Johdanto

Päätin toteuttaa matriisilaskimen staattisena kirjastoluokkana, jotta sen hyödyntäminen muissa projekteissa olisi mahdollisimman suoraviivaista. Tästä syystä ohjelmalla ei ole varsinaista käyttöliittymää, mutta sen github sivustosta löytyvä main luokka sisältää esimerkit metodien käytöstä ja niiden suorityskyvystä.

Ohjelman toiminnot ja toteutus

MatrixCalc-kirjastolla voidaan tällä hetkellä laskea kahden matriisin summa, erotus ja tulo, skaalata matriiseja, sekä laskea matriisin determinantti ja käänteismatriisi. Nykyisellään ohjelman metodit hyväksyvät matriiseja jotka koostuvat double-tyyppisistä liukuluvuista. Yksi tuleva kehityskohde on int- ja float-tyyppisten matriisien hyväksyminen, sillä tämän toteuttaminen on suoraan mahdollista nykyisillä operaatioilla.

Nykyisessä toteutuksessa kaikki metodit luovat palautusta varten uuden matriisin. Tämä lisää operaatioiden tilavaatimusta, mutta säilyttää molempien parametreina annettujen matriisien ominaisuudet jatkokäyttöä varten. Tämä valinta tehtiin koska se mielestäni sopii paremmin staattisen kirjaston tyyliin ja käyttötarkoituksiin.

Metodien toteutuksesta ja aikavaatimuksista

Summa-, erotus- ja skaalaus-operaatiot ovat triviaaleja, ja toteuttamalla nämä operaatiot naiivilla menetelmällä saavutetaan O(n) aikavaatimus. Summaus- ja erotus-operaatioissa ohjelma käy läpi annettujen matriisien jokaisen solun, summaa tai miinustaa tästä toisen annetun matriisin vastaavan solun ja kirjoittaa tuloksen palautettavan matriisin vastaavaan soluun. Skaalauksessa toimenpide on muuten sama, mutta toisena parametrina annetaan matriisin sijaan liukuluku, jolla ensimmäisen matriisin solut kerrotaan. Näissä operaatioissa ei ole paljoa parannettavaa, eikä annetun matriisin tai matriisien sisällöllä tai muodolla ole oleellista vaikutusta suoritustehoon.

Tärkeimpien, ei triviaalien metodien toteutus

Alla on tärkeimpien metodien toteutus pseudokoodina.

```
strassenWrapper (MatriisiA, MatriisiB, PisinSivu)
    AlkuperäinenRivimäärä = MatriisiA.Rivimäärä
    AlkuperäinenSarakemäärä = MatriisiB.Sarakemäärä
    LaskentaMatriisinKoko = Lähin kahden potenssi joka on >= PisinSivu
    Jos MatriisiA.Rivimäärä tai MatriisiA.Sarakemäärä != LaskentaMatriisinKoko
        Suurenna MatriisiA
    Jos MatriisiB.Rivimäärä tai MatriisiB.Sarakemäärä != LaskentaMatriisinKoko
        Suurenna MatriisiB
    MatriisiC = multiplyStrassen(MatriisiA, MatriisiB)
    Jos LaskentaMatriisinKoko = AlkuperäinenRivimäärä ja AlkuperäinenSarakemäärä
        Palauta MatriisiC
    Muuten
        Poista MatriisiC:stä rivit jotka ovat > AlkuperäinenRivimäärä ja sarakkeet
            jotka ovat > AlkuperäinenSarakemäärä
        Palauta MatriisiC
```

```
multiplyStrassen(MatriisiA, MatriisiB)
    MatriisinKoko = MatriisiA.Rivimäärä
    Puolipiste = MatriisinKoko / 2
    //Huom. 11 = Vasen yläkulma, 12 = Oikea yläkulma, 21 = Vasen alakulma, 22 = Oikea
    //alakulma
    Jaa MatriisiA ja MatriisiB neljään osaan (A11, A12 A21, A22, B11, B12, B21, B22)
    Jos MatriisinKoko < strassenCutoff
        //Luodaan Apumatriisit 1-7 kertomalla aikaisemmin luotuja neljäsmatriiseja
        //naiivilla metodilla
        Apumatriisi1 = multiplyNaive(A11 ja A22 summa, B11  ja B22 summa)
        Apumatriisi2 = multiplyNaive(A21 ja A22 summa, B11)
        Apumatriisi3 = mutliplyNaive(A11, B12 ja B22 erotus)
        Apumatriisi4 = multiplyNaive(A22, B21 ja B11 erotus)
        Apumatriisi5 = multiplyNaive(A11 ja A12 summa, B22)
        Apumatriisi6 = multiplyNaive(A21 ja A11 erotus, B11 ja B12 summa)
        Apumatriisi7 = multiplyNaive(A12 ja A22 erotus, B21 ja B22 summa)
    Muuten
        //Luodaan Apumatriisit 1-7 kertomalla aikaisemmin luotuja neljäsmatriiseja
        //strassen metodilla
        Apumatriisi1 = multiplyStrassen(A11 ja A22 summa, B11 ja B22 summa)
        Apumatriisi2 = multiplyStrassen(A21 ja A22 summa, B11)
        Apumatriisi3 = mutliplyStrassen(A11, B12 ja B22 erotus)
        Apumatriisi4 = multiplyStrassen(A22, B21 ja B11 erotus)
        Apumatriisi5 = multiplyStrassen(A11 ja A12 summa, B22)
        Apumatriisi6 = multiplyStrassen(A21 ja A11 erotus, B11 ja B12 summa)
        Apumatriisi7 = multiplyStrassen(A12 ja A22 erotus, B21 ja B22 summa)
    //Lasketaan tulosmatriisin neljäsmatriisit apumatriisien avulla
    C11 = Apumatriisi7 + ((Apumatriisi1 + Apumatriisi4) – Apumatriisi5)
    C12 = Apumatriisi3 + Apumatriisi5
    C21 = Apumatriisi2 + Apumatriisi4
    C22 = Apumatriisi6 + (Apumatriisi3 + (Apumatriisi1 – Apumatriisi2))
    MatriisiC = Yhdistetään matriisit C11, C12, C21, C22
    Palauta MatriisiC

determinant (Matriisi)
    Jos Matriisi ei ole neliö, palauta virhe
    MatriisinKoko = Matriisi.Rivimäärä
    Jos MatriisinKoko = 1
        Palauta Matriisi[1][1]
    Muuten Jos MatriisinKoko = 2
        Palauta (Matriisi[1][1]*Matriisi[2][2]) – (Matriisi[1][2]*Matriisi[2][1])
    Muuten jos MatriisinKoko = 3
        Palauta (Matriisi[1][1]*Matriisi[2][2]*Matriisi[3][3]) +
            (Matriisi[1][2]*Matriisi[2][3]*Matriisi[3][1]) +
            (Matriisi[1][3]*Matriisi[2][1]*Matriisi[3][2]) -
            (Matriisi[1][3]*Matriisi[2][2]*Matriisi[3][1]) -
            (Matriisi[1][2]*Matriisi[2][1]*Matriisi[3][3]) -
            (Matriisi[1][1]*Matriisi[2][3]*Matriisi[3][2])
    Muuten
        //Lasketaan matriisin LU-hajotelma Doolittlen metodilla
        DeterminantinEtumerkki = 1
```

```
                For i = 1, i <= MatriisinKoko, i++
                        For j = i, j <= MatriisinKoko, j++
                                For k = 0, k <= i – 1, k++
                                        Matriisi[i][j] = Matriisi[i][j] -
                                                (Matriisi[i][k]*Matriisi[k][j])
                        For j = i+1, j <= MatriisinKoko, j++
                                For k = 1, k <= i-1, k++
                                        Matriisi[j][i] = Matriisi[j][i] -
                                                (Matriisi[j][k] *Matriisi[k][i])
                                Matriisi[j][i] = Matriisi[j][i] / Matriisi[i][i]
                                Jos edellisellä rivillä tapahtuu nollalla kertominen ja tulos on
                                        NaN, palautetaan 0
                        //Käyttämällä Partial Pivot menetelmää minimoidaan nollalla jakamisen
                        todennäköisyys
                        For Rivi = i+1, Rivi <= MatriisinKoko, Rivi++
                                PivotRivi = i
                                Jos Matriisi[Rivi][i] > Matriisi[i][i]
                                        PivotRivi = Rivi
                                Jos PivotRivi != i
                                        For Sarake = 1, Sarake <= MatriisinKoko, Sarake++
                                                Temp = Matriisi[i][Sarake]
                                                Matriisi[i][Sarake] = Matriisi[PivotRow][Sarake]
                                                Matriisi[PivotRow][Sarake] = Temp
                                        //Jos kahden rivin paikat vaihdetaan, muutetaan
                                        determinantin etumerkkiä
                                        DeterminantinEtumerkki *= -1
        Determinantti = Matriisi[1][1]
        For i = 1, i <= MatriisinKoko, i++
                Determinantti *= Matriisi[i][i]
        Palauta Determinantti*DeterminantinEtumerkki

strassenInvert(Matrix)
        //Huomioitava että matriisin oikeellisuus (neliö muoto) ja yhden pituisen matriisin
                tapaukset on suoritetty metodissa invertMatrix
        MatriisinKoko = Matrix.Rivimäärä
        Puolipiste = MatriisinKoko / 2
        Jos MatriisinKoko <= 2
                Palauta naiveInvert(Matrix)
        Muuten
                Luo neljäsosamatriisit A11, A12, A21, A22
                Kopioi Matrixin sisältö neljäsosamatriiseihin
                //Kutsutaan strassenInvert metodia rekursiivisesti ylävasempaan
                        neljäsosamatriisiin
                A11 = strassenInvert(A11)
                //Lasketaan tulomatriisin neljännekset A11:n käänteismatriisin ja
                        strassenInvert ja multiplyStrassen metodien avulla
                Luo tulomatriisin neljännekset C11,C12,C21,C22
                C22 = strassenInvert( A22 – (A21*A11*A12))
                C11 = A11 + (A11*A12*C22*A21*A11)
                C12 = -A11*A12*C22
                C21 = -C22*A21*A11
                Luodaan tulosmatriisi Ret
```

Kopioidaan neljännesmatriisit C11, C12, C21, C22 matriisiin Ret
Palauta Ret