IMPLEMENTASI METODE STRASSEN PADA MATRIKS JARANG BERORDE BESAR

Dian Wibowo Yeffry Handoko P. Nana Juhana

Jurusan Teknik Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia

Abstraksi

Faktor utama yang menentukan kualitas sebuah perangkat lunak adalah kecepatan proses dan memori yang dibutuhkan untuk pemrosesan. Keduanya bergantung pada algoritma dan struktur data yang digunakan. Dalam permasalahan perkalian matriks jarang, struktur data harus efisien dalam penyimpanan elemen, karena sebagian besar elemen matriks jarang adalah nol. Kami merancang struktur data untuk menyimpan matriks jarang secara efisien dengan hanya menyimpan elemen-elemen tidak nol. Metode Strassen adalah metode untuk mengalikan dua buah matriks berukuran 2×2, dengan metode ini jumlah proses perkalian yang dibutuhkan bisa dikurangi dari 8 proses perkalian menjadi 7 proses perkalian. Dengan metode ini kami mengajukan sebuah algoritma untuk mengalikan matriks jarang berorde besar dengan membagi-bagi sebuah matriks menjadi matriks terpartisi dengan submatriks-submatriks berorde 2×2 dan mengimplementasikan metode tersebut terhadap setiap perkalian submatriks.

I. Pendahuluan

Matriks jarang adalah matriks yang sebagian besar elemennya adalah nol. Matriks seperti ini banyak ditemukan dalam orde besar dan berkembang dari aplikasi nyata yang sudah diterapkan dilapangan [1]. Yang menjadi masalah adalah ruang penyimpanan yang dibutuhkan, jika matriks seperti ini disimpan dengan menggunakan array, akan terjadi pemborosan penyimpanan elemen-elemen bernilai nol, maka perlu dirancang struktur data yang lebih efisien dengan hanya mencatat elemen-

elemen tidak nol dan posisinya didalam matriks.

Selain struktur data, perlu dirancang algoritma yang lebih ringkas dalam penelusuran setiap elemen matriks, agar tidak ada prosedur yang terbuang sia-sia dengan mengalikan elemen nol. Selain itu dimungkinkan mengurangi jumlah perkalian antar elemen matriks dengan metode strassen [3], dengan metode ini jumlah perkalian yang dibutuhkan dalam perkalian dua buah matriks 2×2 dapat dikurangi. Perkalian dilakukan dengan memartisi matriks menjadi matriks terpartisi dengan submatriks-submatriks sebesar 2×2, lalu perkalian dilakukan antar submatriks dengan metode strassen.

Penelitian ini pada akhirnya akan menyimpulkan algoritma dan struktur data mana yang lebih sesuai untuk perkalian matriks jarang orde besar, dengan membatasi matriks-matriks yang digunakan adalah matriks bujursangkar berorde genap dari 1000×1000 hingga 5000×5000 dengan tingkat kejarangan berkisar antara 1% hingga 5%, dan dengan elemen-elemen terbatas antara 1 hingga 9.

II. Tinjauan Pustaka

Manipulasi matriks dapat dilakukan dengan memartisi matriks menjadi beberapa submatriks [2], dengan syarat-syarat tertentu operasi dapat dilakukan dengan menganggap submatriks seolah elemen matriks biasa.

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix}$$

dimana

$$[A1] = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$[A2] = \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix}$$

$$[A3] = \begin{bmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}$$

$$[A4] = \begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix}$$

adalah submatriks-submatriksnya.

$$C = A\begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix} \times B\begin{bmatrix} B1 & B2 \\ B3 & B4 \end{bmatrix}$$

$$C = \begin{bmatrix} (A1 \times B1) + (A2 \times B3) & (A1 \times B2) + (A2 \times B4) \\ (A3 \times B1) + (A4 \times B3) & (A3 \times B2) + (A4 \times B4) \end{bmatrix}$$

Metode Strassen adalah metode untuk mengalikan dua buah matriks berukuran 2×2. Jika dengan metode konvensional, operasi yang dibutuhkan adalah 8 perkalian dan 4 pertambahan, dengan metode ini operasi yang dibutuhkan menjadi 7 perkalian dan 18 pertambahan[3].

Perkalian matriks konvensional

$$C = A\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times B\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} (1 \times 5) + (2 \times 7) & (1 \times 6) + (2 \times 8) \\ (3 \times 5) + (4 \times 7) & (3 \times 6) + (4 \times 8) \end{bmatrix}$$

$$C = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Metode strassen

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$P_1 = (a_{12} - a_{22})(b_{21} - b_{22})$

$P_2 = (a_{11} + a_{22})(b_{11} + b_{12})$

$P_3 = (a_{11} - a_{21})(b_{11} + b_{12})$

$P_4 = (a_{11} + a_{12})b_{22}$

$P_5 = a_{11}(b_{12} - b_{22})$

$P_6 = a_{22}(b_{21} - b_{11})$

$P_7 = (a_{21} + a_{22})b_{11}$

$c_{11} = P_1 + P_2 - P_4 + P_6$

$c_{12} = P_4 + P_6$

$c_{21} = P_6 + P_7$

$c_{11} = P_2 – P_3 + P_5 – P_7$

Linked list adalah struktur data berupa rangkaian elemen saling berkait dimana setiap elemen terhubung dengan elemen lainnya melalui pointer. Pointer adalah alamat elemen pada memori. Penggunaan pointer pada elemen membuat elemen-elemen bersebelahan secara logis walaupun tidak secara fisik di memori.

Kelebihan struktur data linked list dengan pointer jika dibandingkan dengan array adalah, struktur data linked list dengan pointer lebih dinamis. Struktur data array pada inisialisasinya harus mengalokasikan sejumlah memori tertentu untuk menyimpan seluruh elemen data yang belum tentu digunakan, sedangkan struktur data linked list dengan pointer pada inisialisasinya hanya membutuhkan penunjukan terhadap nilai *nil* pada setiap pencatatan, alokasi memori untuk setiap simpul, dilakukan setiap penambahan simpul kedalam list.

III. Perancangan Perangkat lunak

Untuk melakukan analisis, sebuah perangkat lunak akan dibangun untuk mengambil kesimpulan perbandingan kecepatan proses dan perbandingan penggunaan memori, dalam analisis kecepatan proses antara algoritma perkalian matriks biasa dengan algoritma strassen akan diukur dalam satuan milidetik, sedangkan perbandingan penggunaan memori akan membandingkan penggunaan memori antara penyimpanan matriks jarang dengan linked list (*pointer*) dan array dalam satuan byte, pada masing-masing metode.

Metode strassen akan diimplementasikan terhadap matriks terpartisi, jadi akan dibuat dua buah struktur data, yang pertama untuk menampung matriks biasa dan yang kedua untuk matrks terpartisi. Keduanya akan dibuat dengan doubly linked list dengan alamat simpul akhir (last) tercatat dan ditambah dengan sentinel. Linked list dengan jenis tersebut dipilaih untuk memaksimalkan proses pencarian.

Gambar-gambar berikut menerangkan cara sebuah matriks disimpan dalam format terpartisi dan biasa. Ada dua jenis penyimpanan, yakni baris terkompresi dan kolom terkompresi.

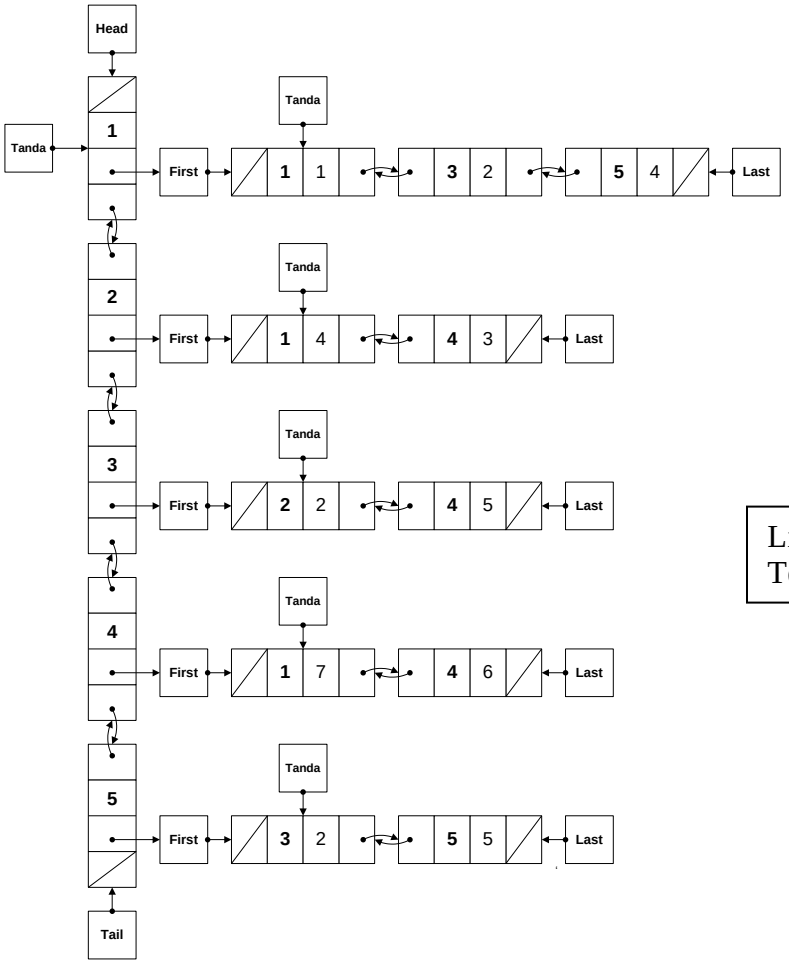
$$A = \begin{bmatrix} 1 & 0 & 2 & 0 & 4 \\ 4 & 0 & 0 & 3 & 0 \\ 0 & 2 & 0 & 5 & 0 \\ 7 & 0 & 0 & 6 & 0 \\ 0 & 0 & 2 & 0 & 5 \end{bmatrix}$$

Matriks Baris Terkompresi

Nomor Baris	**(Nomor Kolom)Elemen**
1	(1)1,(3)2,(5)4
2	(1)4,(4)3
3	(2)2,(4)5
4	(1)7,(4)6
5	(3)2,(5)5

Tabel 3.1 Matriks Kolom Terkompresi

Nomor Kolom	**(Nomor Baris)Elemen**
1	(1)1,(2)4,(4)7
2	(3)2
3	(1)2,(5)2
4	(2)3,(3)5,(4)6
5	(1)4,(5)5



Linked List Matriks Baris Terkompresi

$$A = \begin{bmatrix} 4 & 6 & 0 & 0 & 0 & 0 \\ 1 & 7 & 6 & 0 & 0 & 0 \\ 0 & 5 & 5 & 2 & 0 & 0 \\ 0 & 0 & 3 & 5 & 1 & 0 \\ 0 & 0 & 0 & 9 & 5 & 3 \\ 0 & 0 & 0 & 0 & 4 & 2 \end{bmatrix}$$

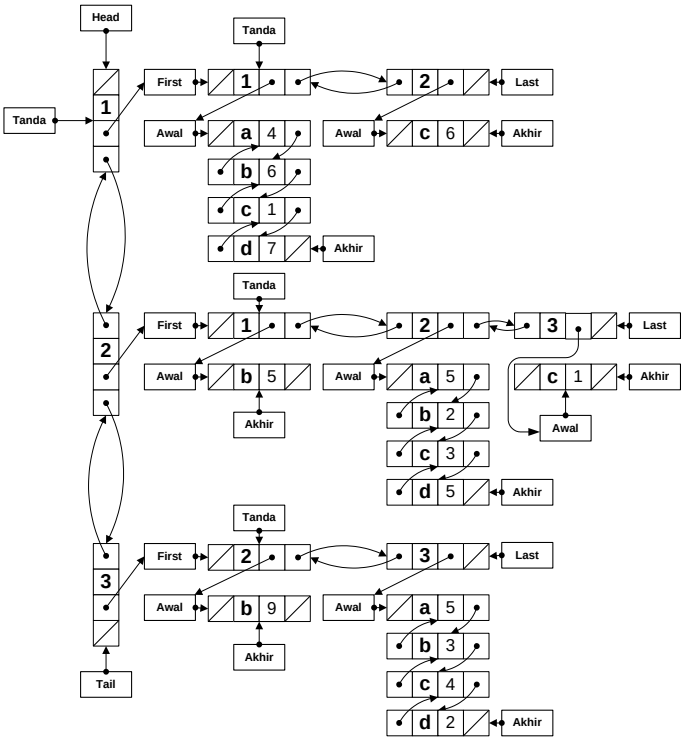
Tabel 3.2 Matriks Partisi Baris Terkompresi

Nomor Partisi Baris	**Nomor Partisi Kolom**		
	1	**2**	**3**
1	(a)4,(b)6,(c)1,(d)7	(c)6	-
2	(b)5	(a)5,(b)2,(c)3,(d)5	(c)1
3	-	(b)9	(a)5,(b)3,(c)4,(d)2

Tabel 3.3 Matriks Partisi Kolom Terkompresi

Nomor Partisi	**Nomor Partisi Baris**

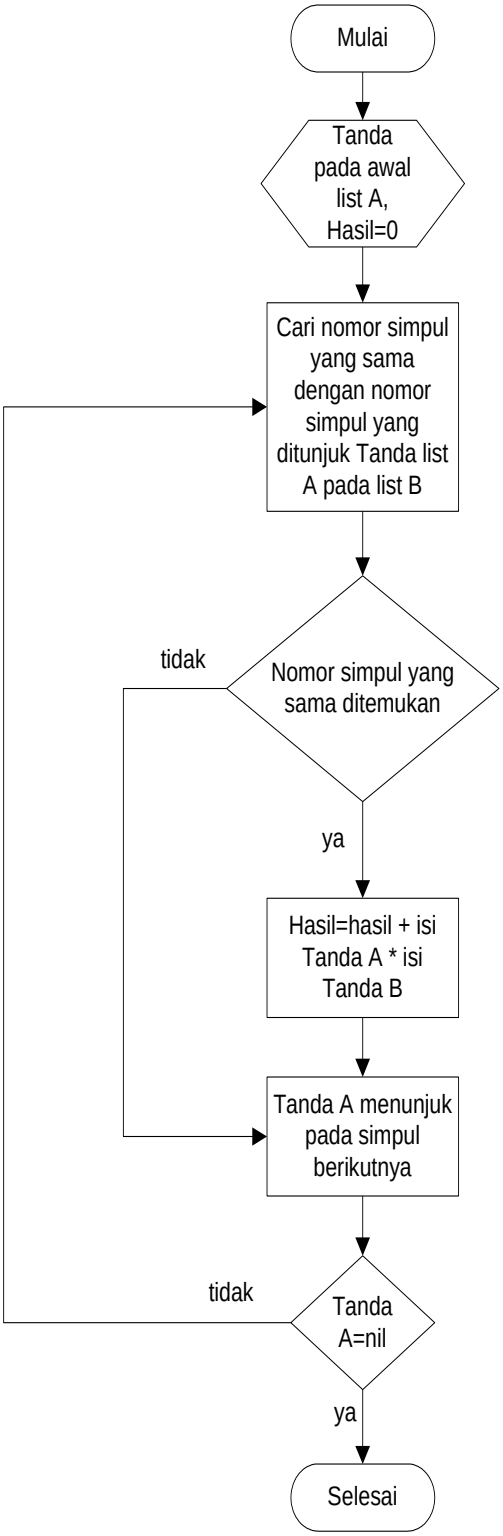
Kolom	1	2	3
1	(a)4,(b)6,(c)1,(d)7	(b)5	-
2	(c)6	(a)5,(b)2,(c)3,(d)5	(b)9
3	-	(c)1	(a)5,(b)3,(c)4,(d)2

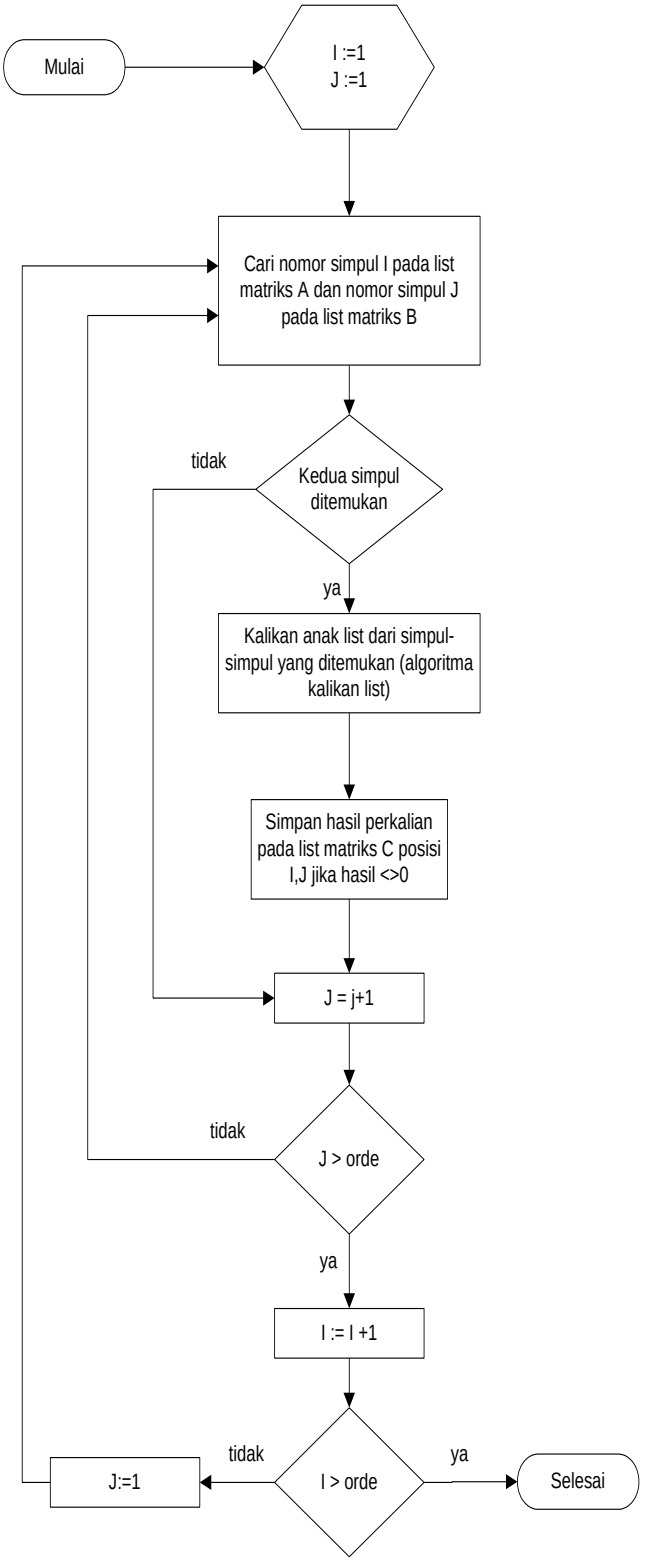


Linked List Matriks Partisi Baris Terkompresi

Dengan pertimbangan rumus perkalian matriks, matriks operand 1 (matriks A) disimpan dalam bentuk baris terkompresi, dan matriks operand 2 (matriks B) disimpan dalam bentuk kolom terkompresi, dan hasil (matriks C) disimpan dalam bentuk baris terkompresi.

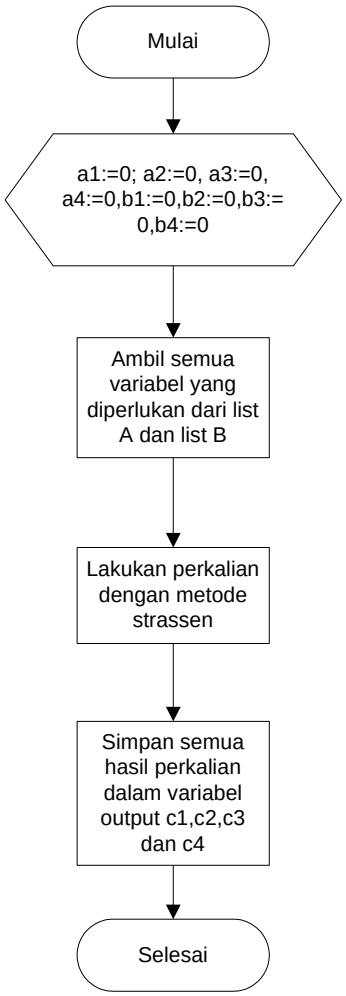
Untuk perkalian matriks biasa secara umum algoritma terdiri dari dua prosedur utama, yaitu prosedur perkalian sublist (baris tertentu × kolom tertentu) yang menghasilkan bilangan skalar sebagai elemen dalam matriks C,perkalian dilakukan terhadap setiap nomor simpul yang sama, jika nomor simpul yang sama tidak ditemukan, maka tidak dilakukan perkalian, karena hasilnya adalah nol, seluruh hasil perkalian akan dijumlahkan sebagai hasil. Prosedur berikutnya adalah prosedur untuk perulangan penghitungan setiap elemen dalam matriks C. Dalam penghitungan setiap elemen, program akan terlebih dahulu mencari baris dan kolom yang memenuhi kriteria, jika salah satu atau keduanya tidak ditemukan maka tidak dilakukan perkalian.


Mulai
Tanda pada awal list A, Hasil=0
Cari nomor simpul yang sama dengan nomor simpul yang ditunjuk Tanda list A pada list B
Nomor simpul yang sama ditemukan
tidak
ya
Hasil=hasil + isi Tanda A * isi Tanda B
Tanda A menunjuk pada simpul berikutnya
Tanda A=nil
tidak
ya
Selesai


Mulai
I :=1
J :=1
Cari nomor simpul I pada list matriks A dan nomor simpul J pada list matriks B
Kedua simpul ditemukan
tidak
ya
Kalikan anak list dari simpul-simpul yang ditemukan (algoritma kalikan list)
Simpan hasil perkalian pada list matriks C posisi I,J jika hasil <>0
J = j+1
J > orde
tidak
ya
I := I +1
I > orde
tidak
ya
J:=1
Selesai

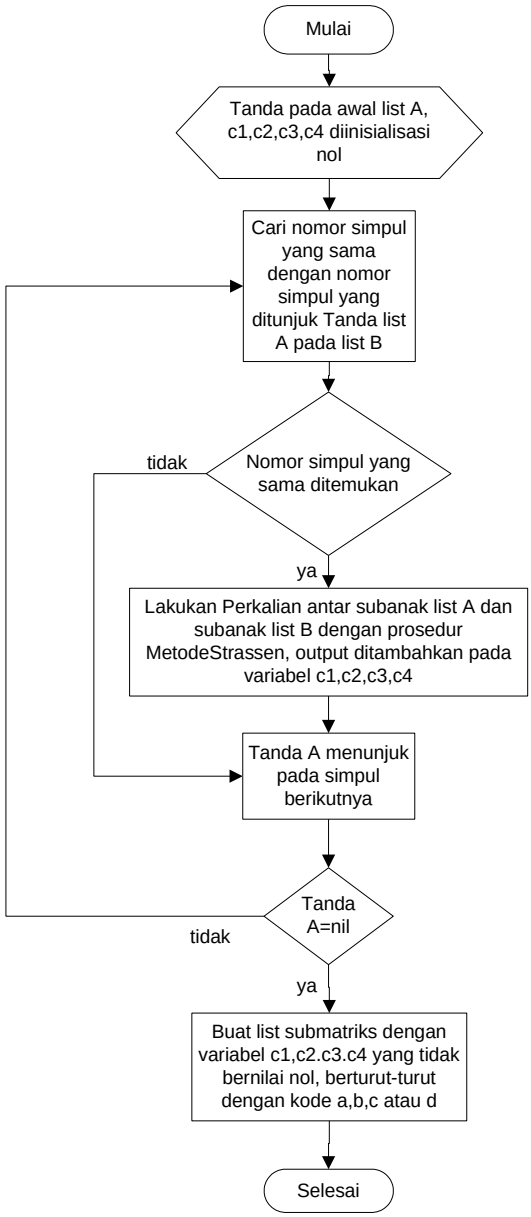
Algoritma perkalian untuk mengalikan list matriks terpartisi tidak terlalu berbeda dengan algoritma matriks biasa, hanya dalam hal ini tidak ada perkalian antar elemen-elemen dari matriks-matriks operan, melainkan submatriks-submatriks dari matriks-matriks operan. Algoritma perkalian antarsubmatriks ini akan menggunakan metode strassen, dan dibuat dalam sebuah prosedur terpisah yaitu prosedur MetodeStrassen.

Dalam perkalian antarsubmatriks (prosedur MetodeStrassen), delapan elemen akan diambil dari submatriks-submatriks operan untuk dimasukkan kedalam persamaan strassen, jika elemen tidak ditemukan dalam list, maka variabel untuk elemen tersebut diisi dengan nilai nol. Setelah semua variabel didapatkan, maka penghitungan hasil dilakukan dan disimpan dalam variabel-variabel elemen matriks.

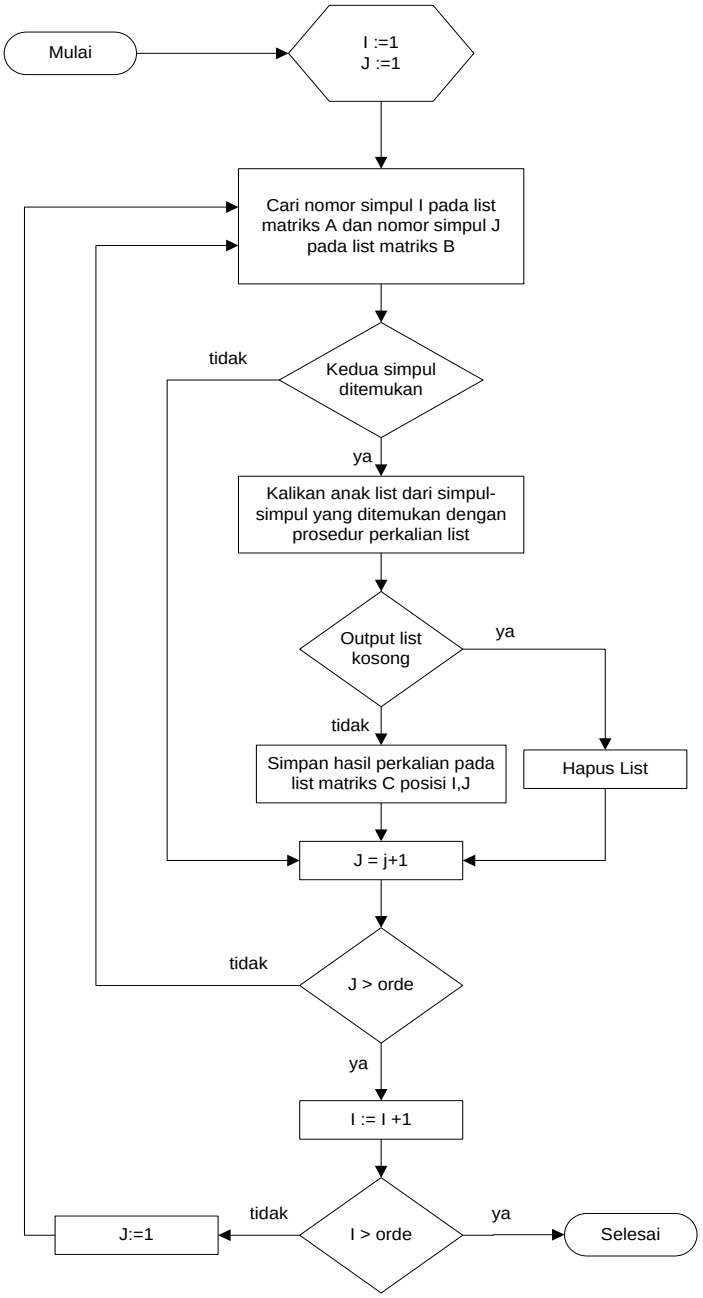


Flowchart Prosedur MetodeStrassen

Prosedur MetodeStrassen merupakan bagian dari algoritma perkalian linked list matriks partisi yang secara umum sama dengan algoritma perkalian linked list matriks konvensional, setelah prosedur perkalian antarsubmatriks, ada dua prosedur lagi yang diperlukan yaitu prosedur perkalian antar listanak dan prosedur perkalian list induk(matriks)



Flowchart Perkalian List Matriks Terpartisi



Flowchart Perkalian Matriks Terpartisi

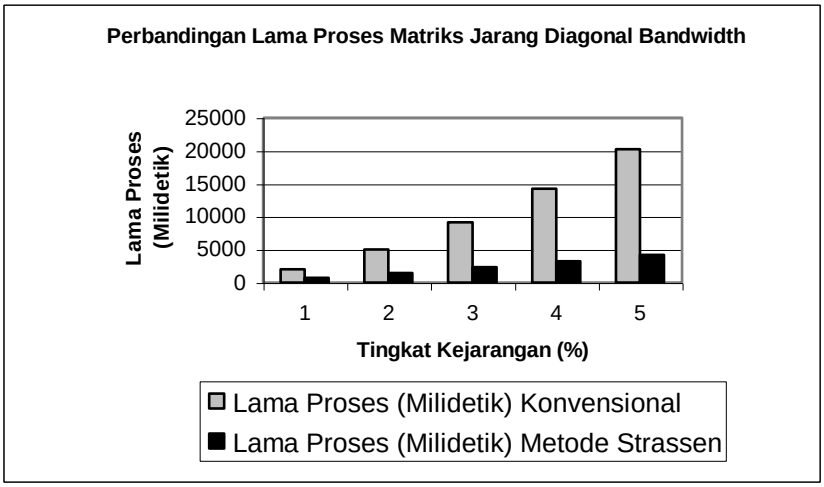
IV. Analisis

Analisis kecepatan proses dilakukan dengan membandingkan lima perkalian matriks 1000×1000, baik dari jenis diagonal bandwidth maupun tersebar, dengan tingkat kejarangan dari 1 % hingga 5 %, pada perangkat lunak dengan metode konvensional dan metode strassen.

Pembandingan pertama membandingkan lama proses perkalian matriks-matriks diagonal bandwidth dengan tingkat kejarangan yang sama, dari pembandingan tersebut didapatkan data dalam tabel dan grafik berikut

Tabel Perbandingan Lama Proses Perkalian Matriks Diagonal Bandwidth

Nama File Matriks (*.mtx)		Orde	Tingkat Kejarangan Matriks	Lama Proses Perkalian (Milidetik)	
A	B			Konvensional	Metode Strassen
A1000d1	B1000d1	1000×1000	1 %	2030	710
A1000d2	B1000d2	1000×1000	2 %	5050	1490
A1000d3	B1000d3	1000×1000	3 %	9120	2360
A1000d4	B1000d4	1000×1000	4 %	14230	3240
A1000d5	B1000d5	1000×1000	5 %	20210	4180

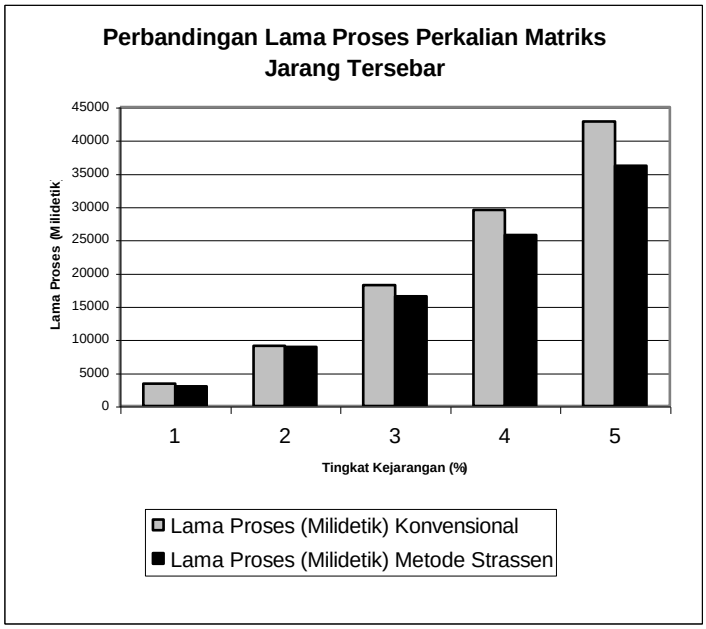


Grafik Perbandingan Lama Proses Perkalian Matriks Diagonal Bandwidth

Pembandingan kedua membandingkan lama proses perkalian matriks-matriks jarang tersebar dengan tingkat kejarangan yang sama, dari pembandingan tersebut didapatkan data dalam tabel dan grafik berikut

Tabel Perbandingan Lama Proses Perkalian Matriks Jarang Tersebar

Nama File Matriks (*.mtx)		Orde	Tingkat Kejarangan Kedua Matriks	Lama Proses Perkalian (Milidetik)	
A	B			Konvensional	Metode Strassen
A1000d1	B1000d1	1000×1000	1 %	3400	2970
A1000d2	B1000d2	1000×1000	2 %	9070	8900
A1000d3	B1000d3	1000×1000	3 %	18180	16530
A1000d4	B1000d4	1000×1000	4 %	29500	25760
A1000d5	B1000d5	1000×1000	5 %	42840	36150



Grafik Perbandingan Lama Proses Perkalian Matriks Jarang Tersebar

Analisis perbandingan penggunaan memori untuk menyimpan matriks jarang membandingkan tiga metode penyimpanan matriks jarang, yaitu penyimpanan dengan array, penyimpanan dengan linked list, dan penyimpanan dengan linked list untuk matriks terpartisi.

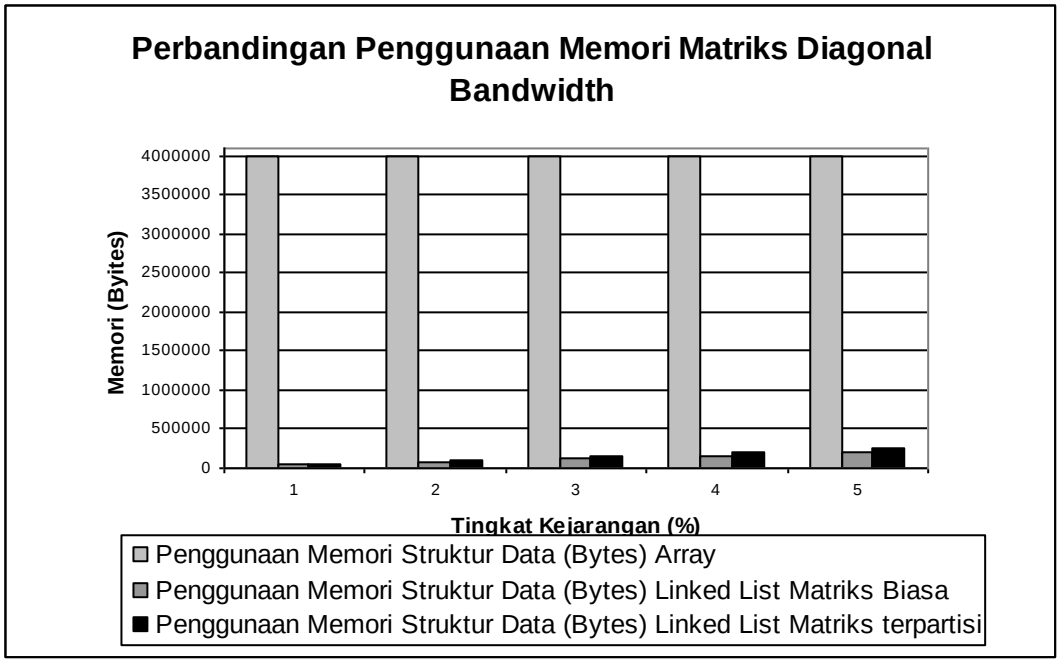
Analisis tersebut dilakukan terhadap dua jenis matriks, yaitu matriks jarang diagonal bandwidth dan matriks jarang tersebar, matriks yang diuji adalah lima buah matriks

dengan tingkat kejarangan dari 1 % hingga 5 % dari masing-masing jenis.

Analisis perbandingan penggunaan matriks diagonal bandwidth menghasilkan data dalam tabel dan grafik berikut

Tabel Perbandingan Penggunaan Memori Matriks Diagonal Bandwidth

Nama File (*.mtx)	Tingkat Kejarangan (%)	Penggunaan Memori Struktur Data (Bytes)		
		Array	Linked List Matriks Biasa	Linked List Matriks terpartisi
A1000d1	1	4000000	43900	53864
A1000d2	2	4000000	83600	103480
A1000d3	3	4000000	123100	152844
A1000d4	4	4000000	162400	201960
A1000d5	5	4000000	206500	250824

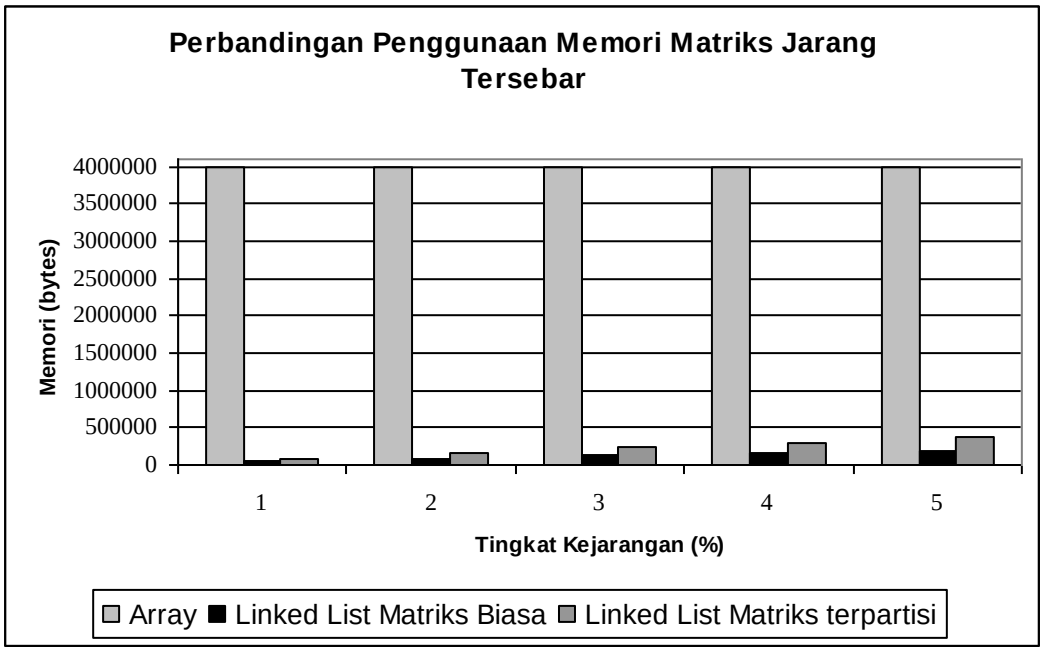


Grafik Perbandingan Penggunaan Memori Matriks Diagonal Bandwidth

Analisis perbandingan penggunaan matriks jarang tersebar menghasilkan data dalam tabel dan grafik berikut

Tabel Perbandingan Penggunaan Memori Matriks Jarang Tersebar

Nama File (*.mtx)	Tingkat Kejarangan (%)	Penggunaan Memori Struktur Data (Bytes)		
		Array	Linked List Matriks Biasa	Linked List Matriks terpartisi
A1000s1	1	4000000	43788	79044
A1000s2	2	4000000	83112	156152
A1000s3	3	4000000	122204	231364
A1000s4	4	4000000	160928	305024
A1000s5	5	4000000	199284	376756



Grafik Perbandingan Penggunaan Memori Matriks Jarang Tersebar

V. Kesimpulan

Dari data hasil analisis, dapat disimpulkan bahwa algoritma dengan implementasi metode strassen lebih cepat dari algoritma perkalian matriks konvensional. Hal ini dipengaruhi beberapa faktor, selain metode strassen yang mengurangi jumlah proses, setiap proses perulangan pada algoritma dengan metode strassen langsung menghasilkan empat elemen,

sehingga jumlah proses yang harus dilakukan menjadi setengah dari algoritma konvensional.

Setelah melakukan analisis, dapat diambil kesimpulan besarnya penggunaan memori penyimpanan matriks dengan linked list secara umum lebih sedikit dari penyimpanan matriks jika dengan menggunakan struktur data array, karena pencatatan hanya dilakukan terhadap elemen tidak nol.

Besarnya penggunaan memori pada aplikasi perkalian matriks metode strassen lebih banyak dari perkalian konvensional. Hal ini disebabkan banyaknya sublist dari linked list penyimpanan matriks terpartisi, sedangkan pada setiap sublist ada tiga pencatatan alamat, yaitu awal list, akhir list dan tanda simpul terakhir yang diakses.

Referensi

1. Rice, John R. 1981.*'Matrix Computations and Mathematical Software'.* McGraw-Hill Book Company: New York.
2. Robinson, John. 1966. '*Structural Matrix Analysis for the Engineer*'. John Wiley & Sons : New York.
3. Golub, Gene H., Van Loan, Charles F.1989.*'Matrix Computations, Second Edition'.* The John Hopkins University Press: London.
4. Bambang Hariyanto. 2000. '*Struktur Data*'. Penerbit Informatika: Bandung.