Aplikasi Bahasa C dalam Perkalian Matriks

Bayu Aji Nugroho, Muhammad Fauzan, Deovie Lentera
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
(13221601, 13220009, 18320037) Email: @std.stei.itb.ac.id

***Abstract*—Perkalian matriks berukuran besar dapat dilakukan dengan bantuan program yang dikembangkan dari pemrograman bahasa C. Ada beberapa algoritma yang dapat digunakan dalam menghitung perkalian matriks diantaranya algoritma strassen, iterative, dan rekursif. Algoritma tersebut tentunya memiliki karakteristik yang berbeda-beda. Algoritma yang baik adalah algoritma yang mangkus atau efisien. Kemangkusan algoritma diukur dari berapa jumlah waktu dan ruang (space) memori yang dibutuhkan untuk menjalankannya. Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan jumlah data yang diproses. Untuk mengetahui kemangkusan algoritma tersebut akan digunakan analisa kompleksitas waktu dan ruang. Dalam percobaan ini ada 4 algoritma program yang digunakan untuk melakukan perkalian matriks yaitu algoritma strassen, iterative, rekursif, dan divide and conquer. Dari hasil percobaan didapatkan .... Dari hasil percobaan tersebut, algoritma .... Adalah algoritma yang paling mangkus atau efisien dari empat algortima yang lain.**

***Index Terms*—Strassen, Iteratif, Rekursif**

I. PENDAHULUAN

Perkalian matriks merupakan perkalian antara 2 buah matriks persegi yang mempunyai ukuran yang sama untuk menghasilkan sebuah matriks baru. Perkalian matriks dengan ukuran yang kecil sangat mudah dilakukan dengan perhitungan manual, namun akan sangat sulit apabila matriks memiliki ukuran yang besar. Untuk memudahkan perkalian matriks yang berukuran besar, maka dapat dilakukan perkalian dengan bantuan program yang dikembangkan dari pemrograman bahasa C.

Ada beberapa algoritma yang dapat digunakan dalam menghitung perkalian matriks diantaranya yaitu algortima strassen, iterative, dan rekursif. Algoritma tersebut tentunya memiliki karakteristik yang berbeda-beda. Algoritma yang baik adalah algoritma yang memiliki efisiensi yang baik ketika digunakan untuk menghitung data yang besar. Dalam percobaan ini kita akan membandingkan tingkat efisiensi dari masing-masing algoritma.

Algoritma yang baik adalah algoritma yang mangkus atau efisien. Untuk menentukan tingkat kemangkusan algoritma, diperlukan metode yang dapat digunakan sebagai dasar analisa. Selain itu juga diperlukan parameter khusus yang dapat dijadikan tolak ukur agar algoritma dapat dikatakan mangkus.

Tujuan dari percobaan ini adalah menentukan tingkat kemangkusan dari masing-masing algoritma sehingga dapat diketahui algoritma yang terbaik untuk melakukan perkalian matrikas dengan ukuran yang besar.

Metode analisa yang digunakan dalam percobaan ini adalah analisa kompleksitas waktu dan ruang. Ada 4 algoritma yang di analisa yaitu algortima strassen, iterative, rekursif, dan divide and conquer.

II. STUDI AAAAPUSTAKA

A. Bahasa Pemrograman C

Bahasa pemrograman adalah suatu bahasa yang hanya dapat dimengerti oleh komputer. Komputer tidak akan jalan tanpa perintah dari kita. Dengan perintah yang kita berikan kepada komputer tersebut, maka komputer tersebut akan membacanya dan memberikan output yang kita inginkan. Banyak bahasa pemrograman yang kita akan temui dan pelajari (sebut saja bahasa C, Cplus-plus, Java, dan lain-lainnya) pada saat kita ingin membuat suatu program. Namun, untuk artikel ini, kita akan berfokus terhadap bahasa yang sering kalian temui sebagai pemula, yaitu bahasa C. Bahasa C adalah bahasa pemrograman prosedural yang dapat digunakan untuk membangun software seperti operating system, database, dan lainnya. Bahasa ini diciptakan oleh Dennis Ritchie untuk menciptakan aplikasi sistem yang dapat berinteraksi dengan hardware secara langsung. Bahasa ini juga mempunyai beberapa fakta yang menarik seperti menjadi penerus bahasa B, menjadi bahasa yang menciptakan operating system yang bernama UNIX, dan telah diformalkan oleh American National Standard Institute (ANSI) pada tahun 1988. Bahasa C tentunya adalah bahasa yang dapat dijadikan sebagai bahasa pemrograman pertama bagi pemula. Namun, perlu kalian ketahui bahwa bahasa C juga dikenal sebagai mother language, system programming language, procedure-oriented programming language, structured programming language, dan mid-level programming language. Berikut penjelasannya. Bahasa C dikenal sebagai mother language karena sebagian besar compiler, kernel, dan lainnya dicatat dalam bahasa ini dan beberapa bahasa pemrograman lainnya mengikuti syntax bahasa ini seperti C++, Java, dan lainnya. Bahasa C sebagai system programming language dapat digunakan untuk melakukan low-level programming. Bahasa C sebagai procedural language menentukan beberapa langkah untuk program agar dapat menyelesaikan masalah. Bahasa C sebagai structured procedural language berarti bahasa ini dapat memecahkan sebuah program menjadi bagian-bagian sehingga dapat dimengerti dengan mudah. Bahasa C sebagai mid-level programming language mendukung kedua low-level dan high-level language. Tentunya, bahasa C dapat digunakan untuk kehidupan sehari-hari kita karena bahasa ini menghasilkan kode yang berjalan hampir secepat kode yang ditulis dalam assembly language. Contoh penggunaan bahasa C antara lain Operating Systems, Language Compilers, Text Editors, Network Drivers, Databases, dan Utilities.

B. *Matriks*

Matriks adalah kumpulan bilangan yang disusun secara baris atau kolom atau kedua-duanya dan di dalam suatu tanda kurung. Bilangan-bilangan yang membentuk suatu matriks disebut sebagai elemen-elemen matriks. Matriks digunakan untuk menyederhanakan penyampaian data, sehingga mudah untuk diolah.

1) ukuran matriks: Ukuran matriks ditentukan oleh jumlah baris dan kolom yang dikandungnya. Matriks dengan kolom m dan n baris disebut matriks m x n atau matriks ”m kali n”, dimana m dan n disebut dimensinya. Sebagai contoh, matriks A di bawah adalah matriks 3 x 2. Matriks dengan satu baris disebut vektor baris, dan matriks dengan satu kolom disebut vektor kolom. Matriks dengan jumlah baris dan kolom yang sama disebut matriks persegi. Matriks dengan jumlah baris atau kolom yang tak terbatas (atau keduanya) disebut matriks tak terbatas. Dalam beberapa konteks, akan bermanfaat untuk mempertimbangkan sebuah matriks tanpa baris atau tanpa kolom, yang disebut matriks kosong.

$$\mathbf{A} = \begin{bmatrix} -1.3 & 0.6 \\ 20.4 & 5.5 \\ 9.7 & -6.2 \end{bmatrix}.$$

Fig. 1. matriks 3x2

2) Perkalian Matriks: Perkalian dua matriks ini bisa dilakukan ketika jumlah kolom A dan jumlah baris B sama. Perkalian matriks tersebut akan menghasilkan suatu matriks dengan jumlah baris yang sama antara matriks A dan B. Syarat dua buah matriks bisa dikalikan jika mempunyai jumlah kolom matriks pertama sama dengan jumlah baris matriks ke dua. Adapun ordo matriks hasil perkalian dua matriks adalah jumlah baris pertama dikali jumlah kolom ke dua. Misalnya matriks P memiliki jumlah kolom sebanyak a dan jumlah baris c. Sedangkan matriks Q memiliki jumlah kolom sebanyak c dan jumlah baris a. Hasil perkalian P dan Q adalah matriks R dengan jumlah kolom a dan jumlah baris d. Perkalian dua buah matriks dapat dilihat pada contoh di bawah ini:

III. Metodologi Penelitian

Penelitian dilakukan dengan membuat program perkalian dengan dimensi matriks yang kecil yaitu ukuran 3x3. Selanjutnya dilakukan perhitungan manual perkalian matriks ukuran 3x3. Dimasukkan dengan angka yang sama antara perhitungan program dan perhitungan manual. Selanjutnya diamati hasil dari kedua perhitungan tersebut. Setelah hasil benar, kemudian

Nama	Ukuran	Contoh
Vektor baris	$1 \times n$	$\begin{bmatrix} 3 & 7 & 2 \end{bmatrix}$
Vektor kolom	$n \times 1$	$\begin{bmatrix} 4 \\ 1 \\ 8 \end{bmatrix}$
Matriks persegi	$n \times n$	$\begin{bmatrix} 9 & 13 & 5 \\ 1 & 11 & 7 \\ 2 & 6 & 3 \end{bmatrix}$

Fig. 2. deskripsi ukuran matriks

dikembangkan perhitungan untuk ukuran 10x10, 100x100, dan 1000x1000.

IV. Implementasi dan Pengujian

A. *Flowchart algoritma program*

program memiliki flowchart sebagai berikut:

B. *Implementasi Algoritma Iterative dalam Perkalian Matriks*

Perkalian matriks pada program C dapat dilakukan menggunakan algortima iterative. Prosedur perkalian matriks menggunakan metode ini memerlukan argumen atau masukan berupa `size` atau ukuran dari matriks, lalu matriks A dan matriks B yang akan dicari hasil perkaliannya. Algoritma ini dapat diimplementasikan dalam pseudocode berikut:

C. *Implementasi Algoritma Rekursif dalam Perkalian Matriks*

Dalam perkalian matriks rekursif, kita akan menerapkan tiga loop iterasi melalui panggilan rekursif. Panggilan rekusif pertama dari kaliMatriksRekursif adalah untuk mengulangi k (kolom 1 atau baris 2). Panggilan rekursif kedua dari

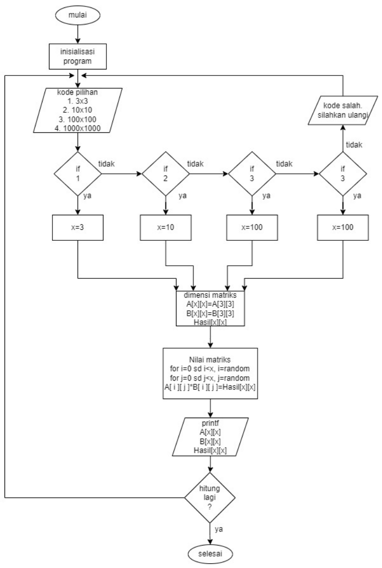


Fig. 3. flowchart program perkalian matriks

Algorithm 1: Perkalian Matriks Iterative `kaliMatriksIterative`

Result: Mengalikan 2 matriks menggunakan algoritma iterasi

procedure *kaliMatriksIterative(int size, int mat_A[size][size], int mat_B[size][size])*

Initialization:

`int` $hasil[i][j] = 0$

Algorithm:

for $i < size$ **do**
  for $j < size$ **do**
    for $k < size$ **do**
      $hasil[i][j] += mat\_A[i][k] * mat\_B[k][j]$

kaliMatriksRekursif adalah untuk mengubah kolom. Panggilan rekursif ketiga adalah untuk mengubah baris. Berikut pseudocode algoritma rekursif pada perkalian matriks:

D. Implementasi Algoritma Strassen's dalam Perkalian Matriks

Algortima Strassens dapat digunakan dalam operasi perkalian matriks. Ide yang digunakan oleh algoritma ini adalah membagi matriks A dan matriks B menjadi 8 submatriks lalu menghitung sub-matriks C dengan operasi yang sudah ditentukan pada studi pustaka secara rekursif. Perlu diperhatikan bahwa, algoritma ini hanya bisa digunakan pada matriks berukuran m $\times$ m dengan m merupakan bilangan kelipatan pangkat 2. Berikut pseudocode algoritma Strassens pada perkalian matriks:

E. Algoritma Display untuk Menampilkan Matriks

Algoritma ini digunakan untuk menampilkan matriks yang akan digunakan pada proses perkalian matriks, yaitu matriks

Algorithm 2: Perkalian Matriks Recursive `kaliMatriksRekursif`

Result: Mengalikan 2 matriks menggunakan algoritma rekursif

procedure *kaliMatriksRecursive(int size, int mat_A[size][size], int mat_B[size][size], int hasil[size][size])*

Initialization:

`int` $row1 = size$
`int` $col1 = size$
`int` $row2 = size$
`int` $col2 = size$
`static int` $i, j, k = 0$

Algorithm:

if $i \geq row1$ **then**
  $return$
else if $i < row1$ **then**
  if $j < col2$ **then**
    if $k < col1$ **then**
      $hasil[i][j] += mat_A[i][k] * mat_B[k][j]$
      $k++$
      $kaliMatriksRecursive(col1, mat\_A, mat\_B, hasil)$
    $k = 0$
    $j++$
    $kaliMatriksRecursive(col1, mat\_A, mat\_B, hasil)$
  $j = 0$
  $i++$
  $kaliMatriksRecursive(col1, mat\_A, mat\_B, hasil)$

A, matriks B, maupun matriks hasil perkalian matriks A dan B. Algoritma ini digunakan karena kita akan banyak menampilkan matriks pada layar. Oleh karena itu dibutuhkan suatu prosedur untuk menampilkan matriks secara efektif dan efisien. Algoritma ini memiliki pseudocode:

F. Analisis Penggantian Pengaksesan Matriks

Pengaksesan matriks biasanya menggunakan algoritma "For" lalu meng-*traverse* elemen matriks satu persatu. Pengakesan matriks biasanya dilakukan pertama kali pada baris, yang secara umum digunakan "i", lalu dilanjutkan dengan pengaksesan kolom, yang secara umum digunakan "j". Berikut contoh pengaksesan matriks dengan cara di atas menggunakan algoritma display yang telah dijelaskan di atas:

Jika kita membalikkan susunan algoritma "For" dalam pengaksesan matriks, di mana kita mengakses terlebih dahulu "j" lalu "i", maka akan didapatkan output:

Berdasarkan dua gambar di atas, dapat ditarik kesimpulan bahwa pembalikan variabel dalam pengaksesan matriks tidak membuat output yang dihasilkan berubah. Hal ini dikarenakan pengulangan "i" dan "j" memiliki batas iterasi yang sama, yaitu "size". Selain itu, iterasi yang dilakukan juga sama, yaitu

Algorithm 3: Perkalian Matriks Strassen's `kaliMatriksStrassens`

Result: Mengalikan 2 matriks menggunakan algoritma Strassens's

procedure *kaliMatriksStrassens(int mat_A[2][2], int mat_B[2][2])*

Initialization:

`int` i, j

`int` $hasil[i][j]$

`int` $row1, row2, m3, m4, m5,$

Algorithm:

$row1 = (mat_A[0][0] + mat_A[1][1]) * (mat_B[0][0] + mat_B[1][1])$

$row2 = (mat_A[1][0] + mat_A[1][1]) * mat_B[0][0]$

$m3 = mat_A[0][0] * (mat_B[0][1] - mat_B[1][1])$

$m4 = mat_A[1][1] * (mat_B[1][0] - mat_B[0][0])$

$m5 = (mat_A[0][0] + mat_A[0][1]) * mat_B[1][1]$

$m6 = (mat_A[1][0] - mat_A[0][0]) * (mat_B[0][0] + mat_B[0][1])$

$m7 = (mat_A[0][1] - mat_A[1][1]) * (mat_B[1][0] + mat_B[1][1])$

$hasil[0][0] = row1 + m4 - m5 + m7$

$hasil[0][1] = m3 + m5$

$hasil[0][1] = m3 + m5$

$hasil[1][1] = row1 - row2 + m3 + m6$

Algorithm 4: Display `display`

Result: Menampilkan matriks pada layar

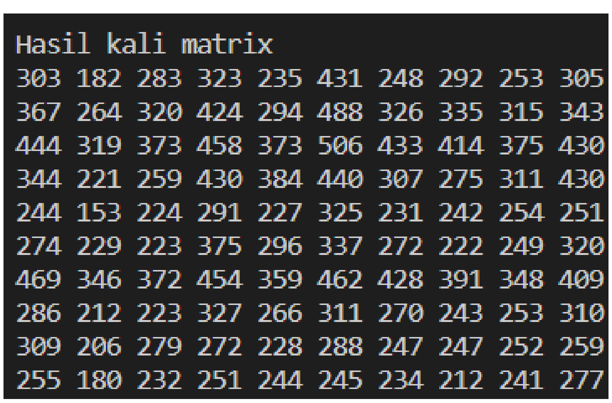
procedure *display(int size, int hasil[size][size]*

Initialization:

`int` i, j

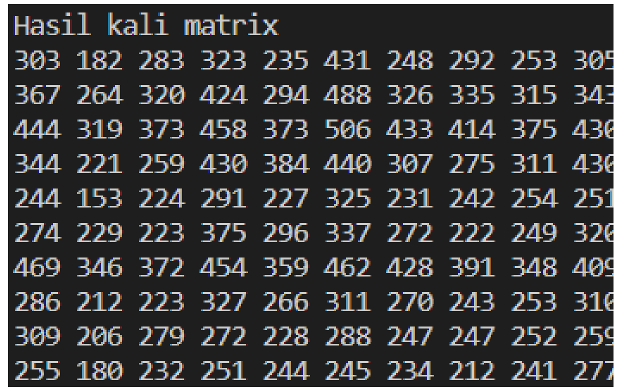
Algorithm:

for $i < size$ **do**
- **for** $j < size$ **do**
  - $print(hasil[i][j])$
  - $j++$
- $i++$



Hasil kali matrix
303 182 283 323 235 431 248 292 253 305
367 264 320 424 294 488 326 335 315 343
444 319 373 458 373 506 433 414 375 430
344 221 259 430 384 440 307 275 311 430
244 153 224 291 227 325 231 242 254 251
274 229 223 375 296 337 272 222 249 320
469 346 372 454 359 462 428 391 348 409
286 212 223 327 266 311 270 243 253 310
309 206 279 272 228 288 247 247 252 259
255 180 232 251 244 245 234 212 241 277

Fig. 4. Akses Matriks 1



Hasil kali matrix
303 182 283 323 235 431 248 292 253 30
367 264 320 424 294 488 326 335 315 34
444 319 373 458 373 506 433 414 375 43
344 221 259 430 384 440 307 275 311 43
244 153 224 291 227 325 231 242 254 25
274 229 223 375 296 337 272 222 249 32
469 346 372 454 359 462 428 391 348 40
286 212 223 327 266 311 270 243 253 31
309 206 279 272 228 288 247 247 252 25
255 180 232 251 244 245 234 212 241 27

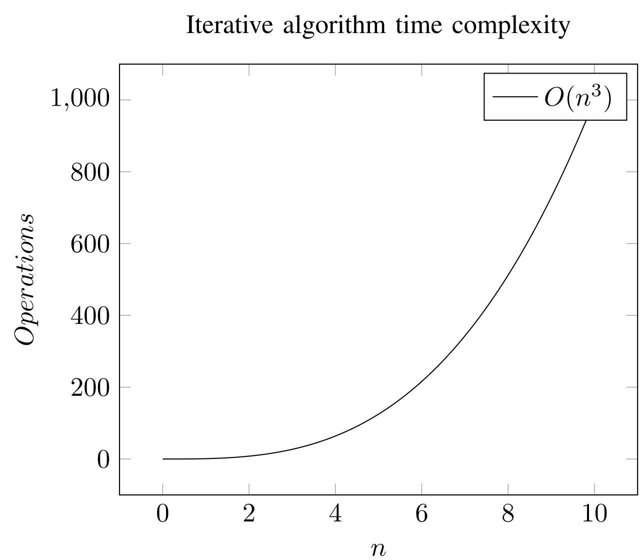
Fig. 5. Akses Matriks 2

dengan penambahan 1 per iterasi. Oleh karena itu, pembalikan variabel ini tidak mengubah output yang dihasilkan.

G. Perhitungan Kompleksitas waktu

1) Kompleksitas Waktu Algoritma Iterative:

Kompleksitas dari algoritma iterative memiliki notasi kompleksitas Big O adalah O(n^3). Hal tersebut dikarenakan terdapatnya 3 nested for loop, yang membuat notasi kompleksitas bernilai O(n^3).

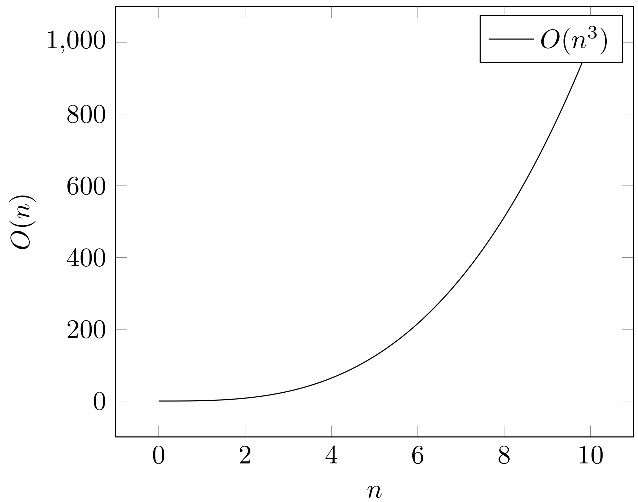


:

2) Kompleksitas Waktu Algoritma Rekursif:

Kompleksitas dari algoritma rekursif memiliki notasi kompleksitas Big O adalah O(n^3). Hal tersebut dikarenakan dipanggilnya fungsi rekursif sebanyak 8 kali, yang menyebabkan notasi bernilai O(n log 8), yang sama dengan O(n^3).

Recursive algorithm time complexity

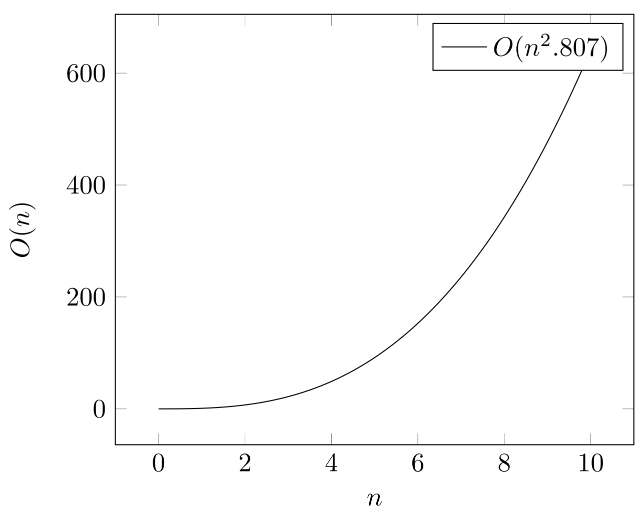


:

3) *Kompleksitas Waktu Algoritma Strassen:*

Kompleksitas dari algoritma Strassen memiliki notasi kompleksitas Big O bernilai O($n^{2.807}$). Hal tersebut dikarenakan, pada algoritma Strassen dipanggil fungsi rekursif sebanyak 7 kali, dibandingkan algoritma yang lain yang memanggil fungsi rekursif sebanyak 8 kali. Hal ini membuat algoritma Strassen memiliki time complexity yang lebih efisien (O($n^{2.807}$)) dibandingkan algoritma biasa (O(n^3))

Strassen's algorithm time complexity



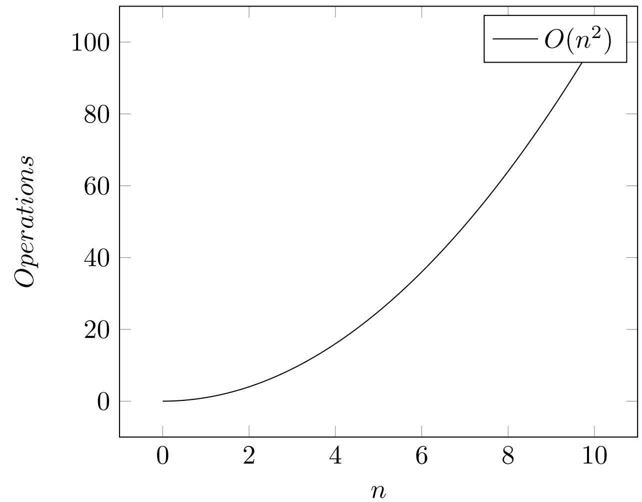
:

H. Perhitungan Kompleksitas Tempat

1) *Kompleksitas Tempat Algoritma Iterative:*

Kompleksitas tempat dari algoritma iterative memiliki notasi kompleksitas Big O bernilai O(n^2). Hal tersebut didapatkan dari variable-variable matrix $n \times n$ yang dipakai oleh algoritma.

Iterative algorithm space complexity

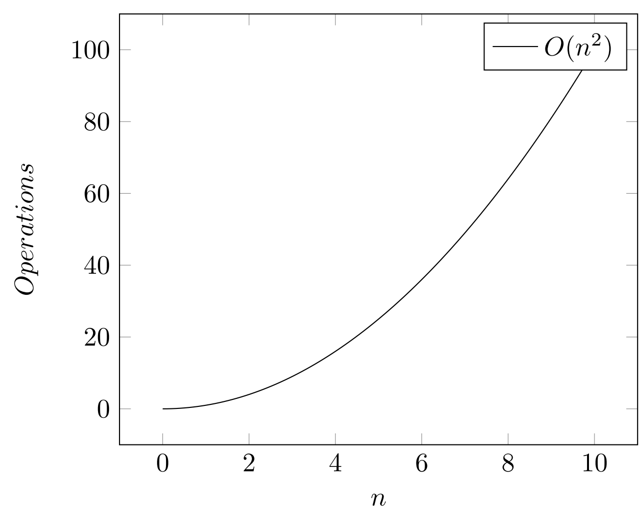


:

2) *Kompleksitas Tempat Algoritma Rekursif:*

Kompleksitas tempat dari algoritma rekursif memiliki notasi kompleksitas Big O bernilai O(n^2). Hal tersebut didapatkan dari variable-variable matrix $n \times n$ yang dipakai oleh algoritma.

Recursive algorithm space complexity

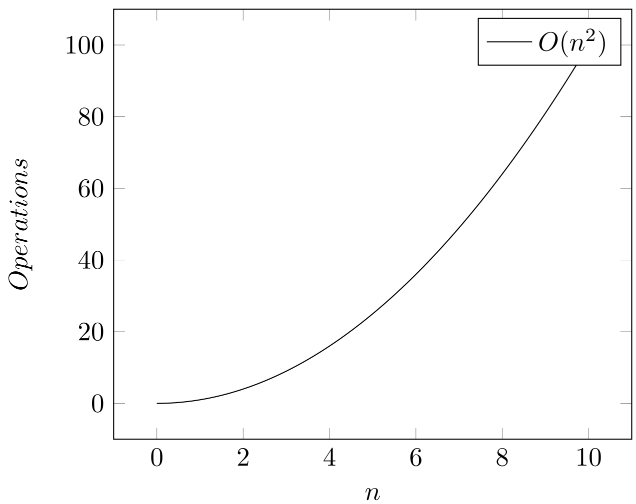


:

3) *Kompleksitas Tempat Algoritma Strassen:*

Kompleksitas tempat dari algoritma Strassen memiliki notasi kompleksitas Big O bernilai O(n^2). Hal tersebut didapatkan dari variable-variable matrix $n \times n$ yang dipakai oleh algoritma.

Strassen's algorithm space complexity



:

V. KESIMPULAN

Program berjalan dengan baik ketika dimensi atau ordo matriks berada dibawah ordo 50x50. Saat berada pada ordo diatas 50x50 perhitungan program berjalan akan tetapi tidak dapat menampilkan hasil secara keseluruhan. Kemudian pada ordo diatas 1000x1000 program tidak dapat berjalan.

REFERENCES

[1] K. H. Rosen, Discrete Mathematics and It's Applications - Seventh Edition. McGraw-Hill.Inc, 2012.

[2] Solichin, Achmad, Pemrograman Bahasa C dengan Turbo C, 2003.