PRIRODNO MATEMATIČKI FAKULTET U BANJOJ LUCI

STUDIJSKI PROGRAM MATEMATIKA I INFORMATIKA

SMJER INFORMATIKA

RIJETKE MATRICE

Student:

Boris Balacki

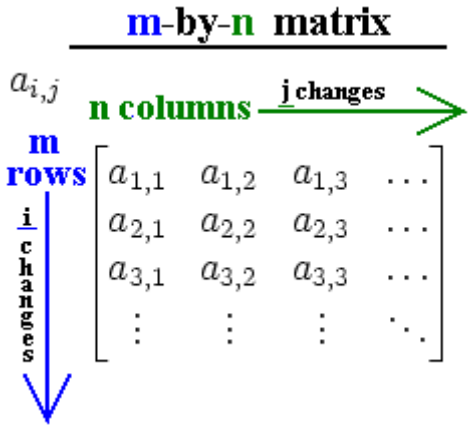
Mentor:

prof. dr. Dragan Matić

Sadržaj

1. Uvod

Dvodimenzionalni niz koji se još naziva i matrica, je niz čiji su elementi jednodimenzionalni nizovi i deklariše kao Niz[m][n].
Ovakav dvodimenzionalni niz se može predstaviti kao tabela koja ima m vrsta i n kolona.
Ako bismo imali matricu a[m][n], njene vrijednosti bi se tabelarno mogle predstaviti na sledeći način:



Ovde vidimo da se elementi jednodimenzionalnog niza identifikuju sa a[i][j] gdje su *i* i *j* pozitivni cijeli brojevi i predstavljaju indekse nizova. Matrice imaju široku primjenu u svijetu informatike.
U sledećim poglavljima upoznati ćemo se sa rijetkim matricama, prednostima i manama rijetkih matrica te njihovim računjem i primjenom rijetkih matrica u informatici.

2. Rijetke matrice

Rijetke matrice predstavljaju matrice čiji su većinski elementi nule. Ne postoji stroga definicija koliko mora biti nultih elemenata da bi se matrica smatrala rijetkom, ali najčešći kriterijum je da je broj nultih elemenata otprilike broj redova ili kolona.

Velike rijetke matrice često se pojavljuju u naučnim ili inženjerskim primjenama pri rješavanju parcijalnih diferencijalnih jednačina.

Prilikom skladištenja i manipulisanja rijetkim matricama na računaru, korisno je i često je potrebno koristiti specijalizovane algoritme i strukture podataka koji koriste prednost proređene strukture matrice. Operacije uz pomoć standardnih struktura i algoritama za guste matrice su spore i neefikasne kada se primjene na matrice malih gustina, jer se vrijeme i memorija troše na procesiranje nula.

2.1. Skladištenje rijetkih matrica

Matrica se obično čuva kao dvodimenzionalni niz. Svaki unos u niz predstavlja element *a[i,j]* matrice i njemu se pristupa putem dva indeksa *i* i *j*. Uobičajeno, *i* je indeks reda, numerisan od vrha do dna, a *j* indeks kolone, numerisan slijeva udesno. *Za* $m \times n$ matricu, količina memorije potrebna za čuvanje matrice u ovom formatu srazmerna je $m \times n$.

U slučaju rijetke matrice, zahtjev za značajnim smanjenjem memorije može se ostvariti skladištenjem samo ne-nultih unosa. U zavisnosti od broja i distribucije ne-nultih unosa, mogu se koristiti različite strukture podataka i ostvariti ogromne uštede u memoriji u poređenju sa osnovnim pristupom. Kompromis je u tome što pristup pojedinačnim elementima postaje složeniji i potrebne su dodatne strukture da bi se jednoznačno mogla povratiti originalna matrica.

2.2. Compressed sparse row (CSR) matrice

Compressed sparse row (CSR) format predstavlja matricu M preko tri (jednodimenzionalna) niza, koji sadrže ne-nulte vrijednosti, obim redova i indekse kolona. Ovaj format omogućava brz pristup redovima. CSR format se koristi najmanje od sredine 1960-ih, a prvi potpuni opis pojavio se 1967. godine.

CSR format čuva rijetku m × n matricu M u obliku reda koristeći tri (jednodimenzionalna) niza (V, COL_INDEX, ROW_INDEX). Neka NNZ označava broj ne-nultih unosa u M.

Primjer 1: Rijetka matrica formata 4 × 4.

$$M_{4\times4} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}$$

V= [5, 8, 3, 6]

COL_INDEX = [0, 1, 2, 1]

ROW_INDEX = [0, 1, 2, 3, 4]

CSR format je jos poznat kao i Yale format jer je prvi put objavljen 1977 godine u izvjestaju Yale Sparse Matrix sa računarske katedre Yale unuverziteta.

3. Algoritmi i način testiranja

3.1. Algoritam za sabiranje matrica

Algoritam 1: Suma matrica

1: Inicijalizacija matrice A

2: Inicijalizacija matrice B

3: Kreiranje matrice A

4: Kreiranje matrice B

5: Inicijalizacija matrice C

6: Sabiranje matrica A i B (C=A+B)

7: Zbir matrica A i B je sačuvan kao matrica C

4: Kraj

Algoritam počinje sa radom kreiranjem matrica A i B. Matrice A i B su nasumično kreirane matrice koje se sastoje od nula i jedinica.

Sabiranje matrica se vrši pomoću dvije ***for*** petlje. Prva for petlja kreće od ***i=0*** i ide do **i<m** tj. do broja redova matrice. Druga ***for*** petlja se pravi unutar predhodne **for** petlje i kreće od ***j=0*** i ide do ***j<n***.

Sabiranje se vrši tako što elemente matrice A saberemo sa elementima matrice B. Pri sabiranju elementi matrica A i B imaju iste indekse.

Matrice se mogu sabrati samo ako imaju isti broj redova i kolona.

```
for( i=0;  i<m;  i++)
{
      for( j=0;  j<m;  j++)
      {
            C[i][j]=A[i][j]+B[i][j]
      }
}
```

3.2. Algoritam za množenje matrica

Algoritam 2: Množenje matrica

1: Inicijalizacija matrice A

2: Inicijalizacija matrice B

3: Kreiranje matrice A

4: Kreiranje matrice B

5: Inicijalizacija matrice C

6: Množenje matrica A i B (C=A*B)

7: Rezultat moženja matrica A i B je sačuvan kao matrica C

4: Kraj

Algoritam počinje sa radom kreiranjem matrica A i B. Matrice A i B su nasumično kreirane matrice koje se sastoje od nula i jedinica.

Množenje matrica se vrši pomoću 3 ***for*** petlje. Prva for petlja kreće od ***i=0*** i ide do **i<m** tj. do broja redova matrice. Druga ***for*** petlja se pravi unutar predhodne **for** petlje i kreće od ***j=0*** i ide do ***j<n***. Treća for petlja se pravi unutar prethodne petlje i kreće od ***k=0 k< broj kolona druge matrice***.

Element rezultujuće matrice C u i -tom redu i j-oj koloni dobijemo tako što skaralno pomnožimo i -ti red matrice A sa j-tom kolonom matrice B.

U i -tom redu prve matrice A, prvi element pomnožimo sa prvim elementom j-e kolone matrice B, zatim drugi element i-tog reda pomnožimo sa drugom elementom j -e kolone, i tako redom do n -tih elementa. Potom ove proizvode saberemo.

Matrice se mogu pomnožiti samo ako je broj kolona prve matrice jednak broju redova druge matrice.

```
for (int i = 0; i < broj redova matrice A; i++)
	{
		for (int j = 0; j < broj kolona matrice B; j++)
		{
			C[i][j] postavljamo na 0

			for (int k = 0; k < broj kolona matrice C; k++)
			{
				C[i][j] = C[i][j] + A[i][k] * B[k][j]
			}
		}

	}
```

3.3. Algoritam za sabiranje matrica zapsanih u CSR format

Algoritam 3: Sabiranje CSR matrica

1: Inicijalizacija matrice A

2: Inicijalizacija matrice B

3: Kreiranje matrice A

4: Transformacija matice A u CSR format

5: Kreiranje matrice B

6: Transformacija matice B u CSR format

7: Inicijalizacija matrice C

8: Sabiranje matrica A i B (C = A + B)

7: Zbir matrica A i B je sačuvan kao matrica C

4: Kraj

Algoritam počinje sa radom kreiranjem matrica A i B. Matrice A i B su nasumično kreirane matrice koje se sastoje od nula i jedinica. Zatim se matrice A i B transformišu u CSR format.Transformacija se vrši tako što so se ne-nulti elementi matrice smještaju u tri niza, u prvi niz (***A[ ], B[ ]***) se smješta vrijednost elementa, u drugi niz (***IA[ ], IB[ ]***) se smješta vrijednost indeksa reda i u treći niz (***JA[ ], JB[ ]***) se smješta vijednost indeksa kolone ne-nultog elementa. Nakon transformacija matrica u CSR format počinje postupak sabiranja matrica. Sabiranje se vrši na sledeći način:

Inicijalizujemo tri indeksa ***i=0 , j=0, k=0***. Indeks ***i*** se koristi za prolazak kroz matricu ***A***, dok se indeks ***j*** koristi za prolazak kroz matricu ***B***. Prvo poredimo vrijednosti niza ***IA[i]*** i niza ***IB[j]***. Ukoliko je vrijednost elementa niza ***IA[i]*** manja od vrijednosti elementa niza IB[j], zaključujemo da se elementi matrice ne nalaze u istom redu i da je element matrice A na manjoj poziciji od elementa amatrice B. Vrijednosti elemenata nizova ***A[i], IA[i], JA[i]*** se kopiraju u odgovarajuće nizove ***C[k], IC[k],JC[k]***. Nakon kopiranja indeksi ***i*** i ***k*** se povećavaju za 1.

Ukoliko je vrijednost elementa niza ***IB[j]*** manja od vrijednosti elementa niza ***IA[i]***, zaključujemo da se elementi matrice ne nalaze u istom redu i da je element matrice ***B*** na manjoj poziciji od elementa amatrice ***A***. Vrijednosti elemenata nizova ***B[j], IB[j], JB[j]*** se kopiraju u odgovarajuće nizove ***C[k], IC[k], JC[k]***. Nakon kopiranja indeksi ***j*** i ***k*** se povećavaju za 1.

Ukoliko je vrijednost elementa ***IA[i]*** jednaka vrijednosti elementa niza ***IB[j]***, zaključujemo da se elementi matrica nalaze u istom redu. Zatim poredimo vrijednosti elemenata niza ***JA[i]*** i vrijednoti elementa niza ***JB[j]***. Ukoliko je vrijednost elementa ***JA[i]*** manja od vrijednoti elementa ***JB[j]*** zaključujemo da se element matrice A nalazi ispred elementa matrice B. Vrijednosti elemenata nizova ***A[i], IA[i], JA[i]*** se kopiraju u odgovarajuće nizove ***C[k], IC[k],JC[k]***. Nakon kopiranja indeksi ***i*** i ***k*** se povećavaju za 1. Ukoliko je vrijednost elementa ***JB[j]*** manja od vrijednoti elementa ***JA[i]*** zaključujemo da se element matrice B nalazi ispred elementa matrice A. Vrijednosti elemenata nizova ***B[j], IB[j], JB[j]*** se kopiraju u odgovarajuće nizove ***C[k], IC[k],JC[k]***. Nakon kopiranja indeksi ***j*** i ***k*** se povećavaju za 1.

Ukoliko je vrijednost elementa ***IA[i]*** jednaka vrijednosti elementa niza ***IB[j]*** i ako je vrijednost elementa ***JA[i]*** jednaka vrijednost elementa ***JB[j]*** zaključujemo da se elementi nalaze na istoj poziciji.

Vrijednosti elemenata nizova ***A[i]*** i ***B[j]*** se sabiraju i upisuju u niz ***C[k] (C[k]= A[i] + B[j])***. U nizove ***IC[k]*** se upisuje vrijednost ***IA[i]*** ili ***IB[j]*** a u niz ***JC[k]*** se upisuje vrijednost ***JA[i]*** ili ***JB[j]***. Nakon toga se indeksi ***i, j*** i ***k*** povećavaju za 1. Postupak se ponavlja sve dok ne dođemo do kraja nizova A i B.

Na kraju sabiranja suma CSR matrica A i B je predstavljena CSR matricom C koja je predstavljena nizovima ***C , IC, JC.***

3.4. Algoritam za množenje CSR matrica

Algoritam 4: Sabiranje CSR matrica

1: Inicijalizacija matrice A

2: Inicijalizacija matrice B

3: Kreiranje matrice A

4: Transformacija matice A u CSR format

5: Kreiranje matrice B

6: Transformacija matice B u CSR format

7: Inicijalizacija matrice C

8: Množenje matrica A i B (C = A + B)

7: Proizvod matrica A i B je sačuvan kao matrica C

4: Kraj

Algoritam počinje sa radom kreiranjem matrica A i B. Matrice A i B su nasumično kreirane matrice koje se sastoje od nula i jedinica. Zatim se matrice A i B transformišu u CSR format.Transformacija se vrši tako što so se ne-nulti elementi matrice smještaju u tri niza, u prvi niz (***A[ ], B[ ]***) se smješta vrijednost elementa, u drugi niz (***IA[ ], IB[ ]***) se smješta vrijednost indeksa reda i u treći niz (***JA[ ], JB[ ]***) se smješta vijednost indeksa kolone ne-nultog elementa. Nakon transformacija matrica u CSR format počinje postupak množenja matrica.

Množenje se vrši tako što pomoću dvije ***for*** petlje. Prva for petlja kreće od ***i=0*** i ide do **i<*broj ne-nultih elemenata prve matrice*** tj. Druga ***for*** petlja se pravi unutar predhodne ***for*** petlje i kreće od ***j=0*** i ide do ***j< broj ne-nultih elemenata druge matrice***. Inicijalizujemo broj **S=0.** Zatim postavljamo sledeći uslov: ako je ***IA[i] jednako JB[j], S+=A[i]+B[j].*** Kada unutrašnja ***for*** petlja prođe kroz sve elemente, suma ***S*** se se upisuje niz ***C[i], IA[i]*** se upisuje u ***IC[i]*** i ***JB[j]*** se upisuje u niz ***JC[i].***

Na kraju množenja proizvod CSR matrica A i B je predstavljen CSR matricom C koja je predstavljena nizovima ***C , IC, JC.***

3.5. Način testiranja

Ukupno je generisano:

- 540 instanci programa sume matrica koje su podijeljene u 6 grupa u zavisnosi od postotka nultih elemenata u matrici. Generisano je po 10 matrica veličina 10×10, 30×30 ,50×50, 70×70, 100×100, 500×500, 1000×1000, 3000×3000 i 5000×5000.

- 540 instanci programa sume CSR matrica koje su podijeljene u 6 grupa u zavisnosi od postotka nultih elemenata u matrici. Generisano je po 10 matrica veličina 10×10, 30×30, 50×50, 70×70, 100×100, 500×500, 1000×1000, 3000×3000 i 5000×5000.

- 420 instanci programa proizvoda matrica koji su podijeljene u 7 grupa u zavisnosti postotka nultih elemenata u matrici. Generisano je po 10 matrica veličina 10×10, 30×30 ,50×50, 70×70, 100×100, 500×500.

- 420 instanci programa proizvoda CSR matrica koji su podijeljene u 7 grupa u zavisnosti postotka nultih elemenata u matrici. Generisano je po 10 matrica veličina 10×10, 30×30 ,50×50, 70×70, 100×100, 500×500.

Programi su napisani u programskom jeziku C, a mjerenja operacija sabiranja i množenja su vršeni pomoću biblioteke <*time.h*>. Iz svake grupe uzeto je prosječno vrijeme izvršavanja operacije. Specifikacije laptopa kojim su vršenja mjerenja:

- Procesor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 4.20 GHz
- RAM memorija: 16GB DDR4 @ 3200 MHz
- Operativni sistem: Windows 11

4. Rezultati mjerenja

Veličina matrice	Postotak nula u matrici						
	40-50	50-60	60-70	70-80	80-90	90-99	
10×10	0	0	0	0	0	0	Vrijeme izavršavanja t[ms]
30×30	0	0	0	0	0	0	
50×50	0	0	0	0	0	0	
70×70	0	0	0	0	0	0	
100×100	0	0.1	0	0	0	0	
500×500	0.6	0.6	0.5	0.6	0.5	0.5	
1000×1000	2.1	1.7	1.7	1.7	1.7	1.6	
3000×3000	18	15.8	16.3	16.5	15.3	16.3	
5000×5000	50.6	40.6	41.3	48.6	41.3	43	

Tabela 1:Rezultati sume matrica

Veličina matrice	Postotak nula u matrici						
	40-50	50-60	60-70	70-80	80-90	90-99	
10×10	0	0	0	0	0	0	Vrijeme izavršavanja t[ms]
30×30	0	0	0	0	0	0	
50×50	0	0	0	0	0	0	
70×70	0	0	0	0	0	0	
100×100	0	0	0	0	0	0	
500×500	1.4	1.2	0.8	0.5	0.2	0	
1000×1000	4.9	4.2	3.3	2.3	1.2	0.4	
3000×3000	40.8	32.4	25.8	18.4	9.7	4.5	
5000×5000	113.3	89.6	67.2	49.6	29.1	9.8	

Tabela 2:Rezultati sume matrica zapisanih u CSR formatu

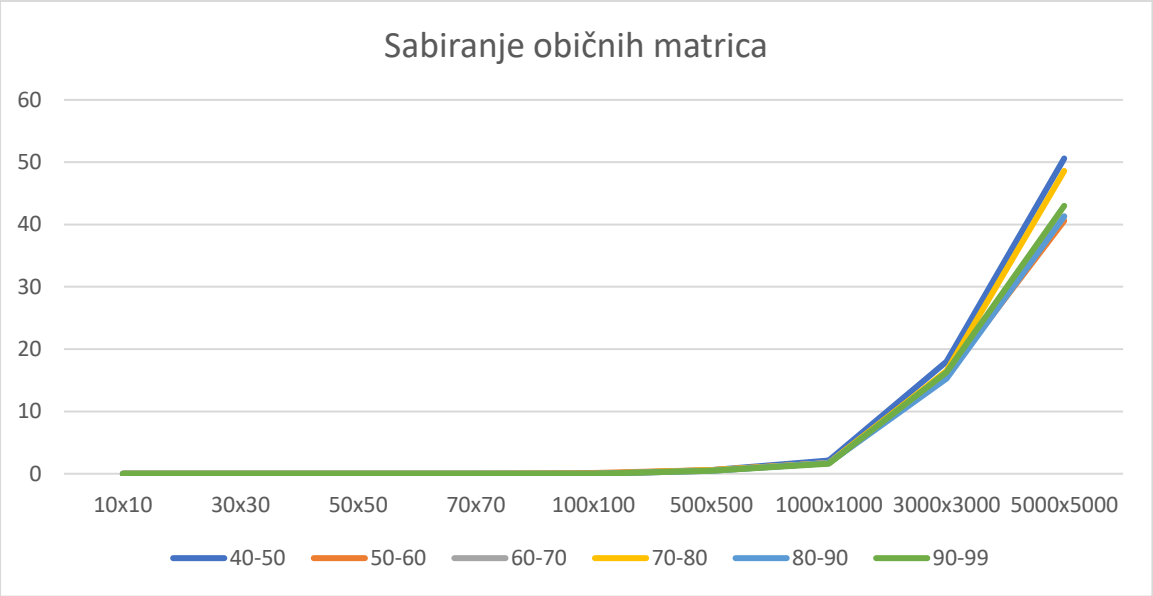
Veličina matrice	Postotak nula u matrici							
	40-50	50-60	60-70	70-80	80-90	90-99	99	
10×10	0	0	0	0	0	0	0	Vrijeme izavršavanja t[ms]
30×30	0	0.1	0.1	0	0	0	0	
50×50	0.3	0.5	0.1	0.5	0.4	0.4	0.5	
70×70	1.5	1	1	1	0.9	1	1	
100×100	3.2	3	2.9	2.9	3.3	3.2	3	
500×500	402.1	375	396.8	363.5	360.8	352.9	355.9	

Tabela 3:Rezultati množenja matrica

Veličina matrice	Postotak nula u matrici							
	40-50	50-60	60-70	70-80	80-90	90-99	99	
10×10	0	0	0	0	0	0	0	Vrijeme izvršavanja t[ms]
30×30	0.4	0.4	0.2	0.2	0	0	0	
50×50	4.5	2.2	1.2	1.4	0.9	0.2	0	
70×70	16.2	12.4	7	3.5	1.8	0.3	0	
100×100	94.6	56.7	31.5	14.3	5.1	1.1	0	
500×500	45840.2	29637.6	19253.5	8809.4	3990.3	313.1	15.9	

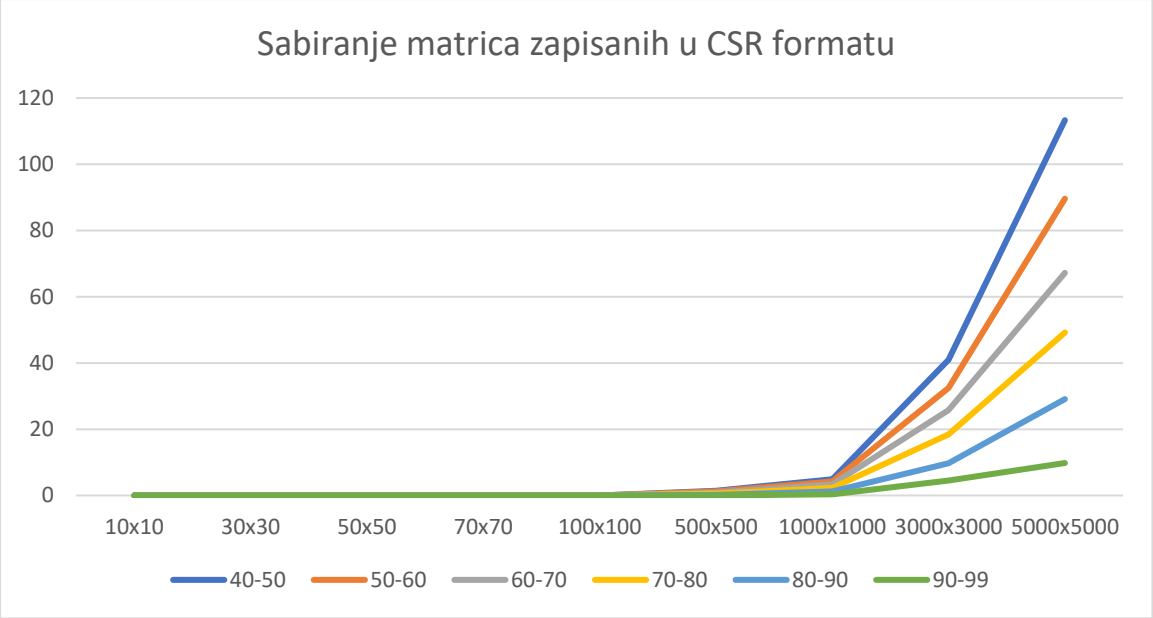
Tabela 3:Rezultati množenja CSR matrica

Iz dobijenih rezultata kod sabiranja običnih matrica možemo uočiti da vrijeme izvršavanja operacije većinski zavisi od veličine matrica, a ne od postotka nula u matrici. Povećanjem veličine matrica dolazi do povećanja vremena koje je potrebno da se izvrši operacija sabiranja matrica.



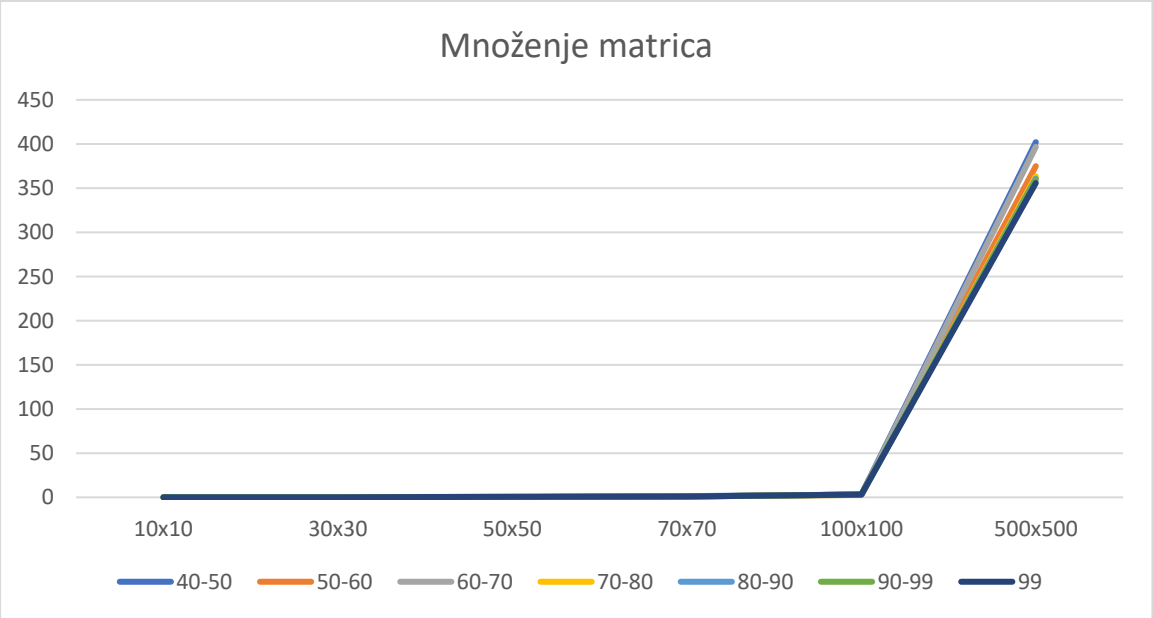
Dijagram 1: Sabiranje matrica

Kod sabiranja matrica predstavljenih u CSR obliku možemo uočiti da vrijeme izvršavanja operacije sabiranja zavisi od veličine matrica, ali i od postotka nula koje se nalaze u matrici. Povećanjem postotka nultih elemenata u matrici dolazi do skraćivanja vremena koje je potrebno da se operacija sabiranja izvrši.



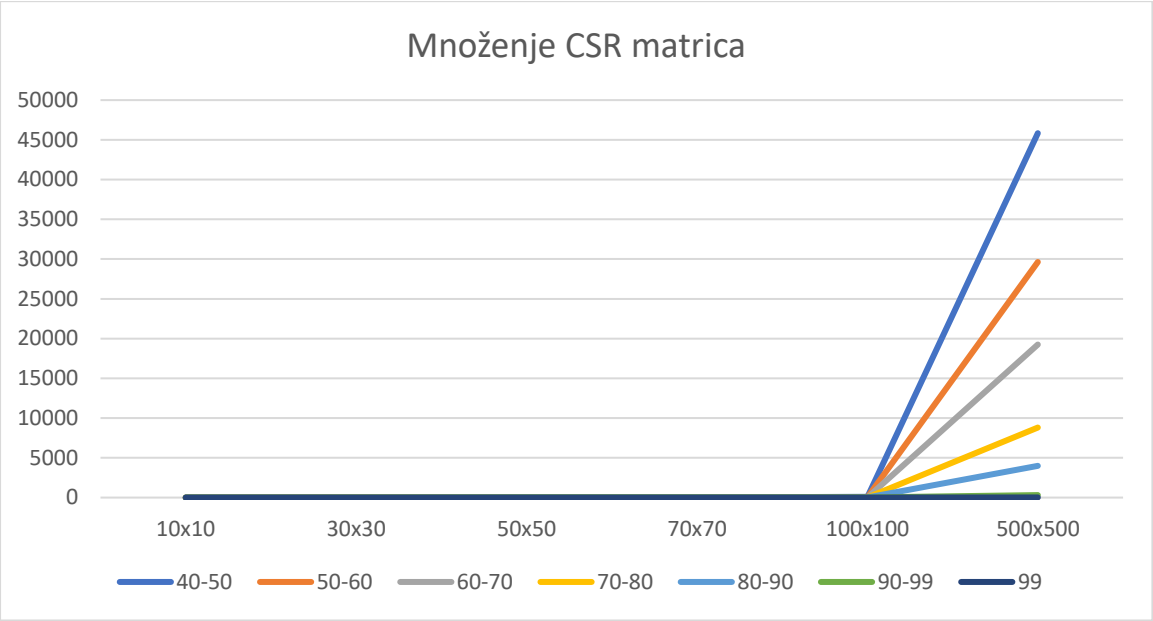
Dijagram 2: Sabiranje matrica zapisanih u CSR format

Kod množenja matrica možemo uočiti da vrijeme izvršavanja operacije uveliko zavisi od veličine matrice. Broj nultih elemenata ne utiče na brzinu izvršavanja programa. Povećanjem veličine matrica dolazi do značajnog povećanja vremena koje je potrebno da se izvrši operacija množenja matrica.



Dijagram 3: Množenje matrica

Kod množenja CSR matrica možemo uočiti da vrijeme izvršavanja operacije sabiranja zavisi od veličine matrica, ali i od postotka nula koje se nalaze u matrici. Povećanjem postotka nultih elemenata u matrici dolazi do drastičnog skraćivanja vremena koje je potrebno da se operacija množenja izvrši.



Dijagram 4: Množenje CSR matrica

5. Zaključak

Iz dobijenih rezultata možemo da uočimo da kod manjih matrica nema velikih razlika izmedju korištenja običnog algoritma za sabiranje marica i CSR algoritma. Kod većih matrica sa manjim postotkom nultih elemenata prednost ima obični algoritam za sabiranje.Sa povećavanjem broja nultih elemenata obični algoritam gubi na brzini. CSR algoritam kod velikih matrica sa malim brojem nultih elemenata gubi na brzini zbog rasta nizova koji su potrebi na čuvanje podataka o matricama. Opadanjem broja ne-nultih elemenata dolazi do značajnog povećanja brzine izvršavanja operacije sabiranja.

Zaključujemo da je obični algoritam za sabrianje matrica bolji za matrice koje imaju veći broj ne-nultih elemenata, dok je CSR algoritam bolji za matrice koje imaju veći broj nultih elemenata.

Kod množenja matrica zaključujemo da brzina operacije množenja značajno raste sa povećavanjem broja elemenata matrica. U slučaju CSR matrica, brzina množenja je značajno manja kod množenja matrica koje imaju manji broj ne-nultih elemenata u odnosu na obični algoritam za množenje matrica. Povećavanjem broja ne-nultih elemenata CSR algoritam dobija značajnu prednost u odnosu na obični algoritam.

6. Literatura

- https://www.geeksforgeeks.org/sparse-matrix-representations-set-3-csr/?ref=rp
- https://www.geeksforgeeks.org/operations-sparse-matrices/?ref=lbp
- https://en.wikipedia.org/wiki/Sparse_matrix
- https://dotnettutorials.net/lesson/addition-of-sparse-matrices/

Referenca ka C kodu: https://github.com/potekar/PMF---rijetke-matrice