Dokumentacija

1) Osnove

Matrica je implementirana sa 3 atributa: broj redova, broj kolona i dvostruki pokazivač. Svi konsturktori rade očekivano i u očekivanom vremenu $O(n^2)$, s tim da pomjerajući konstruktor i pomjerajući operator dodjele rade u $O(1)$

2) Operacije

Sabiranje i oduzimanje rade očekivano u $O(n^2)$, tako što se svaki element jedne matrice sabere sa svakim elementom od druge, pa se eventualno baci izuzetak.

Množenje radi pomoću „podijeli pa vladaj strategije“ koristeći „Strassen, 1969”. Prvo se matrice nadopune s nulama tako da budu istih dimenzija i to $2^t \times 2^t = n \times n$. Sada se ove matrice razbijaju na blok matrice formata $\frac{n}{2} \times \frac{n}{2}$. Ovime dobijamo 8 blok matrica. Označimo matrice sa A i B, sada $A \times B = C. A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$ gdje su A_{ij}, B_{ij}, C_{ij} blok matrice. Pomoću 7 rekurzivnih množenja možemo dobiti C. Sada je kompleksnost ovog procesa zapravo ekvivalentna sljedećoj rekurzivnoj relaciji $T(n) = 7T(n^2) + O(n^2)$, jer nam treba $O(n^2)$ da spojimo sve dijelove. Pomoću master teorema vidimo da je asimptotska kompleksnost algoritma $O(n^{lg\,7}) \approx O(n^{2.81})$. Postoje asimptotski brži algoritmi, ali su nepraktični jer su konstante prevelike.

Djeljenje radi tako što pomnožimo matricu sa inverznom matricom. Prije svega ovoga treba provjeriti da li se matrice uopće mogu pomnožiti i eventualno baciti izuzetak.

Stepenovanje je dosta kompleksnije i stepenovati možemo samo kvadratne matrice, pa to prvo provjerimo i bacimo odgovarajući izuzetak. Primjetimo u slučaju da je stepen negativan moramo koristiti inverznu matricu. Inverznu matricu nalazimo pomoću Gaus-Seidelove metode, tako što tražimo gornju i donju trougaonu, a u međuvremenu iste postupke ponavljamo na jediničnoj matrici istih dimenzija, pa je ta jedinična matrica zapravo inverzna (ako je matrica invertibilna to jest da joj je determinanta različita od 0). Kompleksnost traženja inverzne je $O(n^2)$, jer za svaki element na dijagonali vršimo operaciju na svaki element iznad i ispod njega. Sada kada treba da vršimo stepenovanje, mi radimo tkz. brzo stepenovanje. Ako je n parno onda imamo da je $a^n = a^{\frac{n}{2}} \times a^{\frac{n}{2}}$, a za n neparno $a^n = a^{\frac{n-1}{2}} \times a^{\frac{n-1}{2}} \times a$. Ovo radimo rekurzivno i dobijemo da je

kompleksnost algoritma ekvivalentna sljedećoj rekurzivnoj relaciji $T(n) = T\left(\frac{n}{2}\right) + O(k^{2.81})$, to je asimptotski jednako $O(k^{2.81}lgn)$ gdje je k dimenzija kvadratne matrice.

Transponovanje je poprilično jednostavno, samo zamjenimo redove i kolone, a za to nam treba $O(n^2)$, jer mi za svaki element matrice uradimo sljedeće $m[i][j] = m[j][i]$.

Determinanta matrice se dobije na sličan način kao inverzna, samo u ovom slučaju računamo samo gornju trougaonu matricu, pa množimo sve elemente na dijagonali(pivote), i tako dobijamo determinantu. Samo na početku mjenjamo redove matrica, tako da ukoliko je moguće bude nenulti element na mjestu pivota, s tim da kadgod zamjenimo dva reda pazimo da promjenimo znak rezultata. Naravno prije svega provjerimo da li je matrica kvadratna. Zamjena dva reda se vrši u $O(1)$, samo zamjenimo pokazivače, a traženje trougaone je $O(n^2)$, i množenje pivota je $O(n)$, pa je kompleksnost $O(n^2)$.

3) Ispis

Da bi ispisali matricu, trebamo prvo pronaći element matrice sa najdužim decimalnim zapisom. Sve elemente pretvaramo u stringove, pa uklanjamo 0 sa krajeva. Dužina najdužeg stringa nam je parametar za std::setw(), i tako ispisujemo elemente.

4) Upis (Parser i Evaluator)

Ako posmatramo standardan način stavljanja operanda u jedan stack, a operacija u drugi vidimo da nastaje problem. Nama su operandi matrice, brojevi i slova. Stack može da primi samo jedan tip podataka, pa napravimo klasu Operand koju će naslijediti tri klase: Broj, Znak i Matrica. Potrebne su nam 4 čiste virtualne metode: daj_broj(), daj_znak(), daj_matricu() i daj_tip(). Sada će tip stacka da bude pokazivač na Operand, pa zbog polimorfizma u njega možemo staviti pokazivače na Broj, Znak i Matricu. Pomoću ovih metoda, kada izvadimo Operand iz stacka, možemo provjeriti šta je to, i uzeti jednu od odgovarajućih metoda (daj_broj(), daj_znak(), daj_matricu()) i nad njom vršiti operacije. Ostatak se radi na standardan način gdje gledamo znak i predhodni znak, zatim ispitujemo slučajeve, pa bacamo odgovarajuće izuzetke.