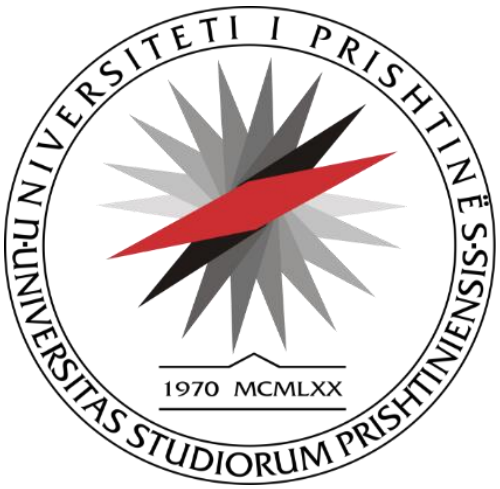
UNIVERSITETI I PRISHTINËS

FAKULTETI I SHKENCAVE MATEMATIKORE DHE NATYRORE

DEPARTAMENTI I MATEMATIKËS



LËNDA: Analizë e algoritmeve

TEMA: Algoritmet për shumëzim të matricave

Profesori:

Elver Bajrami

Besnik Duriqi

Studenti:

Erlisa Lokaj

Ermira Haziri

Përmbajtja

Hyrje

Për shkak të aplikimeve të shumta llogaritëse të shumëzimit të matricës, kërkimi në algoritme efikase për shumëzimin e matricave mund të çojë në përmirësime të gjera të performancës. Në këtë punim kemi qëllim të tregojmë dhe ilustrojmë ndryshimet ndërmjet tre algoritmeve të ndryshëm mirëpo me qëllim të njejtë. Ky qëllim është realizimi i procesit të shumëzimit të matricave në mënyrë sa më korrekte dhe efektive. Këto algoritme janë: Naïve Algorithm , Divide and Conquer algorithm dhe Strassen Algorithm. Secili prej tyre përmban karakteristika unike dhe kompleksitete për realizimin e kësaj detyre të cilat do të përshkruhen në seksionet në vijim. Implementimi i këtyre algoritmeve është realizuar në gjuhën programuese Java e cila na ndihmon të evidentojmë ndryshimin në kohë për secilin algoritëm. Rezultatet tona tregojnë se ndërsa metoda e Strassen përmirëson kompleksitetin teorik të shumëzimit të matricës, ka një numër konsideratash praktike që duhen adresuar që kjo të rezultojë në përmirësime në kohën e ekzekutimit.

Kompleksiteti llogaritës

Kur shumëzojmë dy matrica 2×2, matrica që rezulton është e barabartë me:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} \times b_{11} + a_{12} \times b_{21} & a_{11} \times b_{12} + a_{12} \times b_{22} \\ a_{21} \times b_{11} + a_{22} \times b_{21} & a_{21} \times b_{12} + a_{22} \times b_{22} \end{pmatrix}$$

Shohim që kërkohen $8 = 2^3$ shumëzime dhe $4 = 2^2$ shuma për të gjetur këtë matricë. Duke përgjithësuar për dy matrica, ne gjejmë se llogaritja e matricës që rezulton kërkon n^3 shumëzime dhe $(n-1)n^2$ shuma. Kjo do të thotë se ndërsa rritet n, numri i veprimeve aritmetike të nevojshme për të gjetur produktin e dy matricave $n \times n$ rritet me një shpejtësi shumë më të lartë se madhësia e matricave. Raporti midis hyrjes së një algoritmi dhe burimeve që ai kërkon për të gjetur një zgjidhje quhet kompleksiteti llogaritës i algoritmit. Siç e pamë më lart, në rastin e shumëzimit të matricës, kompleksiteti llogaritës aritmetik është i barabartë me $n^3 + (n-1)n^2 = 2n^3 - n^2$. Sidoqoftë, kjo qasje "naive" nuk është optimale. Në këtë punim ne do të shqyrtojmë një algoritëm për shumëzimin e matricës me kompleksitet më të ulët llogaritës dhe do të kryejmë një eksperiment duke krahasuar dy qasjet.

Kompleksiteti kohor

Për shkak se koha e ekzekutimit të një algoritmi mund të ndryshojë shumë nga makina në makinë, kompleksiteti kohor nuk mund të shprehet thjesht si koha absolute e ekzekutimit të algoritmit në sekonda ose minuta. Në vend që të përdorim kohën e ekzekutimit, ne shikojmë numrin e "hapave" që merr një algoritëm, për të shprehur kompleksitetin e kohës. Këtu, një hap përfaqëson ekzekutimin e një operacioni, koha e ekzekutimit të cilit nuk ndikohet nga madhësia e hyrjes. Ky nocion i kompleksitetit kohor është universal për të gjitha makinat, pasi numri i hapave të nevojshëm për të ekzekutuar algoritmin do të jetë i njëjtë për çdo makinë.

Përshkrimi i algoritmeve

Në matematikë, veçanërisht në algjebër lineare, shumëzimi i matricës është një operacion binar që prodhon një matricë nga dy matrica. Për shumëzimin e matricës, numri i kolonave në matricën e parë duhet të jetë i barabartë me numrin e rreshtave në matricën e dytë. Matrica që rezulton, e njohur si produkti i matricës, ka numrin e rreshtave të matricës së parë dhe numrin e kolonave të matricës së dytë.

$$\begin{vmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{vmatrix} \times \begin{vmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{vmatrix} = \begin{vmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{vmatrix}$$

Algoritmi Naïve

function matrixMultiply(A,B):

Supozojmë që dimensioni i A është $(m \times n)$, dimensioni i B është $(p \times q)$

Fillimi

Nëse n nuk është i njëjtë me p, atëherë Exit

Përndryshe defino C si $(m \times q)$

for i në 0 deri m-1, do

for j në 0 deri q-1, do

for k në 0 deri p, do

$C[i,j] = C[i,j] + (A[i,k] \times A[k,j])$

done

done

done

Fundi

Kompleksiteti kohor i këtij algoritmi shihet se është $T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} O(1) = O(n^3)$.

Algoritmi Divide&Conquer

function MatrixMultiplicationDAQ (A, B)

$n \leftarrow \dim(A)$

$C \leftarrow new\ n \times n\ matrix$

if n==1 **then**

$C \leftarrow (a_{11} \times b_{11})$

return C

else ndaje A dhe B në katër $\frac{n}{2} \times \frac{n}{2}$ kuadrante

$C_{11} \leftarrow MatrixMultiplicationDAQ\ (A_{11}, B_{11}) + MatrixMultiplicationDAQ\ (A_{12}, B_{21})$

$C_{12} \leftarrow MatrixMultiplicationDAQ\ (A_{11}, B_{12}) + MatrixMultiplicationDAQ\ (A_{12}, B_{22})$

$C_{21} \leftarrow MatrixMultiplicationDAQ\ (A_{21}, B_{11}) + MatrixMultiplicationDAQ\ (A_{22}, B_{21})$

$C_{22} \leftarrow MatrixMultiplicationDAQ\ (A_{21}, B_{12}) + MatrixMultiplicationDAQ\ (A_{22}, B_{22})$

$C \leftarrow \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

return C

Kompleksiteti kohor i këtij algoritmi shihet se është $T = 8T\left(\frac{n}{2}\right) + O(n\text{^}2)$.

Algoritmi Strassen

function Strassen(A,B)

n ← dim(A)
C ← një matricë n×n
if n == 1 **then**
  C ← (a_{11}· b_{11})
  return C
else ndaje A dhe B në katër $(\frac{n}{2}) \times (\frac{n}{2})$ kuadrante
M_1 ← Strassen (A_{11}+A_{22}, B_{11}+B_{22})
M_2 ← Strassen (A_{21}+A_{22}, B_{11})
M_3 ← Strassen (A_{11}, B_{12} - B_{22})
M_4 ← Strassen (A_{22}, B_{21} – B_{11})
M_5 ← Strassen (A_{11}+A_{12}, B_{22})
M_6 ← Strassen (A_{21} -A_{11}, B_{11}+B_{12})
M_7 ← Strassen (A_{12} - A_{22}, B_{21}+B_{22})
C_{11} ← M_1 +M_4 – M_5 +M_7
C_{12} ← M_3 +M_5
C_{21} ← M_2 +M_4
C_{22} ← M_1 – M_2 + M_3 +M_6
  C ← $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$
return C

Kompleksiteti kohor i këtij algoritmi shihet se është $T = 7T\left(\frac{n}{2}\right) + O(n^2)$ *rezulton në* $O(N^{\log 7})$ *që përafërsisht është* $O(N^{2.8074})$.

Rezultatet

Për pjesën e mbetur të kësaj detyre, ne kryejmë disa testime për të përcaktuar se si ndryshimi në kompleksitetin midis qasjes së Strassen, DAQ dhe algoritmit naiv ndikon në performancën praktike të shumëzimit të matricës llogaritëse. Ne e bëjmë këtë duke zbatuar të tre algoritmet dhe duke krahasuar kohëzgjatjen e tyre përkatëse për hyrje të madhësive të ndryshme.

Pjesa 1 e testimeve

Tabela e mëposhtme tregon kohën mesatare të ekzekutimit në milisekonda të secilit algoritëm për dimensione të ndryshme të matricave.

Algoritmi	N=4	N=8	N=16	N=32	N=64
Naïve(ms)	0.1258	0.49445	1.35125	2.866958	5.221709
DAQ (ms)	0.0681	0.40233	2.033834	11.15866	92.543
Strassen (ms)	0.0560	0.39395	1.013167	10.01791	66.11175

Vërejmë që për dimensionet 4,8, dhe 16 algoritmi Strassen ka arritur një kohë mesatare të ekzekutimit më të mirë krahasuar me algoritmin Naive dhe DAQ. Mirëpo ky rezultat nuk vazhdoj për dimensionet e matricave 32 dhe 64. Për këto dimensione shohim që algoritmi i thjeshtë Naive arrin rezultatet më të mira. Për ekzekutimin e matricave me dimensione më të vogla algoritmi Naive na jep kohën më të gjatë të ekzekutimit ndërsa kur dimensioni arrin 32 dhe 64 arrin një performancë të mirë krahasuar me dy algoritmet e tjera të implementuara. Algoritmi DAQ ka një

qëndrueshmëri me kohë të gjatë pothuajse në të gjitha dimensionet përveç kur n=4 dhe n=6 që shfaq rezultate më të mira se Naive. Ndërsa algoritmi Strassen dhuroj mjaft premtime në dimensione të vogla mirëpo ndërsa rritej numri i dimensioneve gjithashtu rritej edhe koha e tij e ekzekutimit. Duke pasur parasysh kompleksitetin kohor të algoritmit Strassen pritej të fitohej një rezultat më i mirë sesa që arritëm. Vlen të përmendet që vërehet impakti i numrit të reduktuar të rekurzioneve të algoritmit Strassen krahasur me DAQ nga tabela e mësipërme.

Pjesa 2 e testimeve

Siç u përmend më lartë pritej që të fitohen rezultate më të mira sa i përket algoritmit Strassen. Mirëpo kjo ndodh vetëm në testimet praktike të cilat mund t'i realizojmë ndërsa në dimensione më të larta parashikohet që Strassen përmison kohën mesatare të ekzekutimit të tij.
Si lloj optimizimi është menduar të implementohet një algoritëm hibrid i cili kombinon karakteristikat e algoritmit Strassen dhe algoritmit naiv me shpresën e rezultateve më të mira. Testimet në këtë pjesë shihen në tabelën e mëposhtme:

Algoritmi	N=128	N=256	N=512	N=1024	N=2048
Naive (ms)	25.36	215.37	1851.99	16136.2	180450.1
Hstrassen(ms)	24.23	184.21	1322.68	9007.6	61915.8

Përndryshim nga pjesa 1 në këtë pjesë kemi arritur rezultate më të dëshiruara për algoritmin Strassen sa i përket kohës mesatare të ekzekutimit. Kjo pjesë e testimeve është realizuar vetëm për algoritmin Naive dhe Strassen variant pasi që nga pjesa e kaluar arritëm në përfundim që algoritmi DAQ nuk shfaq ndonjë përmisim kohor ndaj algoritmëve tjerë.

Diskutime

Ndërsa metoda inovative e Strassen për shumëzimin e matricës përmirëson kompleksitetin asimptotik të shumëzimit naiv të matricës, rezultatet tona tregojnë se ajo është e kufizuar në prakticitetin e saj, duke përmirësuar vetëm performancën për matricat përtej përdorimit praktik. Për të përfituar nga kompleksiteti i tij më i ulët, zbatimet praktike të algoritmit Strassen ndalojnë rekursionin herët, duke iu drejtuar algoritmit naiv për shumëzimin e nënmatricave. Rezultatet tona tregojnë se duke përdorur këtë variant hibrid, ndryshimi i vogël në kompleksitet midis algoritmit Strassen dhe qasjes naive mund të sigurojë përmirësime të rëndësishme në kohën e ekzekutimit.

Përfundim

Analiza dhe krahasimi i algoritmeve rezulton të jetë një çështje e ndërlikuar. Analiza asimptotike dhe shënimi big-O ofrojnë një nocion universal të kompleksitetit, duke e bërë më të lehtë përcaktimin dhe krahasimin e kompleksitetit të algoritmeve. Megjithatë, kjo nuk është një masë përfundimtare, pasi shpesh ka faktorë të tjerë që ndikojnë në performancën praktike të një algoritmi. Kjo detyrë e ilustron këtë, duke ekzaminuar algoritmin Strassen për shumëzimin e matricës dhe duke e krahasuar atë me shumëzimin e matricës naive. Ndërsa metoda e Strassen përmirëson kompleksitetin teorik të shumëzimit të matricës, zbatimet duhet të marrin parasysh një numër konsideratash praktike për të përfituar nga ky kompleksitet i reduktuar.