KLASA MATRICA

(Rijetka Matrica)

DOKUMENTACIJA ZA PRVI PROJEKAT

- *Tema 4*

Predmet:
STRUKTURE PODATAKA I ALGORITMI

Student:
SANJA SALIHOVIĆ
salihovicsanja5mail.com

Datum:
25.12.2019.

Univerzitet u Sarajevu Prirodno-matematički fakultet Odsjek za matematiku

Sadržaj

Uvod

Za matricu kažemo da je rijetko popunjena matrica ili "rijetka matrica" ukoliko je većina njenih elemenata jednaka nuli, a elementi koji nisu jednaki nuli obično su pravilno raspoređeni po matrici.

U ovom projektu zadatak je napraviti klasu *Matrica*, koja matricu čuva kao listu listi. Pošto radimo sa tzv. „rijetkim matricama", nema ih smisla čuvati kao nizove nizova, jer bi se rad sa njima sveo na sabiranje i množenje nula.

Ako matrica formata 1000 × 1000 sadrži samo 4 nenulta elementa i to $a_{2,10} = 5$, $a_{2,400} = 15$, $a_{100,50} = 25$, $a_{700,800} = 35$, onda klasa *Matrica* treba čuvati samo listu od 3 liste (od kojih prva ima dva elementa 5 i 15, a preostale dvije po jedan, 25 pa 35).

Drugim riječima, pamti se lista koja sadrži liste u kojima se ne nalaze nulti elementi, tj. one liste u kojima se nalaze nenulti elementi.

Mora se voditi evidencija na koji se red koja lista odnosi, ali i unutar tih listi treba za svaki element voditi evidenciju u kojoj je koloni.

Potrebno je da podržati sabiranje, oduzimanje, množenje i stepenovanje matrica, kao i transponovanje matrice.

Za ovu klasu treba podržati konstruktor sa dva parametra tipa *int*, koji kreira matricu datih dimenzija popunjenu nulama, te konstruktor koji prima dimenzije matrice i vektor koji se sadrži od nenultih elemenata matrice, a elementi vektora su tipa *pair<pair<int,int>,double>*, pri čemu prvi element para predstavlja red i kolonu elementa, a drugi njegovu vrijednost.

ORGANIZACIJA

O Matrici

Za klasu Matrica koriste se dvije strukture; Element i Red.

struct Element

Struktura Element ima atribute: int kolona; double sadrzaj i Element* veza. Dakle, ova struktura ima zadatak da čuva podatke o elementima određenog reda matrice, tačnije o indeksu kolone elementa u matrici, o sadržaju/informaciji elementa na tom mjestu matrice u datoj koloni, te pokazivač na naredni element u redu matrice.

struct Red

Struktura Red ima zadatak da čuva indekse elemenata u redu matrice, zatim pokazivače na elemente reda, pokazivač na naredni red u matrici i pokazivač na trenutni element u redu, koji se postavlja na nullptr. Drugim riječima, struktura Red se sastoji od sljedećih atributa: int index, Element* elementi, Red* veza i Element *trenutni.

private:

Klasa Matrica ima privatne atribute: broj redova i broj kolona, koji su tipa int i pokazivač na matricu (na prvi red u matrici) i postavljamo ga na null pokazivač jer smatramo da se još uvijek ništa ne nalazi u matrici. Matrica je organizovana kao lista nenultih redova gdje svaki red zatim ima listu elemenata koji se nalaze u tom redu (ukoliko su nenulti).

Privatni dio klase također sadrži i neke funkcije.
To su:

- void DodajElement(int i, int j, double el);
- void KreirajMatricuOptimizovano(const vector<pair<pair<int, int>, double>> &elementi);
- static Element *SpojiElement(Red *c, Element *c_el, Element *novi);
- Red *DodajRed(Red *r, Red *c, bool minus);
- Red *SpojiRed(Red *c, Red *novi_red);
- static Matrica SaberiOduzmi(const Matrica &m1, const Matrica &m2, char op);
- static double IzvrsiOperaciju(double a, double b, char op);

public:

Javni ili public dio klase ima konstruktore, operatore, destruktor i neke pomoćne funkcije, kao i operatore.
Ovdje ćemo navesti sadržaj javnog dijela, a nekon toga detaljno opisati svaku od navedenih metoda.

- Matrica(int broj_redova, int broj_kolona);
- Matrica(int broj_redova, int broj_kolona, const vector< pair<pair<int,int>,double>> &elementi, bool optimizacija);
- Matrica(const Matrica &m);
- Matrica& operator =(const Matrica &m);
- ~Matrica();
- Matrica transponuj();
- Matrica stepenuj(int stepen);
- Matrica &T();
- Matrica &Pow(int stepen);

- friend ostream& operator<<(ostream& tok, const Matrica &m);
- friend Matrica operator +(const Matrica &m1, const Matrica &m2);
- friend Matrica operator -(const Matrica &m1, const Matrica &m2);
- friend Matrica operator*(const Matrica &m1, const Matrica &m2);
- Matrica operator*(const double &c);
- Matrica &operator*=(const double &c);
- Matrica operator/(const double &c);
- Matrica &operator/=(const double &c);
- Matrica &operator+=(const Matrica &m);
- Matrica &operator-=(const Matrica &m);
- Matrica &operator*=(const Matrica &m);
- double element(int i, int j) const;

POMOĆNE FUNKCIJE

void DodajElement(int i, int j, double el);

Kod ove funkcije neophodni su parametri indeksa reda i kolone gdje se dodaje element i element tipa double koji se dodaje u matricu.
Na početku se provjerava da li su indeksi elemenata u opsegu matrice i baca se izuzetak ukoliko nisu.
Ukoliko je matrica prazna, poziva se konstruktor nad strukturom Red i kreira se jedan red u matrici, zatim se dinamički alocira novi element koji se smješta utaj red u matricu.
Ako već postoje neki redovi u matrici, traži se mjesto gdje da se ubaci element. Sve dok pokazivač na trenutni red u matrici pokazuje na nešto i sve dok je indeks reda manji od indeksa elementa trba da se pomjera pokazivač reda. Potreban je jedan pokazivač koji ako je jednak nuli će pokazivati na matricu a u suprotnom neka pokazuje na naredni red u matrici.

Ako je red prazan ili je indeks reda veći od indeksa elementa koji se ubacuje tada se pomoću strukture Red dinamički alocira pokazivač na novi element i ako nema elemenata u redu, ubaci se novi, a ako ima, element se doda na kraj reda.

Nakon što se provjerilo da postoji red u matrici, postavlja se pitanje gdje tačno dodati element u matricu.
Postavlja se pokazivač na trenutni red u koji dodajemo element i pokazivač na bilo koji nenulti element (onaj koji se dodaje).
Ako je trenutni red prazan, ubacimo mu element.
Ako nije, doda se element nakon prvog elementa koji ne pokazuje ni na šta u jednom redu

Ukoliko se pokuša postaviti element koji je već postavljen na to isto mjesto, baci se izuzetak.

void KreirajMatricuOptimizovano(const vector<pair<pair<int, int>, double>> &elementi);

Optimizovano znači da svaki element vektora ima sortirane parove indeksa.
Kod ove funkcije prvi uslov koji se provjerava jeste da matrica nije prazna, a zatim se kreiraju sortirani redovi elemenata u matrici.

static Element *SpojiElement(Red *c, Element *c_el, Element *novi);

Ovo je statička pomoćna funkcija koja se koristi u funkcijama DodajRed, SaberiOduzmi. Ona vraća pokazivač na elemente i ima parametre: jedan pokazivač na red u kojem se nalaze elementi, pokazivač na element na koji se želi dodati novi element i pokazivač na novi element – element koji spajamo sa prethodnim.
Ako nema elemenata u redu (pokazuju na null pokazivač), dodaje se naš novi, a ako već postoji neki element u redu, samo se poveže novi s njim, tj. doda se nakon već postojećeg.

Red *DodajRed(Red *r, Red *c, bool minus);

Ovo je pomoćna i ona vraća pokazivač na red koji se dodaje, a ima parametre: pokazivač na prvi red, pokazivač na red matrice i varijablu minus tipa bool koja je postavljena na vrijednost false(dakle plus je).
Prvo se alocira novi prazni red koji pokazuje na indeks reda i dva pokazivača za naredni red i elemente reda, koji još uvijek ne pokazuju ni na šta. Zatim se kreira matrica u kojoj se pomoću funkcije SpojiRed dodaju/spajaju jedan za drugim novi redovi.
Pokazivač *el* pokazuje na elemente jednog reda, a pokazivač *c_el* se kreće kroz elemete od reda koji se pravi u matrici c.
Sve dok se ne popuni red do kraja, treba u novi sadrzaj smiještati elemente.

Ako je minus operacija, tada se nad novim sadržajem treba iskoristiti funkcija IzvrsiOperaciju. Dinamički se alocira novi element koji se posmatra, a pokazivač za kretanje kroz elemente reda pomoću funkcije SpojiElement, spaja, odnosno stavlja te elemente u jedan red koji će se dodati na matricu.

Red *SpojiRed(Red *c, Red *novi_red);

Ovo je također pomoćna funkcija koja se koristi u
Funkcija vraća pokazivač na red matrice i posjeduje dva parametra - pokazivač na već postojeći red i pokazivač na red koji se dodaje(spaja).
Ako je matrica prazna (ima prazan red), ona postaje samo novi red koji se kreira i nije više prazna matrica.
Ukoliko već postoji neki red u matrici, tada se novi kreirani red spaja s njim, tj. s matricom.

static Matrica SaberiOduzmi(const Matrica &m1, const Matrica &m2, char op);

Ovo je statička funkcija članica koja se nalazi u privatnom dijelu klase. To je klasična funkcija sa ograničenim pravom pristupa. Unutar ove funkcije ne postoji pokazivač „this“, tako da ona ne može znati nad kojim objektom je pozvana. Ovdje se koriste pomoćne funkcije IzvrsiOperaciju, DodajRed, SpojiElement.
Ova funkcija sabira ili oduzima dvije matrice. Prolazeći redovima i posmatrajući elemente sa istim indeksima izvršava odgovarajuću operaciju s njima.

static double IzvrsiOperaciju(double a, double b, char op);

Ova statička funkcija članica se također nalazi u privatnom dijelu klase kao i prethodna statička funkcija i njena implementacija je zasnovana na provjeri operacije tj. opertora. Ukoliko je operator znak „+“, vraća se double vrijednost zbira elemenata. S druge strane, ako je operator znak „-“, vraća se double vrijednost razlike elemenata. Ovu funkciju koristimo u funkciji SaberiOduzmi.

double element(int i, int j) const;

Pomoćna funkcija koja traži element i pojavljuje se u operatorskoj funkciji množenja.

KONSTRUKTORI

Matrica(int broj_redova, int broj_kolona): broj_redova(broj_redova), broj_kolona(broj_kolona){ }

Ovaj konstruktor ima dva prametra, broj redova i broj kolona, a on se koristi da obezbjedi veličinu matrice. Na početku su to podrazumijevane vrijednosti - nule, dakle matrica je prazna.

Matrica(int broj_redova, int broj_kolona, const vector< pair<pair<int,int>,double>> &elementi, bool optimizacija){ }

Ovo je konstruktor koji prima dimenzije matrice i vektor koji se sadrži od nenultih elemenata matrice, a elementi vektora su tipa pair<pair<int,int>,double>, pri čemu prvi element para predstavlja red i kolonu elementa, a drugi njegovu vrijednost. Pošto ne smijemo mijenjati treći parameter, tj. vektore elemenata matrice, ispred tipa stavljamo *const*, a referencu & prije naziva varijable kako bi spriječila kopiranje, odnosno optimizirala prijenos parametra.
Kod implementacije ovog konstruktora, provjerava se da li je poslana "optimizovana verzija" matrice jer tada neće biti potrebe da se sortira matrica naknadno i poziva se funkcija KreirajMatricuOptimizovano, a ako korisnik ne pošalje u ispravnom redu matricu, tada se koristi neoptimizovana verzija, matrica se mora prvo sortitati pozivom funkcije DodajElement.
Ako korisnik kaže da šalje optimizovanu verziju(postavi optimizaciju na bool vrijednost true), to je bolje za njega jer će ga to manje koštati i pomoći će programu jer će tada znati da se kreira sortirana matrica. Ako postavi optimizaciju na false, tada se prolazi kroz cijelu matricu i traži odgovarajuće mjesto za element, sto naravno nije optimalno postupak..

KONSTRUKTOR KOPIJE I OPERATOR DODJELE

Matrica(const Matrica &m);

Tzv. „kopirajući konstruktor“ nam omogćava kopiranje vrijednosti jednog objekta u drugi.
Koriste se dva pokazivača na redove matrice, jedan za prolaz kroz matricu koja se kopira, a drugi za prolaz kroz matricu koja se kreira. Prolazeći while petljom dužinom matrice, tj. sve dok se ne dođe do zadnjeg reda u matrici koji pokazuje na nullptr (do kraja matrice), dinamički alociramo novu matricu sa pokazivačima na nove prazne redove, istih dimenzija kao orginalna matrica.

Ukoliko je matrica prazna (pokazivači su „0“ pokazivači), samo joj se dodijele pokazivači na nove prazne redove, odnosno ona postaje nova alocirana prazna matrica.

Ako matrica nije prazna, tada se kopiraju redovi elemenata u nove redove matrice koja se kreira. Ovim se kopiraju redovi matrice u novu matricu.
Potrebno je proći kroz sve elemente reda, tačnije kroz svaku kolonu matrice koja se kopira i isto kao i za redove, treba dinamički alocirati nove elemente matrice koji imaju svoju kolonu, sadržaj i pokazivač, sve dok se ne dođe do posljednjeg elementa u redu.

Matrica& operator =(const Matrica &m);

Kod operatora dodjele, prvo se provjerava da li korisnik želi izvršiti samododjelu. Dakle, ukoliko dva pokazivača pokazuju na isti objekat (na isti element matrice, isti red, kolonu...), nema potrebe da se uništava već postojeći, identični objekat i kreira opet isti takav.

Preciznije, funkcija koja realizira preklopljeni operator dodjele nikada ne smije brisati odredišni objekat bez prethodne garancije da on nije ekvivalentan izvornom objektu.
Ovaj operator dodjele se nikad ne poziva pri inicijalizaciji objekata, nego samo kada se dodjela vrši nad objektom koji od ranije postoji. tako da je za njegove potrebe već alocirana memorija.
Ovdje operator dodjele prvo izvrši sve ono što bi radio destruktor, a zatim sve ono što bi radio kopirajući konstruktor.

Naime, kako se operator dodjele primjenjuje nad objektom koji od ranije postoji, za njegove potrebe je već alocirana memorija. Stoga bi bolja ideja bila da se provjeri da li je alocirana količina memorije dovoljna da prihvati podatke koje treba kopirati. Ukoliko jeste, dovoljno je samo izvršiti kopiranje. Međutim ukoliko nije, potrebno je alocirati neophodnu količinu memorije, obaviti kopiranje u novi prostor, te dealocirati stari prostor.

DESTRUKTOR

~Matrica();

Destruktor treba da oslobodi dodatne resurse koje je objekat zauzeo tokom svog života. Kreira se pokazivač na red u matrici. Sve dok se ne dođe do kraja matrice, treba da se kreira pokazivač na trenutni element u tom redu. Ponovo, sve dok ima elemnata u redu oni se brišu i na kraju se ošriše i red(koji je prazan).

OPERATOR ISPISA

friend ostream &operator<<(ostream &tok, const Matrica &m);

Ovo je friend funkcija članica koja se poziva nad nekim objektom i ispisuje sadržaj koji posjeduje.
Kreira se pokazivač na trenutni red u matrici m i pokazivač na element koji trenutno ne pokazuje ni na šta.
Pomoću jednog indeksa koji se inicijalizira na nulu prolazi se kroz sve redove u matrici. Ukoliko neki red ne postoji, tj. on sadrži sve nule, trebaju se ispisati sve nule u tom redu.
Ako red postoji, tj. ako elementi tog reda nisu jednaki nuli ispiše se sadržaj elementa.

SABIRANJE I ODUZIMANJE

friend Matrica operator +(const Matrica &m1, const Matrica &m2);

friend Matrica operator -(const Matrica &m1, const Matrica &m2);

Ove dvije friend funkcije članice provjeravaju da li su dvije matrice saglasne za sabiranje, odnosno oduzimanje. Za dvije matrice kažemo da se mogu sabrati, odnosno oduzeti, ako i samo ako imaju isti broj redova i isti broj kolona.
Ukoliko nisu dobre dimenzije matrica za navedene operacije, baca se izuzetak sa odgovarajućom porukom, a u suprotnom funkcija vraća funkciju SaberiOduzmi sa operatorom '+' ili '-' (u zavisnosti šta korisnik želi), čiji je povratni tip matrica(rezultat).

Matrica &operator+=(const Matrica &m);
Ovaj operator dodaje neku konstantu na objekat nad kojim se poziva.

Matrica &operator-=(const Matrica &m);
Ovaj operator oduzima neku konstantu od objekta nad kojim se poziva.

MNOŽENJE I DIJELJENJE

friend Matrica operator*(const Matrica &m1, const Matrica &m2);
Prijateljska funkcija koja vraća matricu kao rezultat množenja dvije matrice. Koristi se pomoćna funkcija element koja pomaže pri množenju pravih elemenata na ispravnim indeksima i funkcija DodajElement koja dodaje sume redova i kolona na odgovarajuću poziciju u rezultantnu matricu.

Matrica operator*(const double &c);
Pomoćna funkcija članica operator "puta", kao rezultat vraća Matricu pomnoženu nekom konstantom, tj. množi objekad nad kojim je pozvana sa nekom konstantnom vrijednošću.

Matrica &operator*=(const double &c);
Prolazeći kroz nenulte redove matrice, isto kao i kod dijeljenja, elementi u matrici se množe određenom konstantom.

Matrica &operator*=(const Matrica &m);
Ova pomoćna funkcija množi objekat nad kojim se poziva.

Matrica operator/(const double &c);

Ovo je operator koji vraća matricu podjeljenu sa nekom konstantom.

Matrica &operator/=(const double &c);

Prolazeći kroz matricu i pokazujući na svaki trenutni dodani red koji nije prazan i njegove elemente, tj. sadržaj tih elemenata treba da bude podijeljen sa nekom konstantom.
Ako je ta konstanta jednaka nuli, baca se izuzetak sa odgovarajućom porukom da se pokušava dijeliti sa nulom.

TRANSPONOVANJE

Matrica transponuj();

Transponovanje matrice je urađeno korištenjem funkcije DodajElement. Koristeći konstruktor sa parametrima dimenzija matrice, kreirala se varijabla transponovana(matrica) koja ima dimenzije broj_kolona * broj_redova. Kreira se pokazivač na redove matrice, kao i pokazivači na elemente u redovima matrice. U kreiranu matricu transponovana dodaje se jedan po jedan red u koji se smještaju elementi kolona orginalne matrice. Nakon što se smjesti jedan element, smješta se sljedeći itd., sve dok se ne popune svi redovi transponovane matrice.
Ova funkcija vraća transponovanu matricu.

Matrica &T();

Ova pomoćna funkcija transponuje objekat nad kojim se poziva.

STEPENOVANJE

Matrica stepenuj(int stepen);

Ovo je pomoćna funkcija koja se koristi za stepenovanje matrice. Stepenovanje matrice se defniše tako što matricu množimo samu sa sobom onoliko puta koliki je stepen. Množimo matricu tako što pozivamo operator množenja (puta) i množimo je samu sa sobom onoliko puta koliki je stepen. Ako je stepen jednak nuli, rezultat je dijagonalna matrica (na dijagonali samo jedinice, ostalo nule). Ukoliko korisnik pošalje neki broj manji od nule, baca se izuzetak sa odgovarajućom porukom.
Ova funkcija vraća novu stepenovanu matricu kao rezultat i ima parametar stepen tipa int.

Matrica &Pow(int stepen);

Ova pomoćna funkcija stepenuje objekat nad kojim se poziva.

VALIDACIJA – OBRADA IZUZETAKA

throw logic_error("Definisali ste element više puta!");
Ovaj izuzetak se nalazi u funkciji void DodajElement koja korisniku pošalje poruku da je više puta htio definisati na istom mjestu neki element, što je nemoguće i dovodi do logičke greške.

throw domain_error("Stepen mora biti nenegativan!");
Ovaj izuzetak se nalazi u pomoćnoj funkciji za stepenovanje matrica Matrica stepenuj() i baca se ako korisnik unese neki negativan broj.

throw domain_error("Dijeljenje nulom!");
Ukoliko korisnik pokuša dijeliti sa nulom, dobit će poruku upozorenja. Konkretno, ovaj izuzetak se nalazi u operatorskoj funkciji dijeljenja (atrica &Matrica::operator/=(const double &c)).

throw domain_error("Matrice nisu saglasne za sabiranje.");
throw domain_error("Matrice nisu saglasne za oduzimanje.");

Ako nisu dobre dimenzije matrica za sabiranje i oduzimanje, funkcije Matrica operator+(const Matrica &m1, const Matrica &m2) i Matrica operator+(const Matrica &m1, const Matrica &m2), respektivno, bacaju ova dva navedena izuzetka.

throw domain_error("Indeks reda izvan opsega matrice.");
throw domain_error("Indeks reda izvan opsega matrice.");

Ova dva izuzetka se nalaze u funkciji DodajElement(int i, int j, double el) i vraćaju se korisniku ukoliko pošalje indekse koji nisu unutar dimenzija matrice.

Prilike za unaprijeđenje

Da bi se postigle bolje performanse *Matrice,* bilo bi dobro držati dinamički alociran niz redova koji kasnije imaju liste elemenata. To znači da bismo i za nulte redove imali alociran element u nizu, ali to je mala cijena koju plaćamo za ubrzan pristup redu koji je u tom slučaju onda kompleklsnosti O(1).
Zaključak je da je indeksacija niza znatno brža.