Prav tako da vse sosede v LIFO podatkovno strukturo. Zapomni si h komu spada posamezna točka. Vzame prvega iz LIFO strukture in ponovi postopek. Konča, ko je LIFO struktura prazna. Obstaja še možnost iskanja najkrajše poti na podlagi Dijkstrovega algoritma *(Shortest Path → Dijkstra)*. V danem algoritmu iščemo najkrajšo pot od V1 do vseh ostalih točk. V izbranem primeru ni razlik v določitvi poti. Če bi imeli postavljen model z omejitvami, bi lahko le te tudi upoštevali (glej Sliko 10.34).

Povzetek

Petersen programsko orodje je učinkovito, praktično, enostavno in brezplačno s pomočjo katerega izdelujemo, urejamo in manipuliramo z enostavni grafi in preučujemo njihove lastnosti. Njegovo uporabo priporočamo vsem, ki se ukvarjajo z danim področjem. Z njegovo pomočjo na enostaven in hiter način preverimo rezultate, ki smo jih pridobili z analitičnim izračunom.
Prikažemo podatke o grafu, npr. število točk in njihove stopnje, matriko sosednosti, število sestavnih delov; preverimo, če je graf dvostranski, ali sta dva grafa izomorfna, če je graf podgraf drugega, poiščemo najkrajše poti ipd. Program omogoča pridobitev risbe grafov, do katerih je drugače skoraj nemogoče priti, uporaba animacije pa omogoča enostaven prikaz nekaterih pojmov.
Z izbranim programskim orodjem na podlagi teorije grafov prikažemo izračun optimalne transportne poti iz lokacije A do lokacije E in nekaj drugih možnosti, ki jih ponuja programsko orodje.

Pri opisu programskega orodja Petersen smo uporabili še dodatne vire in literaturo: [92].