PRIRODOSLOVNO-MATEMATIČKI FAKULTET

SVEUČILIŠTE U SPLITU

Ana Marjanović

PROBLEM SEDAM MOSTOVA KOENINGSBERGA

ZAVRŠNI RAD

Split, listopad 2012.

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

SVEUČILIŠTE U SPLITU

ZAVRŠNI RAD

PROBELM SEDAM MOSTOVA KOENINGSBERGA

MENTOR:

Dr. sc. Slavomir Stankov

STUDNET:

Ana Marjanović

SMJER:

Matematika-informatika

NEPOSREDNI VODITELJ:

Dr. sc. Ani Grubišić

Split, listopad 2012.

SADRŽAJ

1. Uvod

U ovom završnom radu obradit ću temu pod nazivom „Problem sedam mostova Koeningsberga“.

Problem sedam mostova usko je vezan sa granom matematike koja se zove teorija grafova pa su stoga osnovne definicije i teoremi iz tog područja iskazani i dokazani u drugom poglavlju. U trećem pogavlju opisan je problem sedam mostova te je prikazana povijest grada Koeningsberga. Jednako tako dan je kratak životopis Leonharda Eulera koji je postavio teoriju grafova i predstavio rješenje problema sedam mostova.

Budući sam programe koji rješavaju problem sedam mostova napisala u programskoj jeziku C, u četvrtom su poglavlju objašnjena neka njegova svojstva te sama struktura programskog jezika. U petom je poglavlju objašnjena struktura pod nazivom polja podatka jer je to struktura na kojoj se temelje programi koji rješavaju dani problem.

U šestom je poglavlju dano Eulerovo rješenje problema te objašnjeno rješenje problema pomoću matrice susjedstva. Jednako tako dano je poopćenje problema i opisan Fleuryjev algoritam.

2. Osnovni pojmovi teorije grafova

Grafovi su jedna od osnovnih matematičkih struktura. Stoga se i pojavljuju u raznim oblicima i raznim situacijama. Mnoge se pojave modeliraju grafovima koji se sastoje od točaka i njihovih spojnica. Na primjer, točke (vrhovi ili čvorovi) mogu predstavljati ljude iz neke skupine, a spojnice (bridovi) parove prijatelja. Graf može predstavljati električnu mrežu čiji su vrhovi električke komponente, a spojnice električne veze. Cestovne, željezničke, zrakoplovne veze itd. daljnji su primjeri modela s grafovima. U računarstvu se često dijagram toka nekog algoritma prikazuje grafom kojem su čvorovi naredbe (instrukcije), a lukovi iz jedne u drugu naredbu su bridovi. Jednako tako grafovima se prezentiraju i razne kompjuterske strukture podataka, umrežavanje i paralelizam računala i njihov sekvencijalni rad, evolucijska ili porodična stabla u biologiji itd.

2.1. Definicija grafa

Definicija 2.1. Graf je uređeni par $G = (V, E)$, gdje je $\emptyset \neq V = V(G)$ skup vrhova, $E = E(G)$ skup bridova disjunktnih s V. Svaki brid $e \in E$ spaja dva vrha $u, v \in V$ koji se zovu krajevi od e. Kažemo još da su vrhovi u i v incidentni s e, a vrhovi u i v susjedni i pišemo $e = \{u, v\}$. Graf sa samo jednim vrhom zove se **trivijalan**, inače je netrivijalan. **Usmjereni graf** ili **digraf** D je graf G u kojem svaki brid ima smjer od početka prema kraju.

Definicija 2.2. Graf je uređene trojka $G = (V, E, \varphi)$ gdje je

$$\varphi: E \to \binom{V}{2}$$

funkcija koja svakom bridu *e* pridružuje 2-člani multiskup vrhova $\varphi(e) = \{u, v\}$ koji se zovu krajevi od *e*.

Definicija 2.3. Brid čiji se krajevi podudaraju zove se **petlja**, a ako su krajevi različiti – **pravi brid** ili **karika**.

Definicija 2.4. Graf je jednostavan ako nema petlji ni višestrukih bridova.

Defincija 2.5. Jednostavan graf u kojem je svaki par vrhova spojen bridom naziva se potpun graf.

2.2. Šetnje, matrica incidencije i matrica susjedstva

Definicija 2.6. Šetnja u grafu G je niz $W := v_0e_1v_1e_2 \cdots e_kv_k$, čiji su članovi naizmjence vrhovi v_i i bridovi e_i, tako da su krajevi od e_i vrhovi v_{i-1} i v_i, $1 \leq i \leq k$.

U jednostavnom je grafu štenja potpuno određena samo nizom svojih vrhova $v_0v_1 \cdots v_k$. Kažemo da je v_0 početak, a v_k kraj šetnje W ili da je W šetnja od v_0 do v_k ili (v_0, v_k)-šetnja.

Definicija 2.7. Neka je G graf s vrhovima $v_1, v_2, \cdots, v_n$ u nekom poretku i bridovima $e_1, e_2, \cdots, e_m$ u nekom poretku. Matrica incidencije grafa G je (pravokutna) $n \times m$-matrica $M = M(G) = [m_{ij}]$, gdje je m_{ij} = broj (0, 1 ili 2) koliko su puta v_i i e_j incidentni. Matrica incidencije potpuno određuje graf. Matrica susjedstva grafa G je (kvadratna) $n \times n$-matrica $A = A(G) = [a_{ij}]$, gdje je a_{ij} = broj bridova koji spajaju v_i i v_j. A je simetrična matrica (tj. $a_{ij} = a_{ji}$) čiji su članovi nenegativni cijeli brojevi. Ako je graf jednostavan, onda je A (0,1)-matrica, tj. $a_{ij} = 0 \; ili \; 1, \forall i, j$, a na glavnoj dijagonali su nule. I obrnuto, svaka takva matrica reprezentira neki graf. A je simetrična, pa ima n realnih svojstvenih vrijednosti $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ čija je suma $TrA = 0$ (ako je G jednostavan).

2.3. Eulerova tura i Eulerova staza

Definicija 2.8. Eulerova tura na grafu je zatvorena staza koja sadži svaki brid. Dakle, to je zatvorena staza koja prolazi svakim bridom točno jedanput. Graf je Eulerov ako dopušta Eulerovu turu.

Teorem 2.1. Povezani graf G je Eulerov ako i samo ako mu je svaki vrh parnog stupnja.

Dokaz: $\Rightarrow$: Neka je C Eulerova tura na G. Svaki put kada C „uđe“ u vrh v mora i izaći iz v. Točnije, ako fiksiramo jednu orijentaciju obilaska Eulerovom turom C, onda se skup bridova incidentnih sa v može particionirati u dva skupa: ulazeći i izlazeći iz v i među njima postoji očita bijekcija.

$\Leftarrow$: Glavni trik je da pogledamo najdužu moguću stazu u G i dokažemo da je Eulerova tura. Neka je, dakle, $T = (v_0, e_1, v_1, \cdots, e_m, v_m)$ najduža staza duljine m. Dovoljno je dokazati da je

(i) $v_0 = v_m$

(ii) $E(G) = \{e_1, e_2, \cdots, e_m\}$

Pretpostavimo da je $v_m \neq v_0$. Tada je v_0 incidentan s neparnim brojem bridova iz T. No, kako je $d_G(v_0)$ paran, postoji brid $e \in E(G)\backslash E(T)$ incidentan s v_0 pa bi T mogli produžiti tim bridom što je u kontradikciji s izborom T. Time smo dokazali (i).

Dokažimo (ii). Uzmimo prvo da je $V(T) \neq V(G)$ i neka je $u \in V(G)\backslash V(T)$. Kako je G povezan odmah slijedi da postoji brid $e = vv_k, v \notin V(T), v_k \in V(T)$. Tada imamo stazu

$$(v, e, v_k, e_{k+1}, v_{k+1}, \cdots, v_{m-1}, v_0, e_1, v_1, \cdots, e_k, v_k)$$

duljine $m + 1$, što je opet kontradikcija s izborom T.

Ako je $V(T) = V(G)$, a $E(T) \neq E(G)$, neka je $e \in E(G)\backslash E(T)$. Tada je e oblika $e = v_k v_l$ za neke k, l. Slično prethodnom slučaju tada imamo stazu

$$(v_k, e_{k+1}, v_{k+1}, \cdots, v_{m-1}, e_m, v_0, e_1, v_1, \cdots, e_k, v_k, e, v_l)$$

duljine $m + 1$ i opet kontradikciju. ■

Definicija 2.9. Eulerova staza je staza koja prolazi svakim bridom grafa točno jedanput (a ne mora biti zatvorena).

Korolar 2.1. Povezan graf G ima Eulerovu stazu ako i samo ako ima najviše dva vrha neparnog stupnja.

Dokaz: Ako G ima Eulerovu stazu onda kao u dokazu *Teorema 2.1.* svaki vrh koji nije početak ni kraj te staze ima parni stupanj. Ako G ima najviše dva vrha neparnog stupnja, onda ih ima 0, 1 ili 2. Ako ih ima 0, tj. svi su parnog stupnja, onda prema *Teoremu 2.1.* G ima zatvorenu Eulerovu stazu. Kako je $\sum d(v) = 2e(G)$, ne može biti samo jedan vrh neparnog stupnja, pa ako ih uopće ima, mora ih biti točno dva, recimo u i v. Neka je $e = uv$ novi brid i promotrimo graf $G + e$. U tom su grafu svi vrhovi parnog stupnja, pa prema *Teoremu 2.1.* ima Eulerovu turu T. Tada je $T - e$ Eulerova staza u G. ■

Upravo ovaj Korolar povlači da graf mostova u Koeningsbergu nema Eulerovu stazu, odnosno da problem sedam mostova nema rješenje.

3. Opis problema

Čovjek se svakodnevno susreće s problemima traženja puta od početne točke A do odredišta B. Svaki takav problem svodi se na rješavanje problema korištenjem mrežnog modela. Jedan od najpoznatijih problema je upravo Problem sedam mostova Koeningsberga za kojeg je 1735. godine švicarski znanstvenik Leonhard Euler donio negativan zaključak.

3.1. Povijest Koeningsberga

Koeningsberg je bio glavni grad Prusije od kasnih godina srednjeg doba pa sve do 1701. godine kada je glavni grad postao Berlin. Izgrađen je s obje strane rijeke Pregel i na dva riječna otoka. Mostovi povezuju obje strane rijeke i otoke.

Slika 1 Sedam mostova Koeningsberga

Ime Koeningsberg doslovno znači kraljeva planina. Grad je teško oštećen tijekom Savezničkog bombardiranja 1944. godine te je osvojen od strane Crvene armije nakon Bitke kod Koeningsberga. Pod vlašću sovjetskog vođe Mikhaila Kalinina preimenovan je u Kalingrad. Danas je glavni grad Ruske Kaliningradske oblasti.

3.2. Tko je bio Leonhard Euler?

Leonhard Euler (1707. – 1783.) je bio švicarski matematičar, fizičar i astronom. Svoju znanstvenu djelatnost razvio je u Brlinu i Petrogradu gdje je držao katedru fizike i matematike. Njegova aktivnost nije stala ni kada je oslijepio jer je tada diktirao radove. Napisao je oko 900 radova. Razvio je teoriju redova, uveo tzv. Eulerove integrale, rješio mnoge diferencijalne jednadžbe, a u diferencijalnoj geometriji dao je prvu formulu

zakrivljenosti ploha (Eulerova poučak). Razvio je teoriju turbina te proučavao širenje zvuka i svjetlosti. Euler je najproduktivniji matematičar u povijesti. Nakon njegove smrti, Sanktpeterburška je akademija još punih 50 godina tiskala njegove neobjavljene radove.

Euler se rodio u švicarskom gradu Baselu. Otac mu je bio pastor u protestantskoj crkvi, a majka je potjecala iz svećeničke obitelji. Imao je dvije mlađe sestre, a jedan od bliskih obiteljskih prijatelja bio je, tada već poznati matematičar, Johann Bernoulli. Već u ranoj dobi pokazao je zanimanje za matematiku i uočio rupe u svom znanju pa je zamolio Bernoullija za redovitu privatnu poduku. Iskusni Bernoulli brzo je uočio Eulerovu izuzetnu nadarenost te nazreo njegov silni znanstveni potencijal. Kad je Euler počeo studirati teologiju, učiti grčki, latinski i hebrejski jezik, njegov se otac ponadao da će sin poći njgeovim stopama. Stoga Bernoulliju nije bilo nimalo lako uvjeriri oca kako mu je sin sudbinski predodređen postati velikim matematičarom. 1727. godine Euler je završio svoju doktorsku disertaciju o širenju zvuka, a iste mu je godine Pariška akademija dodijelila drugu nagradu za rješavanje problema o optimalnom smještanju jarbola na jedrenjak.

1727. Euler je dobio posao na Ruskoj carskoj akademiji znanosti u St. Petersburgu. 1731. posato je profesor fizike i u to je vrijeme napisao knjigu o teoriji glazbe te djelo *Scientia navalis* u kojoj izlaže znanja iz hidrodinamike, gradnje brodova i navigacije. Dvije godine kasnije postao je voditelj matematičkog odjela.

Zbog učestalih nemira u Rusiji, 1741. sa obitelji se preselio u Berlin gdje je postao predvodnikom matematičkog odjela na Sveučilištu. U Berlinu je proveo sljedećih 25 godina i u tom je periodu napisao preko 380 znanstvenih članaka i objavio svoja dva velika djela: *Introductio in analysis infinitorum* i *Institutiones calculi differentialis*. Bio je član gotovo svih značajnih akademija u Europi i dobitnik brojnih priznanja i nagrada.

1738. oslijepio je na desno oko. Sljepoća je bila posljedica trovanja zbog gnojnog čira. Usprkos svemu nastavio je raditi s jednakim žarom, posvetio se izradi atlasa (izradio je pomorsku kartu Rusije) i zbirci karata za Sanktpeterburšku akademiju. Nedugo nakon toga siva mrena prekrila mu je i lijevo oko te je bio potpuno slijep. Gotovo pola svog znanstvenog opusa stvorio je nakon gubitka vid. 1783. brzo i bezbolno umro je od izljeva krvi u mozak.

4. Programski jezik C

Odlučila sam se za pisanje programa u programskom jeziku C jer je to jezik opće namjene, što znači da se u njemu može napisati apsolutno sve: od rješavanja zadataka, do pisanja drivera, operacijskih sustava, tekst procesora ili igara. Kao jezik, ni u čemu ne ograničava. Omogućuje i uključivanje naredbi pisanih asemblerski zbog čega je zajedno s mogućnosšću direktnog pristupa pojedinim bitovima, bajtovima ili cijelim blokovima memorije, pogodan za pisanje sistemskog softvera. C je među popularnijim programskim jezicima i rabe ga mnogi programeri. Programi pisani u C-u su prenosivi (mogu se prevoditi i izvršavati na različitim porodicama računala uz minimalne ili nikakve ispravke) i obično su vrlo brzi. Postoje mnogi prevoditelji za jezik C, a jedan od najšire korištenih je *GNU C Compiler*.

4.1. Svojstva

Programski jezik C spada u proceduralne programske jezike. Razvijen je u 70-im godina 20. stoljeća. Autor ovog programskog jezika je Dennis Ritchie – stvorio je programski jezik za rješavanje praktičnih problema kodiranja sistemskih programa i jezgre operacijskog sustava UNIX, koji je praktički u cijelosti napisan u C-u.

C se tijekom godina dosta mijenjao te je u više navrata neformalno i formalno standardiziran. Kao jedan od najvažnijih jezika u povijesti komercijalne računalne industrije, C je do danas ostao jedini programski jezik prilagođen za sve računalne platforme, od malih sustava pa do mrežnih superračunala. Programi napisani u njemu vrlo su bliski načinu rada hardvera te u načelu zahtijevaju od programera dobro razumijevanje rada procesora, memorije, ulazno-izlaznih sklopa itd. No, rad s registrima procesora i adresiranje memorije apstrahirani su pomoću koncepta varijabli i pokazivača što uz eksplicitne kontrolne strukture i funkcije znatno olakšava programiranje u odnosu na izravno programiranje u strojnim jezicima.

Tokom 1980-ih, Bjarne Stroustrup zajedno s drugim istaživačima proširuje C dodavajući sposobnosti objektno orijentiranog programiranja, a naziv ovog novog programskog jezika je C++. Nažalost, ta 100%-tna kompatibilnost ujedno je i razlog što su problemi koje programiranje u C-u nosi sa sobom naslijeđeni i u C++-u. Efikasno i sigurno programiranje u C-u vrlo je zahtjevna vještina koja traži višegodišnje iskustvo pa je stoga C jezik koji se ne preporučuje početnicima, posebice ako im programiranje nije primarni posao.

Danas se relativno rijetko ukazuje potreba za pisanjem novih korisničkih aplikacija izravno u C-u, pa čak i u vrlo malim sustavima kao što su primjerice mobilni telefoni. Glavno područje njegove upotrebe su sistemski programi na strani poslužitelja, programi prevoditelji i jezgre operativnih sustava, gdje je potreba za najvećom mogućom brzinom izvođenja, efikasnom kontrolom resursa i izravnom kontrolom hardvera od primarne važnosti.

4.2. Struktura programa pisanog u C-u

Najjednostavniji program je onaj koji zadani tekst ispisuje na ekran. Da bi se mogao ispravno napisati čak i tako jednostavan program, potrebno je poznavati strukturu programa odabranog programskog jezika. Struktura programa odnosi se na način pisanja programa i ako se ona ne poštuje, program neće raditi, bez obzira na to što su upotrebljene sve potrebne naredbe.

Svaki program pisan u programskom jeziku C sastavljen je od niza funkcija. Glavna i jedina obavezna funkcija u programu je funkcija *main ()*. Program se može sastojati i od većeg broja funkcija.

Bitna razlika između C-a i ostalih programskih jezika je u tome što u C-u ne postoje ugrađene funkcije. One se nalaze u bibliotekama funkcija koje se isporučuju zajedno s prevoditeljem. Biblioteke funkcija nastale su standardizacijom C-a, pa je dovoljno na početku programa najaviti da će se upotrebljavati određena biblioteka i u cijelom programu dostupne su sve funkcije koje se u njoj nalaze. Tako se biblioteka u kojoj se nalaze funkcije za ulaz i izalz podataka naziva *stdio.h*, matematičke funkcije nalaze se u biblioteci *math.h*, a funkcije za rad sa znakovinim varijablama u biblioteci *string.h*.

Na početku svakog C programa uobičajeno je pisati pretprocesorske naredbe kojima se pozivaju biblioteke funkcija koje se koriste u programu. Najčešće upotrebljavana pretprocesorska naredba je *#include <ime datoteke>* i obavezno se piše prije funkcije *main ()*. Prevođenjem programa na mjesto naredbe *#include* kopira se sadržaj navedene datoteke.

Jednako tako, prije samo pisanja programa treba istaknuti da se programi napisani u C-u sastoje od niza međusobno povezanih funkcija čiji broj nije ograničen. U svakom programu je obavezna samo jedan funkcija (funkcija *main()*) koja označava mjesto na kojem počinje izvršavanje programa. Početak i kraj funkcije označava se vitičastim zagradama. Nakon svake naredbe u C-u mora stajati znak ; koji označava kraj jedne i početak druge naredbe.

5. Polje

Budući će se programi koji rješavaju problem sedam mostova temeljiti na strukutri koja se zove polje objasnimo kakva je to struktura.

5.1. Definicija polja

Često se u programima koriste skupovi istovjetnih podataka. Za označavanje i rukovanje takvim istovjetnim podacima bilo bi vrlo nespretno koristiti za svaki pojedini podatak zasebnu varijablu, primjerice x1, x2, x3 itd. Zamislimo samo da trebamo rezultate nekog mjerenja koji sadrže 100 podataka statistički obraditi, pri čemu svaki podatak treba proći kroz isti niz računskih operacija. Ako bismo koristili zasebna imena za svaki podatak, program bi se trebao sastojati od 100 istovjetnih blokova naredbi koji bi se razlikovali samo u imenu varijable koja se u dotičnom bloku obrađuje. Neefikasnost ovakvog postupka je više nego očita. Daleko efikasnije je sve podatke smjestit pod isto nazivlje, a pojedini rezultat mjerenja dohvaćati pomoću brojčanog indeksa. Ovakvu obradu podataka omogućavaju *polja (nizovi) podataka*.

Polje podataka je niz konačnog broja istovrsnih podataka – *članova polja*. Ti podaci mogu biti bilo kojeg tipa, ugrađenog (primjerice *float* ili *int*) ili korisnički definiranog. Pojedini članovi polja mogu se dohvaćati pomoću cjelobrojnih indeksa te mijenjati neovisno o ostalim članovima polja.

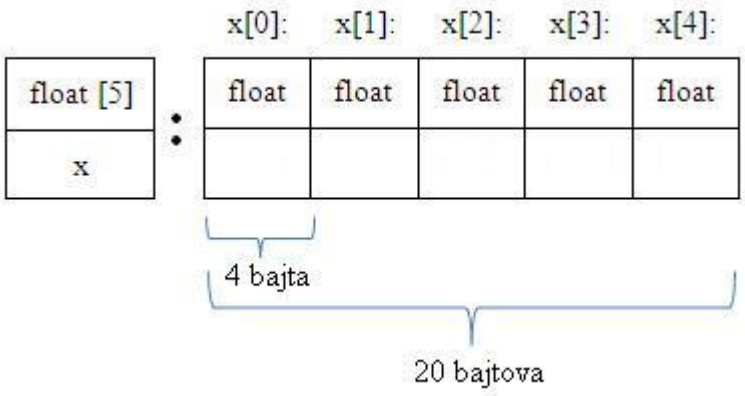
5.2. Jednodimenzionalna polja

Najjednostavniji oblik polja su jednodimenzionalna polja kod kojih se članovi dohvaćaju preko samo jednog indeksa. Članovi polja složeni su u linearnom slijedu, a indeks pojedinog člana odgovara njegovoj udaljenosti od početnog člana.

Želimo li deklarirati jednodimenzionalno polje x koje će sadržavati 5 decimalnih brojeva tipa *float,* napisat ćemo

float x[5];

Prevoditelj će ovom deklaracijom osigurati kontinuirani prostor u memoriji za pet podataka tipa *float* (Slika 2). Ovakvom deklaracijom članovi polja nisu inicijalizirani,

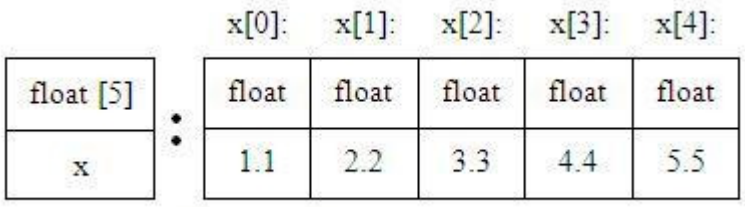


Slika 2Deklaracija polja

tako da imaju slučajne vrijednosti, ovisno o tome što se nalazilo u dijelu memorije koji je dodijeljen (alociran) polju. Članovi polja se mogu inicijalizirati prilikom deklaracije:

float [x] = {1.1, 2.2, 3.3, 4.4, 5.5};

Ovom naredbom se deklarira jednodimenzionalno polje x, te se članovima tog polja pridjeljuju početne vrijednosti. Vrijednosti se navode unutar vitičastih zagrada i odvajaju zarezima. Iako duljina polja nije eksplicitno navedena, prevoditelj će iz inicijalizacijske liste sam zaključiti da je polje duljine 2 i rezervirati odgovarajući memorijski prostor (Slika 3).



Slika 3 Jednodimenzionalno polje s inicijaliziranim članovima

Pri odabiru imena za polje treba voditi računa da se ime polja ne smije podudarati s imenom neke varijable u području dosega polja, tj. u području u kojem je polje vidljivo. Prvi član u polju ima indeks 0, a zadnji član ima indeks za jedan manji od duljine polja.

Jedno od ograničenja pri korištenju polja jest da duljina polja mora biti specificirana i poznata u trenutku prevođenja koda. Duljina jednom deklariranog polja se ne može mijenjati tijekom izvođenja programa.

5.3. Dvodimenzionalna polja

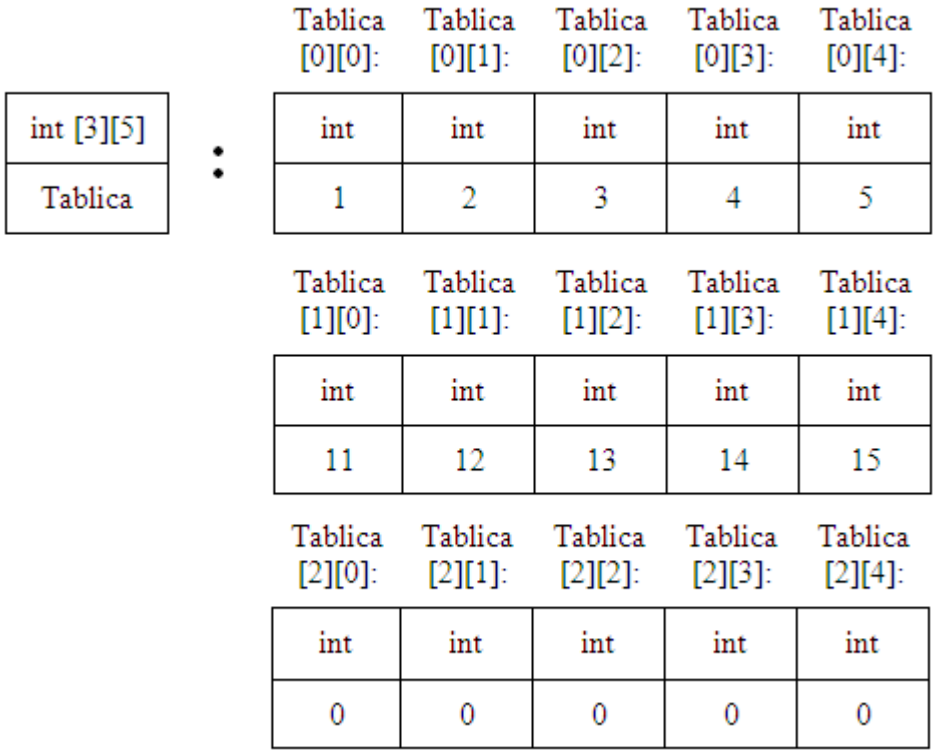
Često se javlja potreba za pohranjivanjem podataka u dvodimenzionalne ili višedimenzionalne strukture. Želimo li pohraniti podatke iz neke tablice s 3 retka i 5 stupaca, najprikladnije ih je pohraniti u dvodimenzionalno polje:

int Tablica [3][5];

Članove tog dvodimenzionalnog polja dohvaćamo preko dva indeksa: prvi je određen retkom, a drugi stupcem u kojem se podatak u tablici nalazi. U suštini se ovo dvodimenzionalno polje može shavtiti kao tri jednodimenzionalna polja od kojih je svako duljine 5 (Slika 4). Pravila koja vrijede za jednodimenzionlna polja vrijede i za višedimenzionalna polja. Članovi višedimenzionalnog polja mogu se također inicijalizirati prilikom deklaracije:

int Tablica [3][5] = {{1, 2, 3, 4, 5}, {11, 12, 13, 14, 15}};

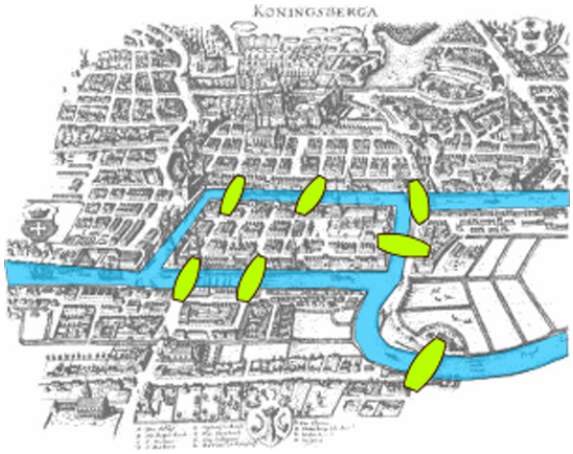
Ovime se inicijaliziraju prva dva retka dok se članovi trećeg retka inicijaliziraju na nulu (kako je prikazano slikom 4).



Slika 4 Prikaz dvodimenzionalnog polja

6. Rješenje problema

Na donjoj slici prikazan je raspored mostova. Četiri dijela grada (sjeverni, južni i dva otoka) međusobno su povezani mostovima. Manji je otok s po dva mosta povezan sa sjevernim i s južnim dijelom grada dok je veći povezan s po jednim mostom sa sjevernim i s južnim dijelom. Jedan je most povezivao oba otoka.

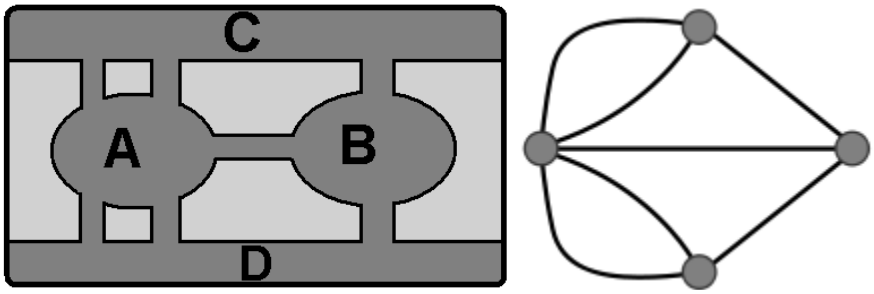


Slika 5 Raspored mostova

Problem Koeningsberga svodi se na pitanje je li moguće prijeći preko svih sedam mostova tako da se preko svakog mosta prijeđe točno jednom.

6.1. Eulerovo rješenje problema

Problem sedam mostova privukao je pozornost Leonhardu Euleru koji je 1735. godine predstavio rješenje problema. Euler je iznio pretpostavku da je izbor puta na kopnu nebitan. Jedino važno svojstvo svake rute je slijed prelazaka preko mostova. Ta mu je pretpostavka omogućila da ukloni sva svojstva, osim kopnenih površina i mostova koji ih povezuju te je na taj način stvorio apstraktni prikaz problema. U tom slučaju svaka kopnena površina (obale rijeke i otoci) prikazana je čvorom ili vrhom, a svaki most kao brid. Bridovi povezuju čvorove i tvore matematičku strukturu koja se naziva graf.



Slika 6 Grafički prikaz mostova

Osim krajnjih točaka šetnje, dolazak na čvor i odlazak mora biti preko mosta (brida). Dakle, tijekom šetnje po grafu, broj puta kada se dođe do čvora (a da čvor nije kraj puta) jednak je broju puta kada se izađe s tog čvora. To znači da broj mostova koji dodiruju kopno mora biti paran ukoliko most želimo prijeći samo jednom. Polovica tih mostova će biti prohodana kada se ide prema kopnu, a polovica kada se ide od kopna. Budući da su svi čvorovi spojeni s neparnim brojem bridova, nije moguće proći sve mostove samo jednom.

Euler je pokazao da šetnja po grafu u kojoj se svaki brid koristi samo jednom ovisi o stupnju čvorova. Stupanj čvora predstavlja broj bridova koji ga dotiču. Šetnja će biti moguća samo ako je graf spojen i ako nula ili dva čvora imaju neparan stupanj. Ovakva šetnja po grafu zove se Eulerova šetnja, a ostvariva je samo ako započinje na čvoru s neparnim stupnjem i završava na čvoru s neparnim stupnjem. Ovakvim načinom razmišljanja Euler je postavio temelje teorije grafova.

6.2. Rješenje problema pomoću matrice susjedstva

Problem sedam mostova Koeningsberga može se na vrlo jednostavan način riješiti pomoću matrice susjedstva.

Način rada programa opisan je pseudokodom:

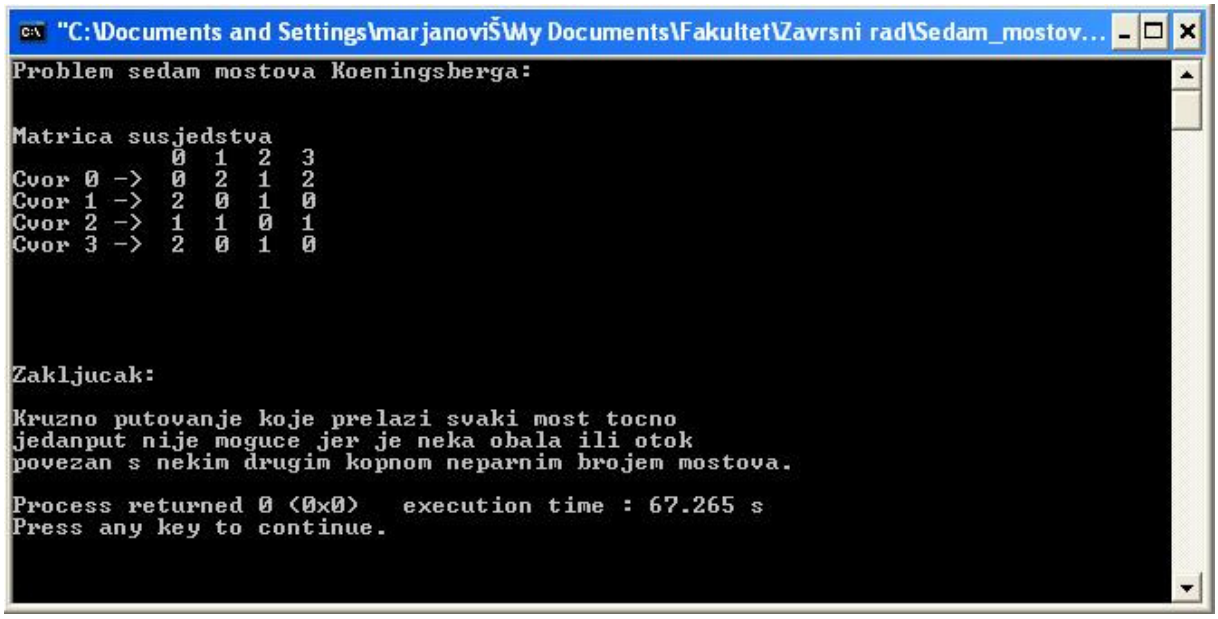
1. Početak
2. Program povezuje 4 čvora sa 7 bridova
3. Program generira matricu susjedstva s obzirom na čvorove i bridove
4. Program provjerava je li moguće prijeći svaki brid točno jednom na način da broji čvorove sa neparnim bridovima
5. Ako je broj čvorova sa neparnim bridovima različit od 0 ili 2 zaključuje da problem nema rješenje i ispisuje odgovarajuću poruku na ekran
6. Kraj

Dakle, program se sastoji od dva dvodimenzionalna polja cijelih brojeva (graf, matrica_susjedstva), polja cijelih brojeva (zbroj_bridova) te cijelih brojeva broj_cvorova i broj_bridova. Broj_cvorova predstavlja broj otoka i obala te iznosi 4 dok broj_bridova predstavlja mostove koji povezuju otoke te iznosi 7. Program povezuje bridove i čvorove te poziva funkcije matrica_susjedstva() i je_li_moguce_proci().

Funkcija matrica_susjedstva generira matricu susjedstva s obzirom na čvorove i bridove koji su ranije uneseni te zbaraja retke matrice, odnosno bridove incidentne odgovarajućem vrhu i sprema ih u polje zbroj_bridova. Budući je raspored bridova i vrhova za problem sedam mostova unaprijed poznat matrica susjedstva ima sljedeći oblik:

$$\begin{bmatrix} 0 & 2 & 1 & 2 \\ 2 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix}$$

Funkcija je_li_moguce_proci zbraja čvorove sa neparnim bridovima i rezultat pohranjuje u varijablu broj_neparnih. Budući je broj_neparnih u ovom slučaju različit od nula i od dva zaključujemo (po Korolaru 2.1.) da problem nema rješenje te program na ekran ispisuje odgovarajuću poruku o tome.



6.2.1. Poopćenje problema

Problem sedam mostova može se poopćiti na sljedeći način: Neka n obala ili otoka povezuje m mostova. Je li moguće svaki od tih mostova prijeći točno jednom?

I ovaj se problem može riješiti pomoću matrice susjedstva na sličan način kao i prije. Način rada programa koji rješava ovaj problem dan je pseudokodm:

1. Početak
2. Korisnik unosi broj bridova i broj čvorova te je li graf usmjeren i težinski
3. Korisnik povezuje bridove sa čvorovima
4. Ako je graf težinski korisnik unosi težinu svakog brida
5. Program generira matricu susjedstva na osnovu unesenih čvorova i bridova te u slučaju usmjerenog grafa ispisuje odgovarajuću poruku ako čvor nije povezan ni s jednim čvorom, ako čvor ima bridove koji vode u njega ali ni jedan iz njega i ako čvor ima bridove prema drugim čvorovima ali ni jedan čvor nema bridove prema njemu.
6. Program provjerava je li moguće prijeći svaki brid točno jednom na način da broji čvorove sa neparnim bridovima
7. Ako je broj čvorova sa neparnim bridovima različit od 0 ili 2 zaključuje da problem nema rješenja i ispisuje poruku na ekran
8. Ako je broj čvorova sa neparnim bridovima točno dva zaključuje da problem ima rješenje ako i samo ako putovanje počinje u jednom i završava u drugom čvoru sa neparnim brojem bridova te ispisuje poruku na ekran
9. Inače, zaključuje da problem ima rješenje i ispisuje poruku na ekran
10. Kraj

Ovaj program ima iste globalne varijable kao i program koji rješava problem sedam mostova, dakle graf, matrica_susjedstva, zbroj_bridova, broj_cvorova i broj_bridova. U ovom slučaju broj čvorova i bridova nije unaprijed poznat te ga unosi korisnik pri pokretanju programa. Jednako tako korisnik unosi veze između bridova i čvorova. Nakon toga se pozivaju funkcije matrica_susjed() i je_li_moguce_proci().

Funkcija matrica_susjed generira matricu susjedstva s obzirom na unesene čvorove i bridove te kao i prethodno zbraja retke matrice. Ova funkcija je nešto složenija od funkcije matrica_susjed u prethodnom programu jer se mora uzeti u obzir da graf može biti usmjeren pa ovisno o tome postoje:

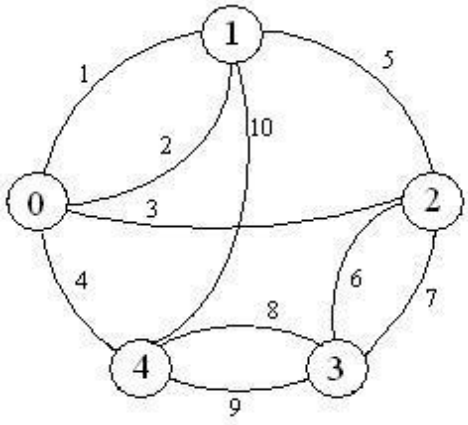
a) čvorovi koji nisu povezani sa niti jednim čvorom
b) čvorovi sa bridovima koji vode u njega ali ne i iz njega

c) čvorovi sa bridovima koji vode prema drugim čvorovima ali ni jednim koji vodi prema njemu

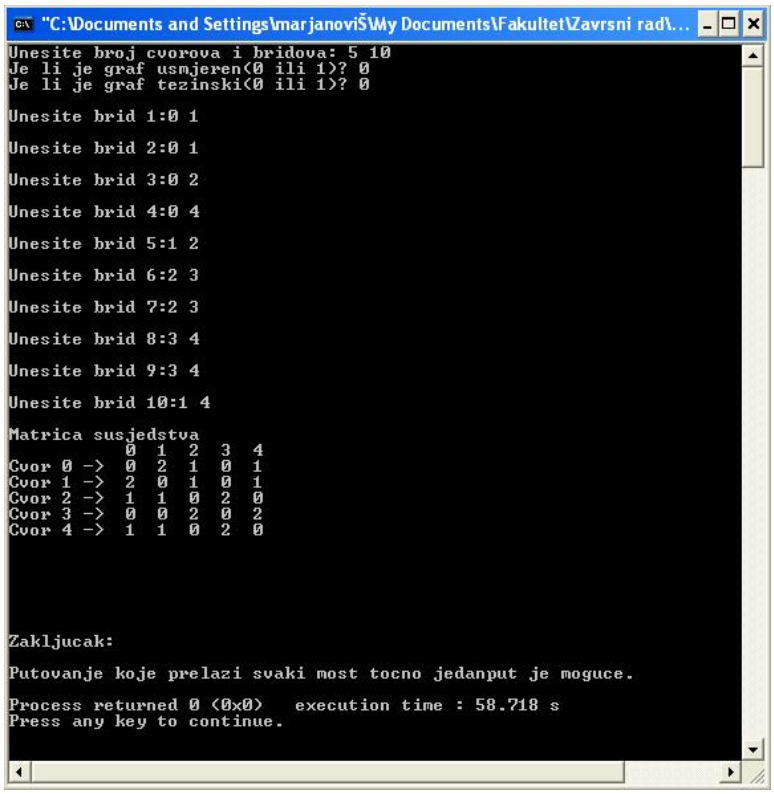
d) čvorovi sa bridovima koji vode iz njega i prema njemu

Funkcija je_li_moguce_proci zbraja vrhove sa neparnim brojem bridova i u ovisnosti o tome zaključuje je li moguće rješenje problema.

Primjer: Postoji li rješenje problema za graf sa 5 vrhova i 10 bridova koji su povezani kao na slici ispod?



Rješenje:



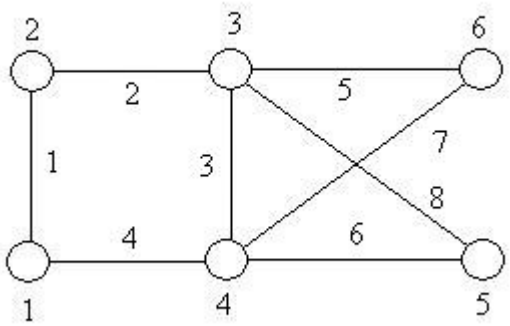
6.3. Rješenje problema pomoću Fleuryjevog algoritma

Ako je G Eulerov graf, onda je svaka Eulerova tura na G optimalna jer se njome svaki brid prijeđe točno jedanput. U tom slučaju problem se riješava Fleuryjevim algoritmom koji konstruira Eulerovu turu tako da počne u nekom vrhu i u svakom koraku bira vezni brid neprijeđenog podgrafa samo da nema alternative. Lako je utvrditi sadrži li graf Eulerovu stazu, tj. Eulerovu turu. Ostaje jedino problem kako ih naći. To se radi pomoću Fleuryjevog algoritma. Fleuryjev algoritam je jednostavan ali veoma spor algoritam za nalaženje Eulerove ture.

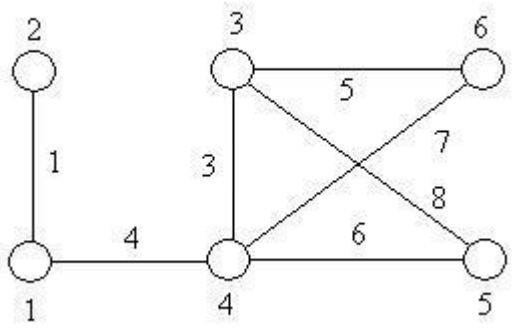
Postupak prema tom algoritmu je sljedeći:

1. Početak
2. Program izabere neki vrh i ide dalje duž bilo kojeg ruba tog vrha na drugi vrh. Nakon toga ukloni taj rub s grafa.
3. Sada smo na vrhu izmijenjenog grafa. Program odabere bilo koji rub od ovog tjemena ali ne smije biti rezni rub (osim u slučaju da nema druge mogućnosti). Ide dalje duž odabranog ruba i uklanja taj rub s grafa.
4. Ponavlja treći korak dok ne iskoristi sve rubove i dođe natrag na vrh s kojeg je započeo.
5. Kraj

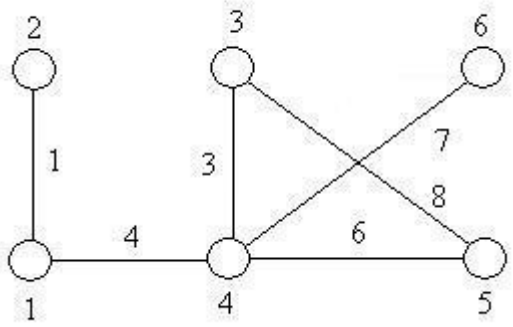
Primjer nalaženja Eulerove ture na grafu sa 6 vrhova i 8 bridova:



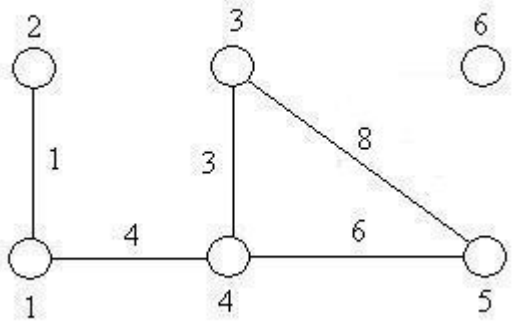
1. Uklonimo brid 2 i dobijemo sljedeći graf



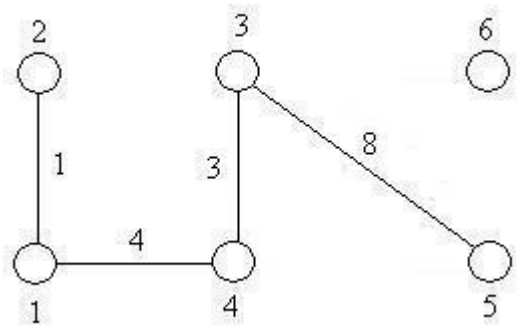
2. Uklonima brid 5 i dobijemo sljedeći graf



3. Uklonimo brid 7 i dobijemo sljedeći graf.



4. Uklonimo brid 6 i dobijemo sljedeći graf



5. Završimo prolaskom kroz bridove 8 → 3 → 4 → 1
6. Dakle, kompletna Eulerova tura je 2 → 5 → 7 → 6 → 8 → 3 → 4 → 1

6.4. Problem kineskog poštara

Problem sedam mostova ima primjenu na mnoge probleme iz teorije grafova. Jedan od poznatijih je problem kineskog poštara koji se može riješiti pomoću Fleuryjevog algoritma.

Kineski poštar treba pokupiti pisma u poštanskom uredu, dostaviti pisma u svim ulicama koje spadaju u njegovo područje i na kraju se vratiti u poštanski ured i pri tome posao treba obaviti sa što manje hodanja kako bi uštedio trud i vrijeme. Dakle, problem kineskog poštara je primjer u kojem pokušavamo naći šetnju kojom prolazimo kroz svaki brid na grafu samo jednom i to učiniti na najkraći mogući način koristeći se usmjerenim ili neusmjerenim grafom. Primjetimo da se problem kineskog poštara razlikuje od problema sedam mostova u tome što problem kineskog poštara zahtjeva da se vratimo u početnu točku i da prilikom putovanja pronađemo najkraći put.

7. Zaključak

I danas, 272 godine nakon Eulerovog rješenja problema Koeningsberških mostova ovaj probmel privlači pažnju i još je uvijek aktualan. Stoga se može reći da je to jedan od najljepših problema teorije grafova.

8. Popis literature

1. Darko Veljan, Kombinatorna i diskretna matematika, Algoritam, Zagreb, 2001.
2. Jurijan Šribar i Boris Motik, Demistificirani C++, Element, Zagrab, 2006.
3. https://sites.google.com/site/sandasutalo/osnove-programiranja/polja-nizovi/visedimenzionalno-polje
4. http://www.fer.unizg.hr/_download/repository/05-Polja-CB.pdf
5. http://hr.wikipedia.org/wiki/Leonhard_Euler
6. http://hr.wikipedia.org/wiki/C_(programski_jezik)
7. http://e.math.hr/math_e_article/br14/fosner_kramberger
8. http://biblio.irb.hr/prikazi-rad?&lang=en...&rad=558428
9. http://www.selimbegovic.com/Dokumenti/Metricki%20problemi%20u%20teoriji%20grafova.pdf