TUGAS AKHIR

ALGORITMA PEMROGRAMAN DINAMIK DALAM PENYELESAIAN MASALAH PERKALIAN RANGKAIAN MATRIKS (*MATRIX-CHAIN*)

Diajukan sebagai salah satu syarat memperoleh
Gelar Sarjana Sains pada Fakultas MIPA
Universitas Diponegoro

DISUSUN OLEH :

HELMIE ARIF WIBAWA

NIM. J 2A 096 027

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS DIPONEGORO
SEMARANG
2001

HALAMAN PENGESAHAN

Lembar 1

Judul : Algoritma Pemrograman Dinamik dalam Penyelesaian Masalah Perkalian Rangkaian Matriks (*Matrix-chain*)

Nama : Helmie Arif Wibawa

NIM : J 2A 096 027

Jurusan : Matematika

Telah lulus ujian sarjana pada tanggal 2 Oktober 2001

Semarang, Oktober 2001

Panitia Penguji Ujian Sarjana
Jurusan Matematika

Ketua,

Drs. Suhartono, MIKomp
NIP. 131 285 523

Jurusan Matematika

Ketua,

Drs. Bayu Surarso, MSc, PhD
NIP. 131 764 886

HALAMAN PENGESAHAN

Lembar 2

Judul : Algoritma Pemrograman Dinamik dalam Penyelesaian Masalah Perkalian Rangkaian Matriks (*Matrix-Chain*)

Nama : Helmie Arif Wibawa

NIM : J 2A 096 027

Jurusan : Matematika

Telah lulus ujian sarjana pada tanggal 2 Oktober 2001

Semarang, 8 Oktober 2001

Pembimbing Utama,

Drs. Suhartono, MIKomp
NIP. 131 285 523

Pembimbing Anggota,

Drs. Bambang Yismianto
NIP. 131 626 757

KATA PENGANTAR

Alhamdulillah, puji syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat dan hidayahNya sehingga penulisan Tugas Akhir yang berjudul " **Algoritma Pemrograman Dinamik dalam Penyelesaian Masalah Perkalian Rangkaian Matriks (*matrix-chain*)** " ini dapat terselesaikan.

Tugas Akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar sarjana strata satu (S-1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Diponegoro Semarang.

Dalam kesempatan ini, ijinkanlah penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Drs. Bayu Surarso, MSc, PhD. selaku Ketua Jurusan Matematika
2. Drs. Suhartono, MIKomp. selaku pembimbing utama yang telah berkenan memberikan bimbingan dan pengarahan hingga selesainya Tugas Akhir ini
3. Drs. Bambang Yismianto selaku pembimbing anggota yang telah berkenan memberikan bimbingan dan pengarahan hingga selsainya Tugas Akhir ini.
4. Dra. Dwi Ispriyanti. Msi. selaku dosen wali yang dengan sabar memberikan nasehatnya kepada penulis selama kuliah.
5. Semua pihak yang membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa dalam Tugas Akhir ini masih banyak terdapat kesalahan dan kekurangan. Oleh karena itu segala kritik yang membangun, tanggapan ataupun dari semua pihak akan penulis terima demi kesempurnaan Tugas Akhir ini.

Akhir kata penulis mengharapkan semoga Tugas Akhir ini dapat bermanfaat bagi para pembaca dan bagi perkembangan ilmu pengetahuan di masa yang akan datang..

Semarang, Oktober 2001

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Penyelesaian perkalian lebih dari dua matriks atau yang disebut perkalian rangkaian matriks dengan menggunakan komputer akan menjamin kecepatan dan ketepatan hasil yang diperoleh. Dalam proses penyelesaiannya, urutan pasangan matriks yang akan dikalikan terlebih dahulu ternyata memberikan dampak yang cukup besar terhadap kecepatan proses penyelesaiannya. Oleh karena itu penentuan urutan pasangan matriks yang tepat sangat dibutuhkan dalam penyelesaian perkalian rangkaian matriks. Algoritma pemrograman dinamik yang biasa digunakan dalam penyelesaian masalah optimasi dapat digunakan untuk menemukan urutan pasangan matriks yang tepat tersebut. Algoritma pemrograman dinamik akan menentukan urutan pasangan matriks yang tepat dengan mencari pasangan matriks yang membutuhkan proses perkalian skalar paling sedikit.

ABSTRACT

Solving multiplication more than two matrix (called matrix-chain multiplication) with computer will have a good result in accuracy and time consuming on computation. The sequence of matrix couple give the impact into the multiplication procces. Therefore precise determination couple of matrix is necessary. The algorithm of dynamic programming that usually used to solve the optimation problem can be used to find the precise couple matrix. It determine sequence of matrix couple by finding couple of matrix which least scalar multiplication.

DAFTAR SIMBOL

A,B,C	= matriks
$A^{(i)}$	= baris ke- i matriks A
$A_{(j)}$	= kolom ke- j matriks A
a_{ij}	= elemen matriks A
b_{ij}	= elemen matriks B
c_{ij}	= elemen matriks C
O	= big Oh
S	= himpunan semua fungsi dengan domain himpunan bilangan riil
T(n)	= running time
p(n)	= banyaknya cara peletakan tanda kurung dalam rangkaian matriks dengan anggota sebanyak n matriks
$f_t(x_t)$ / $f_t(y_t)$	= optimum return function tahap t
x, y	= state atau keadaan
R_t	= nilai kontribusi pada tahap t
@	= simbol operasi matematik (×, + , ÷, -)
t	= tahap
Ai..j	= matriks yang dihasilkan dari bentuk perkalian Ai.Ai+1….Aj
f[i,j]	= banyaknya perkalian skalar pada perhitungan Ai.Ai+1… Aj
s[i,j]	= letak tanda kurung yang tepat pada bentuk perkalian Ai.Ai+1… Aj

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pada saat ini penggunaan komputer telah merebak pada berbagai aspek kehidupan manusia. Sebagai salah satu contohnya adalah penggunaan komputer untuk mempercepat perhitungan yang rumit dan butuh waktu yang banyak jika dilakukan dengan cara perhitungan manual. Dari hal tersebut terlihat bahwa dengan menggunakan komputer dapat dilakukan penghematan waktu, karena prosesnya yang lebih cepat.

Kecepatan komputer dalam melakukan perhitungan maupun proses yang lain dipengaruhi oleh beberapa faktor antara lain faktor perangkat keras dan program aplikasi yang dipakai itu sendiri. Suatu program aplikasi, terutama untuk aplikasi perhitungan biasanya berisi rangkaian prosedur untuk menyelesaikan permasalahan tersebut.

Prosedur yang digunakan dalam setiap aplikasi harus dapat menyelesaikan permasalahan yang ada dan juga diusahakan mempunyai waktu yang cepat dalam eksekusinya. Pada aplikasi menghitung suatu persamaan, cepat atau lambatnya suatu perhitungan terkadang dipengaruhi oleh pemilihan suku mana yang akan diproses terlebih dahulu.

Hal ini dapat dilihat pada permasalahan perkalian rangkaian matriks (*matrix-chain*). Dalam permasalahan ini terdapat suatu perkalian rangkaian matriks yaitu suatu bentuk perkalian sebanyak n matriks secara serempak. Untuk

menyelesaikannya dalam waktu yang paling cepat haruslah diketahui urutan pasangan matriks mana yang harus dikalikan terlebih dahulu, karena jika n matriks itu dikalikan dengan pemilihan pasangan matriks secara acak, maka dapat terjadi pemilihan pasangan matriks itu menghasilkan urutan pasangan yang menyebabkan program aplikasi mempunyai waktu eksekusi yang paling lama.

Dalam hal ini algoritma pemrograman dinamik merupakan salah satu cara untuk dapat mengetahui pasangan matriks terurut manakah yang harus didahulukan untuk diproses dalam program aplikasi.

1.2. Perumusan Permasalahan

Dalam penulisan Tugas Akhir ini algoritma pemrograman dinamik akan digunakan untuk menyelesaikan masalah perkalian rangkaian matriks, sehingga permasalahan dalam tugas akhir ini dapat dirumuskan sebagai berikut :

- Bagaimanakah cara kerja algoritma pemrograman dinamik untuk menyelesaikan masalah perkalian rangkaian matriks ?
- Menentukan running-time algoritma pemrograman dinamik dalam menyelesaikan masalah perkalian rangkaian matriks.

1.3. Tujuan Penulisan

Tujuan yang ingin dicapai dalam penulisan Tugas Akhir ini adalah untuk menyelesaikan masalah perkalian rangkaian matriks dengan menggunakan algoritma pemrograman dinamik dan menentukan besarnya running-time yang

dibutuhkan algoritma pemrograman dinamik untuk menyelesaikan masalah perkalian rangkaian matriks tersebut.

1.4. Pembatasan Masalah

Dalam penulisan Tugas Akhir ini masalah dibatasi pada :

- Cara kerja algoritma pemrograman dinamik dalam penyelesaian masalah perkalian rangkaian matriks.
- Prosedur penyelesaian yang digunakan adalah prosedur rekursif mundur.
- Analisa algoritma yang dilakukan adalah analisa running-time pada proses penentuan urutan pasangan matriks.
- Analisa running-time dilakukan dengan asumsi terjadi keadaan terburuk.

1.5. Sistematika Penulisan

Penulisan Tugas Akhir ini dikelompokkan menjadi empat bagian, mulai bab I sampai dengan bab IV. Bab I merupakan pendahuluan yang meliputi latar belakang masalah, perumusan masalah, pembatasan masalah, tujuan penulisan, dan sistematika penulisan.

Bab II adalah materi penunjang yang berisi beberapa materi yang digunakan sebagai pendukung materi pembahasan masalah pada bab berikutnya. Bab II ini membicarakan tentang konsep dasar perkalian matriks, konsep dasar algoritma, dan notasi big Oh.

Bab III berisi pembahasan tentang algoritma pemrograman dinamik dalam penyelesaian masalah perkalian rangkaian matriks. Pada bagian ini akan dikupas masalah perkalian rangkaian matriks itu sendiri serta bagaimana cara kerja algoritma pemrograman dinamik untuk menyelesaikan masalah tersebut. Selain itu, pada bab III ini juga akan dibahas tentang running time yang dibutuhkan algoritma pemrograman dinamik untuk menyelesaikan masalah perkalian rangkaian matriks. Penulisan Tugas Akhir ini diakhiri pada bab IV yang berisi tentang kesimpulan dari seluruh bab sebelumnya.

BAB II

MATERI PENUNJANG

Dalam bab ini dijabarkan tentang beberapa teori yang dapat menunjang pembahasan pada bab III. Teori yang dijabarkan merupakan konsep dasar yang meliputi matriks, operasi perkalian matriks, algoritma, notasi big Oh, dan running time.

2.1. Matriks dan Operasi Perkalian Matriks

2.1.1. Matriks

Definisi 2.1.1.1.

Sebuah matriks adalah suatu susunan segi empat siku-siku dari bilangan-bilangan. Bilangan-bilangan dalam susunan tersebut dinamakan entri dalam matriks.

Contoh 2.1.1

Susunan berikut ini merupakan matriks.

$$\begin{bmatrix} 3 & 5 \\ 1 & 4 \\ 7 & 11 \end{bmatrix}, \quad \begin{bmatrix} 0 & 3 & -2 & 8 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 5 & 5 & 5 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 5 \end{bmatrix}$$

Seperti yang ditunjukkan pada contoh 2.1.1, maka ukuran matriks dapat bermacam-macam. Ukuran matriks itu sendiri dijelaskan dengan menyatakan banyaknya baris dan banyaknya kolom pada matriks tersebut. Sebagai contoh yaitu matriks pertama pada contoh 2.1.1 mempunyai 3 baris dan 2 kolom

sehingga ukuran dari matriks tersebut adalah 3 kali 2 yang biasanya ditulis dengan 3 x 2.

Dalam penulisannya, huruf besar digunakan untuk menyatakan matriks-matriks, sedangkan huruf kecil digunakan untuk menyatakan entri dalam matriks. Jika A adalah sebuah matriks, maka a_{ij} digunakan untuk menyatakan entri yang terdapat pada baris ke- i dan kolom ke- j dari A. Sehingga suatu matriks berukuran m x n secara umum dapat ditulis

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

sedangkan baris ke-i dari matriks A adalah matriks berukuran 1 x n dan untuk i=1,2,…,m, baris ke-i tersebut dapat ditulis

$$A^{(i)} = \begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix}$$

dan untuk j= 1,2,…,n kolom ke-j dari matriks A dapat ditulis

$$A_{(j)} = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}$$

2.1.2. Operasi Perkalian Matriks

Definisi 2.1.2.1

Jika A adalah suatu matriks dan c adalah suatu skalar, maka hasil kali cA adalah matriks yang diperoleh dengan mengalikan masing-masing entri dari A oleh c.

Contoh 2.1.2

Jika A adalah matriks

$$A = \begin{bmatrix} -1 & 3 \\ 4 & 0 \\ 1 & 2 \end{bmatrix}$$

maka

$$3A = \begin{bmatrix} -3 & 9 \\ 12 & 0 \\ 3 & 6 \end{bmatrix} \quad \text{dan} \quad (-2A) = \begin{bmatrix} 2 & -6 \\ -8 & 0 \\ -2 & -4 \end{bmatrix}$$

Definisi 2.1.2.2.

Jika A adalah matriks berukuran m x r dan B adalah matriks berukuran r x n, maka hasil kali AB adalah matriks berukuran m x n yang entri-entrinya ditentukan sebagai berikut. Untuk mencari entri pada baris ke- i dan kolom ke- j dari matriks AB, dipilih baris ke- i dari matriks A dan kolom ke- j dari matriks B. Entri-entri yang bersesuaian dari baris dan kolom tersebut dikalikan bersama-sama dan kemudian hasil kali yang diperoleh dijumlahkan.

Misalkan C = AB dan c_{ij} adalah entri dari matriks C, maka c_{ij} berada pada baris ke- i dan kolom ke- j dari AB. Misalkan $A^{(i)}$ sebagai notasi dari baris ke- i matriks A dan $B_{(j)}$ sebagai kolom ke- j dari matriks B, maka dapat ditulis

$$C_{(j)}^{(i)} = A^{(i)}B_{(j)} \qquad (2.1.2.1)$$

Persamaan 2.1.2.1 dapat digambarkan pada hasil kali antara sembarang matriks A dan sembarang matriks B yang memenuhi syarat untuk operasi perkalian matriks.

$$AB = \begin{bmatrix} * & * & \cdots & * \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \cdots & * \end{bmatrix} \begin{bmatrix} b_{11} & * & \cdots & * \\ b_{21} & * & \cdots & * \\ \vdots & \vdots & \vdots & \vdots \\ b_{r1} & * & \cdots & * \end{bmatrix} = \begin{bmatrix} * & * & \cdots & * \\ c_{21} & * & \cdots & * \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \cdots & * \end{bmatrix}$$

Terlihat bahwa $c_{21} = a_{2*}\, b_{*1}$

$$= a_{21}.b_{11} + a_{22}.b_{21} + \ldots + a_{2r}.b_{r1}$$

Sehingga secara umum persamaan 2.1.2.1 dapat ditulis lengkap sebagai

$$c_{ij} = a_{i1}.b_{1j} + a_{i2}.b_{2j} + \ldots + a_{ir}.b_{rj} \qquad (2.1.2.2)$$

Untuk memperjelas definisi perkalian dua buah matriks disajikan contoh 2.1.3

Contoh 2.1.3

Cari hasil kali AB dari

$$A = \begin{bmatrix} 1 & 0 & -1 & 2 \\ 0 & 2 & 1 & 3 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & -1 \\ 1 & 1 \\ 2 & 0 \\ 1 & 2 \end{bmatrix}$$

Penyelesaian

Karena A adalah matriks berukuran 2 x 4 dan B adalah matriks berukuran 4 x 2, maka hasil kali AB adalah matriks berukuran 2 x 2. Entri pada baris ke- 2 dan kolom ke- 2 dari AB dihitung sebagai berikut.

$$\begin{bmatrix} 1 & 0 & -1 & 2 \\ 0 & 2 & 1 & 3 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 1 \\ 2 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} \square & \square \\ \square & 8 \end{bmatrix}$$

$(0.(-1)) + (2.1) + (1.0) + (3.2) = 8$

Perhitungan untuk hasil selebihnya adalah

$(1.0) + (0.1) + ((-1).2) + (2.1) = 0$

$$(1.(-1)) + (0.1) + ((-1).0) + (2.2) = 3$$

$$(0.0) + (2.1) + (1.2) + (3.1) = 7$$

$$AB = \begin{bmatrix} 0 & 3 \\ 7 & 8 \end{bmatrix}$$

2.1.3. Sifat Assosiatif dari Perkalian Matriks

Meskipun sifat komutatif untuk perkalian bilangan riil tidak berlaku pada operasi perkalian matriks, namun banyak sifat-sifat operasi pada bilangan riil yang berlaku pada matriks. Salah satu di antara sifat yang paling penting adalah sifat assosiatif pada operasi perkalian.

Theorema 2.1.3.1

Jika matriks A bersesuaian dengan matriks B dalam operasi perkalian matriks dan matriks B bersesuaian dengan matriks C, maka matriks A akan bersesuaian dengan matriks BC dan matriks AB akan bersesuaian dengan matriks C sehingga

$$A(BC) = (AB)C \qquad (2.1.3.5)$$

Bukti

Misalkan matriks A berukuran m x n, matriks B berukuran n x p, dan matriks C berukuran p x s maka matriks (AB) berukuran m x p dan matriks (BC) berukuran n x s. Baris ke-i dari matriks (AB) untuk i=1,2,…,m adalah

$$(AB)^{(i)} = (A^{(i)}B_{(1)} \quad A^{(i)}B_{(2)} \quad \dots \quad A^{(i)}B_{(p)})$$

dan kolom ke-j dari matriks (BC) untuk j= 1,2,…,s adalah

$$(BC)_{(j)} = \begin{pmatrix} B^{(1)}C_{(j)} \\ B^{(2)}C_{(j)} \\ \vdots \\ B^{(n)}C_{(j)} \end{pmatrix}$$

sehinga untuk sembarang posisi i dan j berlaku

$$((AB)C)^{(i)}_{(j)} = (AB)^{(i)}C_{(j)}$$

$$= (\mathrm{A}^{(i)}\mathrm{B}_{(1)}).\mathrm{c}_{1j} + (\mathrm{A}^{(i)}\mathrm{B}_{(2)}).\mathrm{c}_{2j} + \ldots + (\mathrm{A}^{(i)}\mathrm{B}_{(p)}).\mathrm{c}_{pj}$$

$$= (a_{i1}b_{11} + a_{i2}b_{21} + \ldots + a_{in}b_{n1}).c_{1j} + (a_{i1}b_{12} + a_{i2}b_{22} + \ldots + a_{in}b_{n2}).c_{2j}$$

$$+ \ldots + (a_{i1}b_{1p} + a_{12}b_{2p} + \ldots + a_{in}b_{np}).c_{pj} \qquad \text{(i)}$$

$$(A(BC))^{(i)}_{(j)} = A^{(i)}(BC)_{(j)}$$

$$= a_{i1}.(B^{(1)}C_{(j)}) + a_{i2}.(B^{(2)}C_{(j)}) + \ldots + a_{in}.(B^{(n)}C_{(j)})$$

$$= a_{i1}(b_{11}c_{1j} + b_{12}c_{2j} + \ldots + b_{1p}c_{pj}) + a_{i2}(b_{21}c_{1j} + b_{22}c_{2j} + \ldots + b_{2p}c_{pj})$$

$$+ \ldots + a_{in}(b_{n1}c_{1j} + b_{n2}c_{2j} + \ldots + b_{np}c_{pj}) \qquad \text{(ii)}$$

karena i dan j adalah sembarang maka pembandingan antara persamaan (i) dan (ii) menunjukkan bahwa (AB)C = A(BC)

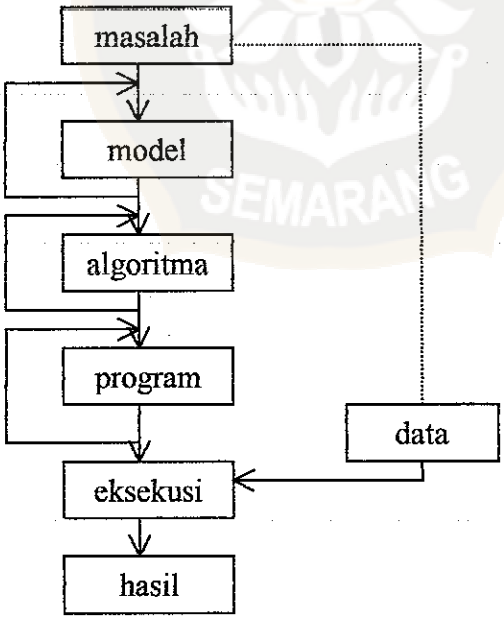
2.2. Algoritma

Terbentuknya model matematika yang sesuai pada suatu pemecahan masalah hanya merupakan bagian dari pemecahan masalah tersebut. Oleh karena itu untuk melengkapi proses pemecahan masalah tersebut diperlukan suatu metode atau langkah-langkah yang akan digunakan untuk membawa model matematika

tersebut ke arah penyelesaiannya. Langkah-langkah tersebut biasa disebut dengan algoritma.

Algoritma pertama kali diperkenalkan oleh seorang ahli matematika yaitu Abu Ja'far Muhammad Ibnu Musa Al Khawarizmi, dimana secara bahasa algoritma mempunyai arti suatu metode khusus untuk menyelesaikan suatu masalah yang nyata. Di dalam bidang pemrograman, algoritma didefinisikan sebagai suatu metode khusus yang tepat dan terdiri dari serangkaian langkah yang terstruktur dan ditulis secara matematis yang akan dikerjakan untuk menyelesaikan masalah dengan bantuan komputer.

Di dalam prosesnya, hubungan antara masalah, algoritma, dan hasil dapat digambarkan sebagai berikut :



Gambar 2.2.1. Diagram hubungan masalah, algoritma, dan hasil

Suatu algoritma yang baik mempunyai beberapa kriteria sebagai berikut :

1. Adanya input

   Suatu algoritma mempunyai input yang diperoleh dari himpunan objek yang telah ditetapkan.

2. Adanya output

   Dalam menyelesaikan suatu masalah, algoritma harus menghasilkan output yang merupakan solusi atau hasil dari masalah yang sedang diselesaikan.

3. Efektif dan efisien

   Suatu algoritma dikatakan efektif jika algortima tersebut dapat menghasilkan suatu solusi yang sesuai dengan masalah yang sedang diselesaikan. Sedangkan algortima yang efisien adalah algoritma yang mempunyai waktu proses yang singkat dan penggunaan memorinya sedikit.

4. Jumlah langkahnya berhingga

   Banyaknya langkah yang disusun harus berhingga, karena jika tidak demikian maka proses yang dilakukan akan memerlukan waktu yang relatif lebih lama.

5. Berakhir

   Proses yang dilakukan dalam mencari suatu penyelesaian masalah harus berhenti atau berakhir dengan hasil akhir berupa solusi atau berupa informasi tentang tidak diketemukannya solusi. Dengan perkataan lain, baik dalam kondisi solusinya ada maupun tidak maka proses tetap harus mempunyai akhir.

6. Terstruktur

Urutan dari barisan langkah-langkah yang digunakan harus disusun sedemikian rupa agar proses penyelesaiannya tidak berbelit-belit dan memungkinkan waktu proses menjadi lebih singkat serta akan mempermudah di dalam melakukan pemeriksaan ulang.

2.3. Pengertian dan Sifat-Sifat Big Oh

Definisi 2.3.1

Diberikan S, himpunan semua fungsi yang mempunyai domain D yang mewakili himpunan bilangan riil. Dikatakan f(n) = O(g(n)), dibaca ' f(n) adalah big-Oh dari g(n)' jika ditemukan pasangan bilangan positif c dan k sedemikian hingga berlaku

$$| f(n) | \leq c.| g(n) |$$

untuk semua $n \in D$ dan $n \geq k$

Dari definisi 2.3.1 maka untuk menunjukkan f(n) = O (g(n)) hanya dibutuhkan untuk menentukan sepasang konstanta c dan k sedemikian hingga jika $n \geq k$ maka $| f(n) | \leq c.| g(n) |$. Penulisan f(n) = O(g(n)) berarti bahwa f(n) merupakan anggota dari O(g(n)) .

Contoh 2.3.1

Jika suatu fungsi $f(n) = 3n^3 + 2n^2$ merupakan fungsi dari waktu tempuh suatu algoritma maka f(n) adalah big-Oh dari n^3, yang dinotasikan $f(n) = O(n^3)$

Penyelesaian

Berdasarkan definisi 2.3.1 bahwa jika $f(n) = O(n^3)$ maka terdapat dua konstanta bulat positif c dan k sedemikian hingga berlaku

$$| 3n^3 + 2n^2 | \leq c.| n^3|$$

$$n = 0 \rightarrow f(0) = 0$$

$$n = 1 \rightarrow f(1) = 3 + 2 = 5$$

ambil $k = 0$ dan $c = 5$ sehingga berlaku $3n^3 + 2n^2 \leq 5n^3$ untuk $n \geq 0$.

Jadi terbukti bahwa $f(n) = O(n^3)$.

Teorema 2.3.1

Misalkan $f(n) = a_p n^p + a_{p-1}n^{p-1} + \ldots + a_1 n + a_0$ adalah fungsi polinomial derajat p dengan $a_p \neq 0$ dan $g(n) = n^p$, maka $f(n) = O(g(n))$.

Bukti :

Untuk membuktikan teorema 2.3.1 akan dicari sepasang konstanta c dan k sedemikian hingga $| f(n) | \leq c.| g(n) |$ untuk $n \geq k$

Dipilih $c = | a_p | + | a_{p-1} | + \ldots + | a_1 | + | a_0 |$ dan $k = 1$, maka untuk $n \geq 1$

$$| f(n) | = | a_p n^p + a_{p-1}n^{p-1} + \ldots + a_1 n + a_0 |$$

$$\leq | a_p n^p | + | a_{p-1}n^{p-1} | + \ldots + | a_1 n | + | a_0 |$$

$$\leq | a_p | | n^p | + | a_{p-1} | | n^{p-1} | + \ldots + | a_1 | | n | + | a_0 |$$

$$\leq | a_p | | n^p | + | a_{p-1} | | n^p | + \ldots + | a_1 | | n^p | + | a_0 | | n^p |$$

$= (| a_p | + | a_{p-1} | + \ldots + | a_1 | + | a_0 |) | n^p |$

$= c. | n^p |$

$= c.| g(n) |$

terlihat bahwa $| f(n) | \leq c.| g(n) |$, sehingga terbukti bahwa $f(n) = O(g(n))$.

Teorema 2.3.2

Misalkan S adalah himpunan semua fungsi dengan domain D yang merupakan himpunan bilangan riil. Fungsi-fungsi $f_1(n)$, $f_2(n)$, $g_1(n)$, dan $g_2(n)$ adalah anggota-anggota S sedemikian hingga $f_1(n) = O(g_1(n))$ dan $f_2(n) = O(g_2(n))$ maka berlaku :

(a). $f_1(n) + f_2(n) = O(\max \{g_1(n),g_2(n)\})$.

(b). $f_1(n) . f_2(n) = O(g_1(n) . g_2(n))$

Bukti

(a). Akan dibuktikan bahwa terdapat sepasang konstanta c dan k sedemikian hingga untuk $n \geq k$ berlaku

$$|f_1(n) + f_2(n) | = c.| \max\{g_1(n),g_2(n)\} |$$

Karena $f_1(n) = O(g_1(n))$ dan $f_2(n) = O(g_2(n))$ maka terdapat c_1, c_2, k_1 dan k_2 sedemikian hingga untuk $n \in D$ di mana $n \geq k_1$ dan $n \geq k_2$ berlaku :

$$| f_1(n) + f_2(n) | \leq | f_1(n) | + | f_2(n) |$$

$$\leq c_1.| g_1(n) | + c_2.| g_2(n) |$$

$$= c_1,g_1(n) + c_2.g_2(n)$$

$\leq c_1.\max\{g_1(n), g_2(n)\} + c_2.\max\{g_1(n), g_2(n)\}$

$= (c_1+c_2).\max\{g_1(n), g_2(n)\}$

$\leq c.|\max\{g_1(n), g_2(n)\}|$,untuk $c = c_1 + c_2$

maka untuk $k = \max\{k_1, k_2\}$ terbukti bahwa

$f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$.

(b). Karena $f_1(n) = O(g_1(n))$ dan $f_2(n) = O(g_2(n))$ maka terdapat c_1, c_2, k_1, dan k_2 bilangan positif sedemikian hingga untuk $n \in D$ dengan $n \geq k_1$ dan $n \geq k_2$ maka berlaku $|f_1(n)| \leq c_1.|g_1(n)|$ dan $|f_2(n)| \leq c_2.|g_2(n)|$.

Pilih $k = \max\{k_1, k_2\}$ dan $c = c_1.c_2$, maka

$|f_1(n).f_2(n)| = |f_1(n)|.|f_2(n)|$

$\leq c_1.|g_1(n)|.c_2.|g_2(n)|$

$= c_1.c_2.|g_1(n)|.|g_2(n)|$

$= c.|g_1(n).g_2(n)|$

sehingga terbukti bahwa $f_1(n).f_2(n) = O(g_1(n).g_2(n))$

2.4. Running Time

Proses suatu algoritma di dalam mencari solusi dari suatu masalah memerlukan waktu tertentu, di mana satuan waktu yang dibutuhkan diharapkan dalam waktu yang relatif singkat. Di dalam suatu analisa algoritma untuk menentukan kecepatan prosesnya digunakan istilah running time dan biasanya

dinotasikan dengan T(n). Adapun hal-hal yang mempengaruhi besar kecilnya nilai running time adalah :

a. Banyaknya langkah

Semakin banyak langkah atau instruksi yang digunakan maka semakin besar nilai running time yang dibutuhkan dalam proses tersebut.

b. Ukuran atau besar data

Ukuran atau besar data yang digunakan akan sangat berpengaruh pada proses perhitungan.

c. Jenis operasi

Jenis operasi yang digunakan juga berpengaruh pada running time. Jenis operasi tersebut meliputi operasi aritmetika, operasi nalar atau logika.

Karena terdapat beberapa hal yang dapat mempengaruhi besar kecilnya nilai running time suatu algoritma maka terdapat kemungkinan bahwa di dalam proses penyelesaian masalah suatu algoritma mempunyai kondisi yang berbeda. Oleh karena itu perhitungan running time suatu algoritma dalam proses pencarian solusi suatu masalah dibedakan dalam tiga keadaan, yaitu :

(i). Keadaan terburuk (*Worst Case*)

Merupakan suatu keadaan yang terburuk dari proses di dalam suatu algoritma. Sehingga waktu yang ditempuh oleh algoritma tersebut adalah waktu yang maksimum.

(ii). Keadaan rata-rata (*Average Case*)

Hal ini merupakan suatu keadaan dari running time yang ekivalen dengan nilai harapan dari fungsi running time untuk setiap input yang mungkin. Analisa untuk kondisi ini jarang dilakukan, karena adanya kesulitan dalam menentukan suatu data yang secara umum dapat mewakili keadaan rata-rata data yang digunakan.

(iii). Keadaan Terbaik (*Best Case*)

Merupakan suatu keadaan yang terbaik dari proses di dalam suatu algoritma. Sehingga waktu yang ditempuh oleh algoritma tersebut adalah waktu yang minimum.

Untuk selanjutnya dalam menentukan running time suatu algoritma diasumsikan terjadi keadaan terburuk. Hal ini dilakukan dengan beberapa alasan sebagai berikut :

(1). Keadaan terburuk dapat menerima semua batasan data masukan meskipun untuk masukan yang tidak bagus.

(2). Running time pada keadaan terburuk dari suatu algoritma merupakan batas atas dari running time atau dapat dikatakan sebagai waktu maksimal yang dibutuhkan. Sehingga dengan demikian akan memberikan suatu jaminan bahwa tidak ada running time yang lebih besar.

(3). Untuk beberapa algoritma, keadaan terburuk merupakan kejadian yang sering terjadi. Sebagai contoh adalah dalam pencarian informasi pada suatu basis

data, keadaan terburuk selalu terjadi apabila informasi yang terjadi tidak berada dalam basis data.

Suatu algoritma mempunyai ciri-ciri khusus yang mungkin berbeda satu dengan yang lain, oleh karena itu dibutuhkan identifikasi algoritma secara signifikan. Untuk menganalisa running time suatu algoritma harus diketahui aturan-aturan yang dapat menyederhanakan perhitungan. Aturan-aturan yang diperlukan dalam analisa tersebut adalah :

(1). Running time setiap assigment (tugas), baca (read), dan stateman write besarnya adalah O(1).

(2). Loop

Running time dari suatu loop adalah jumlah iterasi yang dilakukan untuk menyelesaikan statemen-statemen dalam loop tersebut.

(3). Loop berkalang

Jika dalam suatu algoritma terdapat loop berkalang, yaitu loop yang berada di dalam loop, maka yang menjadi pedoman perhitungan adalah loop yang paling dalam. Running time-nya adalah hasil dari perkalian semua ukuran loop.

(4). Statemen berurutan

Apabila terdapat statemen berurutan, maka penentuan running time-nya menggunakan running time maksimal dari running time yang ada.

(5). Statemen if kondisi then statemen

Running time untuk memeriksa kondisi secara normal adalah O(1). Dan untuk statemen adalah running time pada saat statemen dieksekusi (kondisi terpenuhi).

Contoh 2.4.1

Misalkan himpunan A merupakan sebuah array yang terdiri dari n elemen, yaitu A(1), A(2), ..., A(n). Akan dilakukan pengurutan bilangan dengan urutan naik. Variabel key menunjukkan variabel kunci untuk perbandingan. Algoritma pengurutan bilangan tersebut adalah sebagai berikut

Insertion_sort(A)		Times
1	for j ← 2 to n	n
2	do key ← A[j]	n –1
3	{insert A[j] into the sorted sequence A[1..j-1]}	n –1
4	i ← j – I	n –1
5	while i > 0 and A[i] > key	$\sum_{j=2}^{n} t_j$
6	do A[i+1] ← A[i]	$\sum_{j=2}^{n} (t_j - 1)$
7	i← I – 1	$\sum_{j=2}^{n} (t_j - 1)$
8	A[i + 1] ← key	n -1

Dengan times adalah jumlah maksimal eksekusi yang dapat dilakukan untuk setiap statemen dan t_j adalah harga running time statemen pada saat j.

Penjelasan

Running time algoritma pada contoh 2.4.1 adalah jumlah running time dari setiap statemen yang dijalankan.

$T(n) = n + (n-1) + (n-1) + \sum_{j=2}^{n} t_j + \sum_{j=2}^{n}(t_j - 1) + \sum_{j=2}^{n}(t_j - 1) + (n-1)$

Keadaan terbaik terjadi jika array data yang dimasukkan dalam keadaan sudah terurut naik. Apabila terjadi keadaan terbaik, maka hanya statemen pada nomer 5 yang dikerjakan sedangkan statemen 6 dan 7 tidak perlu dikerjakan sehingga $t_j = 1$ untuk $j = 2,3,\ldots,n$ dan running time-nya adalah

$T(n) = n + (n-1) + (n-1) + (n-1) + (n-1)$

$= 5n + 4$

$= O(n)$

Jika array data tersaji dalam keadaan sebaliknya, yaitu tersaji dalam keadaan terurut menurun (urut dari besar ke kecil), maka terjadi keadaan terburuk. Dalam keadaan ini maka elemen A[j] harus dibandingkan dengan setiap elemen terurut pada subarray A[1..j-1] secara menyeluruh, sehingga $t_j = j$ untuk $j = 2,3,\ldots,n$.

Diketahui bahwa

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1 \quad \text{dan} \quad \sum_{j=2}^{n}(j-1) = \frac{n(n-1)}{2}$$

sehingga running time-nya adalah

$$T(n) = n + (n-1) + (n-1) + \left(\frac{n(n+1)}{2} - 1\right) + \left(\frac{n(n-1)}{2}\right) + \left(\frac{n(n-1)}{2}\right) + (n-1)$$

$$= \frac{3}{2}n^2 + \frac{7}{2}n - 4$$

$= O(n^2)$

BAB III

Algoritma Pemrograman Dinamik dalam Penyelesaian Masalah Perkalian Rangkaian Matriks

Pada bagian ini dijelaskan tentang perkalian suatu rangkaian matriks, algoritma pemrograman dinamik dan langkah-langkahnya dalam penyelesaian masalah perkalian rangkaian matriks, serta analisa running time.

3.1. Perkalian Suatu Rangkaian Matriks

Proses penyelesaian perkalian matriks dengan menggunakan komputer kadang kala memerlukan proses yang banyak. Hal ini disebabkan karena di dalam suatu perkalian matriks terjadi operasi perkalian dan penjumlahan elemen-elemen dalam matriks, yaitu operasi antara elemen baris suatu matriks dengan elemen kolom matriks yang sebelah kanan. Dari hal tersebut diketahui bahwa banyaknya elemen atau yang biasa disebut dengan besarnya ukuran matriks sangat berpengaruh terhadap banyaknya proses yang akan dilakukan.

Selain itu apabila banyaknya matriks yang akan dikalikan tersebut lebih dari dua atau terjadi perkalian suatu rangkaian matriks, maka ada satu faktor lagi yang menjadi penentu banyak sedikitnya proses yang akan dilakukan. Faktor yang dimaksud adalah faktor pengambilan keputusan tentang urutan perkalian yang akan dilakukan, yaitu pasangan matriks mana yang harus dikalikan terlebih dahulu. Hal ini dapat dijelaskan seperti berikut ini.

Misalkan diberi suatu barisan ($A_1, A_2, \ldots, A_n$) yaitu suatu barisan sebanyak n matriks untuk dikalikan dan diharapkan untuk mendapat hasil dari bentuk perkalian

$$A_1.A_2.A_3.\ldots.A_n \qquad (3.1.1)$$

Bentuk perkalian 3.1.1 dapat ditentukan nilai hasilnya dengan menggunakan algortima standar untuk mengalikan suatu pasangan matriks. Namun dalam proses penyelesaiannya tentu bentuk perkalian 3.1.1 tidak akan dilakukan secara simultan dalam arti matriks-matriks tersebut tidak akan dikalikan secara bersamaan, akan tetapi pasti akan ditentukan pasangan matriks mana yang akan dikalikan terlebih dahulu. Untuk menentukan pasangan-pasangan mana yang akan dikalikan terlebih dahulu maka bentuk perkalian 3.1.1 tersebut diberi tanda kurung secara penuh.

Suatu bentuk perkalian rangkaian matriks dikatakan diberi tanda kurung secara penuh apabila sebuah matriks tunggal atau hasil perkalian sepasang matriks di dalam suatu tanda kurung, dilingkupi dengan tanda kurung. Karena operasi perkalian matriks bersifat assosiatif, maka banyaknya pilihan dalam menentukan letak tanda kurung dalam suatu bentuk perkalian rangkaian matriks adalah sebagai berikut

$$p(n) = \begin{cases} 0 & \text{, jika } n = 0 \\ 1 & \text{, jika } n = 1 \\ \sum_{k=1}^{n-1} p(k).p(n-k) & \text{, jika } n > 1 \end{cases}$$

Selanjutnya untuk $n > 1$ akan berlaku

$$\sum_{k=1}^{n-1} p(k).p(n-k) = \frac{1}{n}\binom{2n-2}{n-1} \qquad \text{, jika } n > 1$$

dengan p(n) adalah banyaknya pilihan peletakan tanda kurung dan n adalah banyaknya matriks dalam rangkaian.

Sebagai penjelasannya, berikut ini diberikan langkah untuk mendapatkan bentuk persamaan p(n) tersebut.

Misalkan diambil p(n) sebagai banyaknya pilihan peletakan tanda kurung pada suatu bentuk perkalian rangkaian matriks dengan banyaknya matriks yang akan dikalikan adalah sebanyak n. Maka untuk menyelesaikannya bentuk perkalian rangkaian matriks tersebut dapat dipecah di antara matriks ke- k dan ke- k+1 untuk k = 1,2, …,n-1, hal ini dapat dilakukan karena sifat perkalian matriks yang asosiatif. Oleh karena itu untuk mencari banyaknya cara dalam menentukan letak tanda kurung pada bentuk perkalian rangkaian matriks tersebut, maka dapat dicari dahulu pada bagian k matriks di satu pihak dan pada bagian n-k matriks di lain pihak, kemudian kedua hasil yang diperoleh dikalikan. Proses pencarian ini dapat dilakukan untuk nilai k = 1,2,…,n-1. Kemudian dapat diperhatikan bahwa jika n = 0 (tidak ada matriks yang akan dikalikan) maka p(0) = 0 dan jika n =1 (hanya ada 1 matriks dalam bentuk perkalian rangkaian matriks) maka hanya ada satu pilihan letak tanda kurung tersebut sehingga p(1) = 1. Dengan demikian diperoleh rumus hubungan rekursif

$$p(n) = \begin{cases} 0 & , \text{jika } n = 0 \\ 1 & , \text{jika } n = 1 \\ \sum_{k=1}^{n-1} p(k).p(n-k) & , \text{jika } n > 1 \end{cases} \quad \ldots(3.1.2)$$

Misalkan g(x) adalah fungsi pembangkit dari p(n) dengan x adalah suatu variabel yang sembarang maka g(x) akan berbentuk

$$g(x) = \sum_{n=0}^{\infty} p(n).x^n \qquad \ldots(3.1.3)$$

$$= p(0) + p(1).x + \sum_{n=2}^{\infty} p(n).x^n$$

$$= 0 + x + \sum_{n=2}^{\infty} p(n).x^n$$

$$g(x) - x = \sum_{n=2}^{\infty} p(n).x^n \qquad \ldots(3.1.4)$$

Kemudian dapat diperhatikan bahwa kuadrat dari g(x) akan mempunyai bentuk sebagai berikut

$$g(x).g(x) = (0 + p(1).x + p(2).x^2 + p(3).x^3 + \ldots)^2 \qquad \ldots(3.1.5)$$

Dengan menggunakan metode penjumlahan polinomial, maka akan diperoleh persamaan fungsi pembangkit berikut ini.

$$g(x) - x = \sum_{n=2}^{\infty} p(n).x^n$$

$$= \sum_{n=2}^{\infty} (p(1).p(n-1) + p(2).p(n-2) + \ldots + p(n-1)p(1))x^n$$

$$= (p(1).x + p(2).x^2 + p(3).x^3 + \ldots)^2$$

$$= (g(x))^2 \qquad \ldots(3.1.6)$$

Persamaan fungsi pembangkit (3.1.6) tersebut dapat diselesaikan dengan menggunakan rumus kuadrat a,b,c yang biasa digunakan untuk menyelesaikan $ay^2 + by + c = 0$ dimana dalam hal ini a = 1, b =1, c = -x, dan y = g(x).

sehingga diperoleh

$$g(x) = \frac{1}{2}(1 \pm \sqrt{1-4x})$$

dan karena syarat awal p(0) = 0, maka fungsi yang memenuhi syarat ini adalah

$$g(x) = \frac{1}{2}(1 - \sqrt{1-4x}) \qquad \ldots(3.1.7)$$

Dengan menggunakan rumus ekspansi binomial newton, maka ekspansi dari $\sqrt{1-4x} = (1-4x)^{\frac{1}{2}}$ dengan | -4x | < 1 akan mempunyai bentuk

$$\sqrt{1-4x} = \sum_{n=0}^{\infty} \binom{1/2}{n} (-4x)^n \qquad \ldots(3.1.8)$$

Dengan demikian bentuk umum koefisien binomial untuk pecahan ½ adalah

- untuk n = 0

$$\binom{1/2}{n} = \binom{1/2}{0} = 1$$

- untuk n > 0

$$\binom{1/2}{n} = \frac{(\frac{1}{2})!}{(\frac{1}{2}-n)!.n!}$$

$$= \frac{\frac{1}{2}.(\frac{1}{2}-1).(\frac{1}{2}-2)...(\frac{1}{2}-(n-1))}{n!}$$

$$= \frac{(\frac{1}{2}).(-\frac{1}{2}).(-\frac{2}{3})...(-\frac{1}{2}(2n-3))}{n!}$$

$$= \frac{(-1)^{n-1}}{2^n}.\frac{1.3.5....(2n-3)}{n!}$$

$$= \frac{(-1)^{n-1}}{2^n}.\frac{1.2.3.4.5.6.....(2n-3).(2n-2)}{2.4.6...(2n-2).n!}$$

$$\binom{1/2}{n} = \frac{(-1)^{n-1}}{n.2^{2n-1}} \cdot \frac{(2n-2)!}{(n-1)!.(n-1)!}$$
$$= \frac{(-1)^{n-1}}{n.2^{2n-1}} \cdot \binom{2n-2}{n-1} \quad \ldots(3.1.9)$$

sehingga bentuk persamaan (3.1.8) menjadi

$$\sqrt{1-4x} = 1.(-4x)^0 + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n.2^{2n-1}} \binom{2n-2}{n-1} (-4x)^n$$
$$= 1 + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n.2^{2n-1}} \binom{2n-2}{n-1} (-1)^n (2^2)^n (x)^n$$
$$= 1 + \sum_{n=1}^{\infty} \frac{(-2)}{n.} \binom{2n-2}{n-1} (x)^n$$
$$= 1 - 2\sum_{n=1}^{\infty} \frac{1}{n.} \binom{2n-2}{n-1} (x)^n \quad \ldots(3.1.10)$$

dengan memasukkan bentuk terakhir $\sqrt{1-4x}$ dari persamaan (3.1.10) ke persamaan (3.1.7) akan diperoleh

$$g(x) = \frac{1}{2}(1 - (1 - 2\sum_{n=1}^{\infty} \frac{1}{n.} \binom{2n-2}{n-1} (x)^n))$$
$$= \sum_{n=1}^{\infty} \frac{1}{n.} \binom{2n-2}{n-1} (x)^n \quad \ldots(3.1.11)$$

Sehingga dari bentuk persamaan (3.1.3) dan (3.1.11) didapatkan

$$\sum_{n=0}^{\infty} p(n).x^n = \sum_{n=1}^{\infty} \frac{1}{n.} \binom{2n-2}{n-1} (x)^n$$
$$0 + \sum_{n=1}^{\infty} p(n).x^n = \sum_{n=1}^{\infty} \frac{1}{n.} \binom{2n-2}{n-1} (x)^n$$

dari bentuk persamaan yang terakhir didapatkan $p(n) = \frac{1}{n} \binom{2n-2}{n-1}$

keterangan :

p(n) : banyaknya pilihan dalam menentukan letak tanda kurung untuk rangkaian dengan anggota sebanyak n matriks

n : banyaknya matriks dalam rangkaian

g(x) : fungsi pembangkit dari p(n)

x : sembarang variabel

Contoh 3.1.1.

Jika diberikan barisan matriks (A_1, A_2 ,A_3, A_4), maka bentuk perkalian matriks $A_1.A_2.A_3.A_4$ dapat diberi tanda kurung secara penuh dalam lima cara yang berbeda yaitu :

(A_1.(A_2.(A_3.A_4)))

(A_1.((A_2.A_3).A_4))

((A_1.A_2).(A_3.A_4))

((A_1.(A_2.A_3)).A_4)

(((A_1.A_2).A_3).A_4)

Perbedaan cara dalam peletakan tanda kurung memberikan pengaruh pada banyaknya proses yang akan dilakukan untuk menyelesaikan perkalian tersebut. Banyaknya proses yang akan dilakukan oleh suatu cara peletakan akan berbeda dengan banyaknya proses yang akan dilakukan oleh cara peletakan yang lain. Perbedaan banyaknya proses tersebut ternyata cukup besar. Untuk mengetahui adanya perbedaan banyak proses yang akan dilakukan karena cara peletakan tanda kurung yang berbeda, berikut ini disajikan algoritma perkalian matriks.

Pseudocode Perkalian_matriks (A,B)

```
1. if jumlah_kolom_[A] ≠ jumlah_baris_[B]
2.         then error "dimensi tidak sesuai"
3.         else for i ← 1 to jumlah_baris_[A]
4.                 do for j ← 1 to jumlah_kolom_[B]
5.                     do c[i,j] ← 0
6.                     for k ← 1 to jumlah_kolom_[A]
7.                         do c[i,j] ← c[i,j] + A[i,k].B[k,j]
8. return c
```

Dari algoritma perkalian_matriks(A,B) diketahui bahwa perkalian dua buah matriks A dan B dapat dilakukan jika jumlah kolom matriks A sama dengan jumlah baris matriks B. Misalkan A matriks yang berukuran p x q dan B matriks yang berukuran q x r, maka matriks hasilnya yaitu C adalah matriks berukuran p x r. Sedangkan banyaknya proses yang akan dilakukan untuk menghasilkan matriks C didominasi oleh banyaknya perkalian skalar pada baris ke- 7 algoritma perkalian_matriks(A,B) yaitu sebanyak p.q.r.

Untuk menunjukkan perbedaan banyak proses yang akan dilakukan sebagai akibat dari perbedaan peletakan tanda kurung, diberikan suatu barisan matriks (A_1, A_2, A_3, A_4). Misalkan ukuran dari matriks-matriks tersebut secara urut adalah 30 x 1, 1 x 40, 40 x 10, dan 10 x 25, maka banyaknya perkalian skalar yang dilakukan untuk masing-masing bentuk peletakan tanda kurung adalah sebagai berikut :

a) $(A_1.(A_2.(A_3.A_4)))$: Perkalian $(A_3.A_4)$ mengerjakan 40.10.25 = 10.000 perkalian skalar dan menghasilkan matriks 40 x 25. Perkalian matriks A_2 dengan matriks hasil kali $(A_3.A_4)$ mengerjakan 1.40.25 = 1.000 perkalian skalar dan menghasilkan matriks 1 x 25. Sedangkan perkalian yang terakhir mengerjakan 30.1.25 = 750 perkalian skalar. Sehingga banyaknya perkalian skalar yang dikerjakan adalah 10.000 + 1.000 + 750 = 11.750 perkalian skalar.

b) $(A_1.((A_2.A_3).A_4))$: Perkalian $(A_2.A_3)$ mengerjakan 1.40.10 = 400 perkalian skalar dan menghasilkan matriks 1 x 10. Perkalian matriks hasil dengan matriks A_4 mengerjakan 1.10.25 = 250 perkalian skalar dan menghasilkan matriks 1 x 25. Perkalian yang terakhir mengerjakan 30.1.25 = 750 perkalian skalar. Sehingga banyaknya perkalian skalar yang dikerjakan adalah 400 + 250 + 750 = 1.400 perkalian skalar.

c) $((A_1.A_2).(A_3.A_4))$: Perkalian $(A_1.A_2)$ mengerjakan 30.1.40 = 1.200 perkalian skalar dan menghasilkan matriks 30 x 40. Sedangkan perkalian $(A_3.A_4)$ mengerjakan 40.10.25 = 10.000 perkalian skalar dan menghasilkan matriks 40 x 25. Perkalian antara dua matriks hasil mengerjakan 30.40.25 = 30.000 perkalian skalar, sehingga banyaknya perkalian skalar yang dikerjakan adalah 1.200 + 10.000 + 30.000 = 41.200 perkalian skalar.

d) $((A_1.(A_2.A_3)).A_4)$: Perkalian $(A_2.A_3)$ mengerjakan 400 perkalian skalar dan perkalian matriks A_1 dengan matriks hasil kali $(A_2.A_3)$ mengerjakan 30.40.10 = 12.000 perkalian skalar. Sedangkan perkalian yang terakhir mengerjakan 30.10.25 = 7.500 perkalian skalar, sehingga jumlah perkalian skalar yang dikerjakan adalah 400 + 12.000 + 7.500 = 19.900 perkalian skalar.

e) $(((A_1.A_2).A_3).A_4)$: Perkalian $(A_1.A_2)$ mengerjakan 30.1.40 = 1200 perkalian skalar dan menghasilkan matriks 30 x 40. Perkalian matriks hasil dengan matriks A_3 mengerjakan 30.40.10 = 12.000 perkalian skalar dan menghasilkan matriks 30 x 10. Sedangkan untuk perkalian yang terakhir mengerjakan 30.10.25 = 7.500 perkalian skalar. Jadi banyaknya perkalian skalar yang dikerjakan adalah sebanyak 1200 + 12.000 + 7.500 = 20.700 perkalian skalar.

3.2. Algoritma Pemrograman Dinamik

Algoritma pemrograman dinamik adalah suatu langkah-langkah yang berisi tehnik matematis yang digunakan untuk membuat suatu keputusan dari serangkaian keputusan yang saling berkaitan. Tujuan utama penggunaan algoritma pemrograman dinamik dalam menyelesaikan suatu masalah adalah untuk mempermudah masalah optimasi yang mempunyai karakteristik tertentu. Sehingga dengan digunakannya algoritma pemrograman dinamik ini diharapkan untuk memperoleh solusi yang optimal dari beberapa solusi yang mungkin dihasilkan. Sedangkan solusi optimal yang diambil bisa berupa nilai maksimum atau nilai minimum, hal ini tergantung dari kondisi permasalahan yang dihadapi.

Ide dasar algoritma pemrograman dinamik dalam menyelesaikan suatu masalah adalah sama dengan metode devide and conquer yaitu memecah masalah menjadi beberapa sub masalah yang lebih kecil sehingga memudahkan penyelesaiannya. Kemudian mengkombinasikan solusi-solusi yang diperoleh untuk menyelesaikan masalah yang sesungguhnya. Pemecahan masalah menjadi

sub masalah yang lebih kecil dilakukan sampai didapatkan sub masalah yang paling kecil. Tetapi di dalam prosesnya algoritma pemrograman dinamik ini tidak mempunyai suatu formula matematis yang standar. Oleh karena itu persamaan-persamaan yang digunakan harus dikembangkan agar dapat memenuhi kondisi yang ada.

Tidak semua permasalahan dapat diselesaikan dengan menggunakan algoritma pemrograman dinamik. Algoritma pemrograman dinamik hanya efektif untuk menyelesaikan suatu permasalahan yang mempunyai ciri tertentu. Keistimewaan dasar yang mencirikan masalah pemrograman dinamik diberikan berikut ini.

1. Permasalahan dapat dibagi dalam beberapa sub masalah yang disebut tahap-tahap di mana suatu keputusan kebijakan atau solusi dibutuhkan di setiap tahap. Pengambilan keputusan pada masing-masing tahap tidaklah secara serentak tetapi tahap demi tahap.
2. Setiap tahap memiliki keadaan (state) yang merupakan berbagai kondisi yang mungkin di mana sistem berada pada tahap tertentu dari keseluruhan permasalahan.
3. Pada setiap tahap, keputusan kebijakan dihubungkan dengan tahap yang berdekatan melalui fungsi transisi.
4. Keputusan kebijakan pada setiap tahap berpengaruh untuk merubah keadaan sekarang menjadi keadaan yang berkaitan dengan tahap sebelumnya atau tahap berikutnya. Hal ini disebabkan oleh adanya suatu return function.

5. Terdapat hubungan rekursif yang mengindentifikasi kebijakan optimal pada tahap t bila diketahui kebijakan optimal untuk tahap t+1 atau t-1. Dari hubungan rekursif yang ada, maka untuk menyelesaikan suatu masalah pemrograman dinamik , terdapat dua macam prosedur rekursif, yaitu :

a. Prosedur rekursif maju

Prosedur rekursif maju dalam menyelesaikan masalah akan bergerak dari depan ke belakang yaitu penyelesaian akan dimulai dari tahap 1 kemudian bergerak maju ke tahap 2 dan seterusnya tahap demi tahap sehingga misalkan terdapat sebanyak m tahap, maka proses akan berakhir pada tahap m. Pada setiap tahap akan dicari nilai optimum return function yang merupakan hasil operasi antara nilai kontribusi tahap bersangkutan dengan nilai optimum dari return function dari tahap sebelumnya. Oleh karena itu jika prosedur rekursif maju ini digunakan maka sebelumnya akan ditetapkan terlebih dahulu nilai optimum return function dari tahap nol yaitu sebesar 0 (nol). Hal ini disebabkan karena pada setiap tahap, prosedur rekursif maju membutuhkan nilai optimal return function dari tahap sebelumnya. Sehingga bentuk hubungan rekursif yang ada dapat ditulis menjadi

$f_0(x_0) = 0$

$f_t(x_t) = \text{opt}\ \{R_t(k_t)\ @\ f_{t-1}(x_t\ @\ k_t)\}$

b. Prosedur rekursif mundur

Berbeda dengan prosedur rekursif maju, prosedur rekursif mundur di dalam prosesnya membutuhkan nilai optimum return function dari tahap sesudahnya ($f_{t+1}(y_{t+1})$). Hal ini disebabkan karena prosedur rekursif mundur ini bergerak

dari belakang ke depan. Pada setiap tahap akan dicari nilai optimum return function yang merupakan hasil operasi antara nilai kontribusi pada tahap bersangkutan dengan nilai optimum return function dari tahap sesudahnya. Misalkan terdapat sebanyak m tahap maka pertama kali akan ditetapkan nilai optimum return function pada tahap m+1 sebesar 0 (nol). Perhitungan dimulai dari tahap m kemudian bergerak mundur ke tahap m-1 dan seterusnya tahap demi tahap sampai berakhir pada tahap 1. Sehingga bentuk hubungan rekursif yang ada dapat ditulis menjadi

$f_{m+1}(y_{m+1}) = 0$

$f_t(y_t) = \text{opt} \{ R_t(k_t) @ f_{t+1}(y_t @ k_t)\}$

Keterangan :

f(x) / f(y)	= optimum return function
x atau y	= state atau keadaan
k	= alternatif keputusan
R_t	= nilai kontribusi pada tahap t
$x_t @ k_t$ atau $y_t @ k_t$	= fungsi transisi
@	= simbol operasi matematik (misal : x, ÷, +, -, ...)
t	= tahap ke

Untuk pembahasan selanjutnya yaitu dalam penyelesaian masalah perkalian rangkaian matriks, prosedur yang digunakan adalah prosedur rekursif mundur.

6. Dengan menggunakan hubungan rekursif ini prosedur penyelesaian bergerak dari tahap ke tahap sampai kebijaksanaan optimum pada tahap terakhir ditemukan

Dari keistimewaan dasar yang menjadi ciri permasalahan, maka dalam proses menyelesaikan suatu masalah akan terdapat empat langkah penyelesaian. Langkah-langkah tersebut yaitu :

1. Menentukan struktur dari suatu solusi yang optimal

   Pada langkah pertama ini akan diusahakan untuk menentukan struktur dari suatu solusi yang optimal. Hal yang dilakukan adalah mencari tahap-tahap penyelesaian masalah dimana pada setiap tahap akan dicari solusi terbaik yang dapat menjamin keoptimalan hasil secara keseluruhan. Langkah ini dilakukan dengan memecah masalah menjadi sub-sub masalah yang lebih kecil sesuai dengan konsep devide and conquer.

2. Mencari persamaan rekursif

   Langkah kedua ini berusaha untuk mencari suatu persamaan rekursif yang disusun berdasarkan pada hubungan timbal balik antara suatu tahap dengan tahap sebelumnya atau tahap berikutnya. Dengan formula matematis ini diharapkan untuk mendapatkan suatu nilai yang dapat dijadikan sebagai pertimbangan dalam menentukan keputusan kebijakan pada setiap tahap.

3. Perhitungan dengan menggunakan persamaan rekursif

   Setelah persamaan rekursif ditemukan, maka pada langkah ketiga algoritma pemrograman dinamik akan melakukan perhitungan dengan menggunakan persamaan rekursif tersebut. Dalam proses perhitungan ini nilai yang diperoleh pada suatu tahap akan digunakan untuk membantu dalam perhitungan pada tahap yang lainnya. Dengan didapatkannya nilai pada setiap

tahap, maka keputusan kebijakan untuk masing-masing tahap dapat ditentukan.

4. Membangun suatu solusi yang optimal

Pada langkah ini algoritma pemrograman dinamik berusaha membangun suatu solusi yang optimal berdasarkan keputusan kebijakan yang diambil.

3.3. Penyelesaian Masalah Perkalian Rangkaian matriks

3.3.1. Struktur Peletakan Tanda Kurung yang Optimal

Langkah pertama algoritma pemrograman dinamik dalam menyelesaikan suatu masalah adalah berusaha untuk menentukan struktur dari solusi yang optimal. Dalam penyelesaian masalah perkalian rangkaian matriks, langkah pertama tersebut dilakukan sebagai berikut. Pemberian tanda kurung secara penuh pada bentuk perkalian matriks $A_1.A_2...A_n$ akan memecah bentuk perkalian tersebut di antara A_k dan A_{k+1} dengan k adalah suatu bilangan bulat positif yang terletak pada interval $1 \leq k < n$, sehingga akan terbentuk sub perkalian rangkaian matriks $A_1.A_2...A_k$ dan $A_{k+1}.A_{k+2}...A_n$. Kemudian untuk suatu nilai k bentuk perkalian matriks $A_1.A_2...A_k$ dan $A_{k+1}.A_{k+2}...A_n$ akan diproses dan matriks yang dihasilkan dari kedua perkalian tersebut dikalikan untuk menghasilkan matriks yang diharapkan.

Karena pemberian tanda kurung adalah secara penuh maka pemecahan perkalian rangkaian matriks menjadi sub perkalian rangkaian matriks akan terus dilakukan sampai didapatkan sub perkalian rangkaian mariks yang terkecil yang berupa perkalian sepasang matriks. Hal yang perlu diperhatikan dalam langkah

pertama ini adalah bahwa peletakan tanda kurung pada sub rangkaian pertama yaitu $A_1.A_2...A_k$ dalam pemberian tanda kurung secara penuh yang optimal pada bentuk perkalian $A_1.A_2...A_n$ harus merupakan suatu peletakan tanda kurung yang optimal pada sub rangkaian tersebut. Letak tanda kurung yang diberikan harus menyebabkan banyaknya proses yang akan dilakukan dalam penyelesaian masalah adalah yang paling sedikit. Sedangkan dasar dalam penentuan nilai k akan dibahas pada subbab 3.3.2 dan proses pencarian nilai k tersebut dibahas pada subbab 3.3.3.

Hal yang sama juga harus terjadi pada sub rangkaian $A_{k+1}.A_{k+2}...A_n$ yaitu peletakan tanda kurung pada sub rangkaian ini juga harus optimal. Selain itu sesuai dengan sifat algoritma pemrograman dinamik dalam menyelesaikan masalah yaitu memecah masalah sampai menghasilkan sub masalah terkecil, maka peletakan tanda kurung yang optimal harus dipenuhi sampai dengan sub masalah yang terkecil. Sehingga dapat dilihat bahwa di dalam suatu perkalian rangkaian matriks, peletakan tanda kurung akan optimal jika peletakan tanda kurung pada sub perkalian rangkaian matriksnya juga optimal. Oleh karena itu untuk mendapatkan suatu peletakan tanda kurung yang optimal pada suatu perkalian rangkaian matriks terlebih dahulu dicari peletakan tanda kurung yang optimal pada sub perkalian rangkaian matriksnya.

3.3.2 Mencari Persamaan Rekursif

Seperti yang telah dijelaskan pada sub bab 3.1.1 bahwa setiap peletakan tanda kurung yang berbeda menyebabkan banyak proses yang berbeda satu sama lain yang disebabkan oleh adanya perbedaan jumlah perkalian skalar yang

dilakukan. Oleh karena itu dalam menentukan peletakan tanda kurung yang optimal akan dicari suatu nilai yang mewakili banyaknya perkalian skalar yang minimal. Misalkan diambil notasi $A_{i..j}$ sebagai matriks yang dihasilkan dari proses perhitungan bentuk perkalian rangkaian matriks $A_i.A_{i+1}...A_j$ dan f[i,j] adalah nilai minimal dari banyaknya perkalian skalar yang akan dilakukan untuk menghitung bentuk perkalian matriks $A_{i..j}$ dengan $1 \leq i \leq j \leq n$, maka banyaknya perkalian skalar yang akan dilakukan pada kondisi terbaik untuk menghitung bentuk perkalian matriks $A_{i...n}$ adalah f[1,n].

Nilai dari f[i,j] didefinisikan secara rekursif sebagai berikut :

a. Jika i = j maka rangkaian yang dimaksud terdiri atau tersusun dari hanya satu matriks. Dengan demikian dapat dikatakan $A_{i..i} = A_i$, hal ini berarti tidak ada perkalian matriks dan tidak dibutuhkan perkalian skalar, sehingga nilai f[i,j] = 0.

b. Jika i < j maka diasumsikan bahwa pemberian tanda kurung memecah bentuk perkalian $A_iA_{i+1}...A_j$ di antara A_k dan A_{k+1} dengan $i \leq k < j$. Sehingga nilai f[i,j] adalah nilai terkecil dari banyaknya perkalian skalar untuk menghitung sub rangkaian $A_i.A_{i+1}...A_k$ dan $A_{k+1}.A_{k+2}...A_j$ ditambah banyaknya perkalian skalar untuk mengalikan $A_{i..k}$ dan $A_{k+1..j}$ yang merupakan dua matriks yang dihasilkan dari dua sub perkalian rangkaian matriks tersebut. Jika diasumsikan A_i mempunyai ukuran p_{i-1} x p_i untuk i = 1,2,...,n maka $A_{i..k}$ mempunyai ukuran p_{i-1} x p_k dan $A_{k+1..j}$ mempunyai ukuran p_k x p_j. Sehingga untuk mengalikan $A_{i..k}$ dan $A_{k+1..j}$ membutuhkan

sebanyak $p_{i-1}.p_k.p_j$ perkalian skalar sebagaimana yang telah dijelaskan pada sub bab 3.1 dan pada akhirnya didapatkan

$$f[\, i,j\,] = f[\, i,k\,] + f[\, k+1,j\,] + p_{i-1}.p_k.p_j \qquad (3.3.1)$$

Sebagaimana telah disebutkan pada sub bab 3.3.1 bahwa untuk mendapatkan peletakan tanda kurung yang optimal pada suatu perkalian rangkaian matriks maka terlebih dahulu harus ditemukan peletakan tanda kurung yang optimal pada sub perkalian rangkaian matriksnya. Oleh karena itu nilai f[i,k] dan f[k+1,j] pada persamaan 3.3.1 adalah sudah ditemukan sebelumnya.

Persamaan 3.3.1 ini mengasumsikan bahwa nilai dari k adalah tertentu atau sudah diketahui padahal sebenarnya belum diketahui. Di dalam masalah perkalian rangkaian matriks ini terdapat sebanyak j – i kemungkinan harga k, dengan kemungkinan harga k tersebut adalah k = i, k = i+1, k = k+2, …, k = j - 2, atau k = j – 1. Karena peletakan tanda kurung yang optimal harus menggunakan salah satu dari harga k, maka yang perlu dilakukan adalah memeriksa semua kemungkinan nilai k untuk menemukan peletakan tanda kurung yang terbaik. Sehingga definisi rekursif untuk f[i,j] dalam pemberian tanda kurung secara penuh pada bentuk perkalian rangkaian matriks $A_i.A_{i+1}\ldots Aj$ dapat ditulis :

$$f[i,j] = \begin{cases} 0 & , jika \quad i = j \\ \min\limits_{i \le k < j} \{p_{i-1}.p_k.p_j + f[i,k] + f[k+1,j]\} & , jika \quad i < j \end{cases} \qquad (3.3.2)$$

Bentuk persamaan 3.3.2. telah sesuai dengan bentuk persamaan rekursif pada prosedur rekursif mundur. Hal ini dapat dilihat dengan mengasumsikan bahwa $f_t(y_t) = f[i,j]$, $(y_t) = [i,j]$, $R_t(k_t) = p_{i-1}.p_k.p_j$, dan $f_{t+1}(y_t \,@\, k_t) = f[i,k]+f[k+1,j]$. Sedangkan nilai 0 jika i = j sesuai dengan nilai $f_{m+1}(y_{m+1}) = 0$. Hal ini terjadi

dengan menganggap bahwa dalam permasalahan terdapat sebanyak m tahap dan tahap m+1 merupakan tahap perkalian rangkaian matriks dengan anggota sebanyak satu. Untuk membantu dalam menemukan harga k agar terbentuk solusi yang optimal, maka didefinisikan s[i,j] sebagai suatu harga k yang memecah bentuk perkalian $A_i.A_{i+1}...A_j$ untuk menghasilkan nilai f[i,j] yang optimal.

3.3.3. Perhitungan dengan Menggunakan Persamaan Rekursif

Pada bagian ini akan dihitung nilai dari banyaknya perkalian skalar yang terkecil untuk semua kemungkinan yang ada dan menentukan nilai s[i,j] dengan menggunakan persamaan rekursif 3.3.2. Langkah yang dikerjakan dengan menggunakan prosedur perhitungan mundur. Sehingga dari suatu bentuk perkalian rangkaian matriks, pertama kali akan dihitung banyaknya perkalian skalar pada sub perkalian rangkaian matriks terkecil yang berupa perkalian sepasang matriks. Hal ini dilakukan pada semua kemungkinan yang ada. Setelah hasilnya diperoleh, maka proses perhitungan akan bergerak ke depannya yaitu dengan melakukan proses perhitungan pada sub rangkaian matriks yang lebih besar dan dilakukan perhitungan dengan langkah yang sama. Langkah tersebut dilakukan sampai pada sub rangkaian terbesar sehingga langkah akhirnya akan dilakukan perkalian antara dua matriks yang dihasikan dari dua sub rangkaian terbesar tersebut.

Untuk melakukan proses perhitungan ini akan digunakan dua cara yaitu cara manual dan cara komputerisasi. Perhitungan secara manual sengaja diberikan untuk memberikan gambaran kerja dari proses yang akan berlangsung dalam cara komputerisasi. Dengan cara manual ini akan diketahui bagaimana mendapatkan

suatu nilai dari banyaknya perkalian skalar yang optimal (minimal) dan terpilihnya nilai s[i,j] sebagai nilai k yang tepat.

Berikut ini akan dilakukan perhitungan secara manual. Dengan menggunakan contoh kasus pada sub bab 3.1 yaitu terdapat perkalian rangkaian matriks $A_1.A_2A_3A_4$ dengan ukuran masing-masing matriks adalah

matriks	Ukuran
A_1	30 x 1
A_2	1 x 40
A_3	40 x 10
A_4	10 x 25

dan akan dicari letak tanda kurung yang optimal, maka permasalahan dapat dirumuskan

1. Tahap ke t mewakili perkalian rangkaian matriks dengan banyakya anggota adalah n-t+1 = 4 – t + 1, sehingga setiap tahap dapat dijabarkan
   a) Tahap 1 : perkalian rangkaian matriks dengan anggota sebanyak 4 –1+ 1= 4
   b) Tahap 2 : perkalian rangkaian matriks dengan anggota sebanyak 4 –2+ 1= 3
   c) Tahap 3 : perkalian rangkaian matriks dengan anggota sebanyak 4 –3+ 1= 2
   d) Tahap 4 : perkalian rangkaian matriks dengan anggota sebanyak 4 –4+ 1= 1
2. Keadaan tahap t (y_t) mewakili kemungkinan perkalian rangkaian matriks yang bisa terjadi pada tahap t dan dinyatakan dengan i,j dengan i adalah posisi matriks awal dan j adalah posisi matriks akhir dalam perkalian rangkaian matriks tersebut.

3. Alternatif tahap t (k_t) mewakili posisi tanda kurung atau tempat terjadinya pemisahan dan s[i,j] adalah nilai k yang tepat.
4. $R_t(k_t)$ mewakili perkalian sekalar yang dibutuhkan untuk mengalikan dua matriks ($R_k = p_{i-1}.p_k.p_j$).
5. $f_t(y_t)$ mewakili nilai minimal dari banyaknya perkalian skalar pada tahap t dengan kondisi y_t.
6. Bentuk persamaan rekursif mundur yang ada adalah

$$f_4[y_4] = f[i_4,j_4] = 0$$

$$f_t[y_t] = f_t[i_t,j_t] = \min_{i_t \le k_t < j_t} \{ R_{kt}(k_t) + f_{t+1}[i_t,k_t] + f_{t+1}[k_t+1,j_t] \}$$

Dengan menggunakan prosedur rekursif mundur maka proses perhitungan yang ada adalah sebagai berikut :

Tahap 4

$$f_4[y_4] = f[i_4,j_4] = 0$$

Tabel 3.1. Tabel perhitungan untuk tahap 4

y_4	$f_4(y_4) = f_4(i_4,j_4)$
1,1	0
2,2	0
3,3	0
4,4	0

Tahap 3

$$f_3[y_3] = f_3[i_3,j_3] = \min_{i_3 \le k_3 < j_3} \{ R_3(k_3) + f_4(i_3,k_3) + f_4(k_3+1,j_3) \}$$

Tabel 3.2. Tabel perhitungan untuk tahap 3

Y_3	$R_3(k_3)+f_4(i_3,k_3)+f_4(k_3+1,j_3)$			$f_3[y_3]=f_3[i_3,j_3]$	$s_3[i_3,j_3]$
	$k_3=1$	$k_3=2$	$k_3=3$		
1,1	-	-	-	0	-
2,2	-	-	-	0	-
3,3	-	-	-	0	-
4,4	-	-	-	0	-
1,2	30.1.40+0+0=1200	-	-	1.200	1
2,3	-	1.40.10+0+0=400	-	400	2
3,4	-	-	40.10.25+0+0=10.000	10.000	3

Tahap 2

$$f_2[y_2] = f_2[i_2,j_2] = \min_{i_2 \le k_2 < j_2} \{ R_2(k_2) + f_3[i_2,k_2] + f_3[k_2+1,j_2] \}$$

Tabel 3.3. Tabel perhitungan untuk tahap 2

y_2	$R_2(k_2)+f_3[i_2,k_2]+f_3[k_2+1,j_2]$			$f_2[y_2]=f_2[i_2,j_2]$	$s_2[i_2,j_2]$
	$k_2=1$	$k_2=2$	$k_2=3$		
1,1	-	-	-	0	-
2,2	-	-	-	0	-
3,3	-	-	-	0	-
4,4	-	-	-	0	-
1,2	30.1.40+0+0=1200	-	-	1.200	1
2,3	-	1.40.10+0+0=400	-	400	2
3,4	-	-	40.10.25+0+0=10.000	10.000	3
1,3	30.1.10+0+400=700	30.40.10+1200+0=13.200	-	700	1
2,4	-	1.40.25+0+10000=11.000	1.10.25+400+0=650	650	3

Tahap 1

$$f_1[y_1] = f_1[i_1,j_1] = \min_{i_1 \le k_1 < j_1} \{ R_1(k_1) + f_2[i_1,k_1] + f_2[k_1+1,j_1] \}$$

Tabel 3.4. Tabel perhitungan untuk tahap 1

y_1	$R_1(k_1) + f_2[i_1,k_1] + f_2[k_1+1,j_1]$			$f_1[y_1]=f_1[i_1,j_1]$	$s_1[i_1,j_1]$
	$k_1 = 1$	$k_1 = 2$	$k_1 = 3$		
1,1	-	-	-	0	-
2,2	-	-	-	0	-
3,3	-	-	-	0	-
4,4	-	-	-	0	-
1,2	30.1.40+0+0=1200	-	-	1.200	1
2,3	-	1.40.10+0+0=400	-	400	2
3,4	-	-	40.10.25+0+0=10.000	10.000	3
1,3	30.1.10+0+400=700	30.40.10+1200+0=13.200	-	700	1
2,4	-	1.40.25+0+10000=11.000	1.10.25+400+0=650	650	3
1,4	30.1.25+0+650=1400	30.40.25+1200+10000=41200	30.10.25+700+0=8200	1.400	1

Setelah diketahui proses perhitungan dengan cara manual, berikut ini akan disajikan pseudocode order rangkaian matriks untuk melakukan proses perhitungan harga perkalian skalar yang optimal dan menentukan s[i,j] dengan menggunakan bantuan komputer. Pada pseudocode tersebut diasumsikan bahwa matriks A_i mempunyai ukuran p_{i-1} x p_i dengan i = 1,2,…,n. sedangkan sebagai masukan dalam proses perhitungan ini adalah suatu himpunan P = $\{p_0,p_1,p_2,\ldots,p_n\}$ yang merupakan nilai ukuran matriks yang ada dengan banyaknya anggota himpunan P = n + 1. Pseudocode ini menggunakan sebuah tabel pembantu f[1..n, 1..n] untuk menyimpan harga f[i,j] dan sebuah tabel pembantu s[1..n,1..n] yang menyimpan nilai k yang didapatkan dari harga optimal dalam perhitungan f[i,j].

Pseudocode order rangkaian matriks(p)

1. n ← panjang [p] – 1
2. for i ← 1 to n do
3. f[i,i] ← 0
4. for L ← 2 to n do
5. for i ← 1 to n – L + 1 do
6. j ← i + L –1
7. f[i,j] ← ∞
8. for k ← i to j – 1 do
9. q ← $p_{i-1}.p_k.p_j$ + f[i,k] + f[k+1,j]
10. if q < f[i,j]
11. then f[i,j] ← q
12. s[i,j] ←k
13. return m dan s

Penjelasan

- Langkah 1

  Langkah pertama pseudocode order rangkaian matriks adalah menentukan nilai n sebesar panjang[p] –1 dengan panjang[p] adalah banyak anggota himpunan P.

- Langkah 2

  Langkah 2 ini merupakan perintah mengerjakan langkah3 untuk i = 1 sampai dengan n

- Langkah 3

  Langkah 3 ini menentukan nilai f[i,i] = 0 yang merupakan nilai f untuk perkalian rangkaian matriks dengan anggota rangkaian sebanyak 1 sebagaimana telah dijelaskan pada sub bab 3.3.2

- Langkah 4

  Langkah ini merupakan perintah mengerjakan langkah 5 untuk L = 2 sampai n

- Langkah 5

  Langkah 5 ini merupakan perintah mengerjakan langkah 6,7, dan 8 untuk i =1 sampai dengan n – L + 1.

- Langkah 6

  Pada langkah 6 ini ditentukan nilai j sebesar i + L – 1

- Langkah 7

  Pada langkah 7 ini ditentukan nilai f[i,j] sebesar ∞. Nilai f[i,j] = ∞ ini ditentukan terlebih dahulu untuk membantu mencari nilai f[i,j] yang terkecil yaitu pada saat dilakukan pembandingan yang dikerjakan pada langkah 10.

- Langkah 8

  Langkah 8 ini merupakan perintah mengerjakan langkah 9 dan 10 untuk k = i sampai dengan j –1.

- Langkah 9

  Pada langkah 9 ini akan dilakukan proses perhitungan f[i,j] dengan menggunakan persamaan rekursif 3.3.2 dan hasil yang diperoleh disimpan dalam variabel q.

- Langkah 10

  Pada langkah 10 ini dilakukan pembandingan antara nilai f[i,j] dengan nilai q. langkah ini dilakukan dengan mengadakan test apakah : q < f[i,j] ?

  Jika ya maka proses dilanjutkan dengan mengerjakan langkah 11 dan 12

  Jika tidak maka proses akan melompat ke langkah 13.

- Langkah 11

  Langkah 11 ini mengganti nilai f[i,j] yang ada dengan nilai q

- Langkah 12

  Langkah 12 ini menentukan nilai s[i,j] sebesar = k yang diperoleh pada saat kondisi q < f[i,j] terpenuhi.

- Langkah 13

  Pada langkah 13 ini nilai f[i,j] dan s[i,j] akan dikembalikan lagi ke sistem.

Dari langkah 1 sampai dengan langkah 13 terlihat bahwa proses perhitungan rekursif terjadi pada looping yang dikerjakan pada langkah 4 sampai dengan 12. Sedangkan pemanggilan secara rekursif sendiri terjadi pada langkah 9, yaitu pada saat dihitung nilai q yang merupakan nilai sementara dari f[i,j], maka dipanggil lagi nilai f[i,k] dan f[k+1,j].

Dari contoh kasus pada perhitungan manual diketahui bahwa banyaknya matriks dalam permasalahan adalah sebanyak n = 4 dan himpunan P = {30, 1, 40, 10, 25}. Tabel 3.5 menggambarkan hasil kerja pseudocode order rangkaian matriks dengan menggunakan contoh kasus tersebut. Karena f[i,j] didefinisikan hanya pada $i \leq j$, maka hanya bagian atas diagonal utama pada tabel yang digunakan.

Tabel 3.5. Tabel f dan s untuk n = 4

a) tabel f

$$\begin{bmatrix} 0 & 1200 & 700 & 1400 \\ - & 0 & 400 & 650 \\ - & - & 0 & 10000 \\ - & - & - & 0 \end{bmatrix}$$

b) tabel s

$$\begin{bmatrix} - & 1 & 1 & 1 \\ - & - & 2 & 3 \\ - & - & - & 3 \end{bmatrix}$$

Dari tabel terlihat bahwa jumlah minimal perkalian skalar untuk mengalikan keempat matriks adalah f[1,4] = 1400. Dengan menggunakan tabel f, maka nilai minimum f[i,j] untuk melakukan proses perkalian rangkaian matriks pada suatu sub rangkaian $A_i.A_{i+1}...A_j$ dapat ditemukan pada persilangan garis dari arah tegak dan mendatar. Setiap baris dalam arah yang sejajar diagonal utama dalam tabel berisi masukan-masukan nilai f[i,j] untuk rangkaian matriks yang jumlah anggotanya sama atau tahapnya sama. Sebuah masukan f[i,j] dihitung dengan menggunakan hasil dari $p_{i-1}.p_k.p_j$ dengan k = i, i+1,...,j-i dan semua masukan dari f[i,k] dan dari f[k+1,j] sebagaimana telah dijelaskan pada proses perhitungan secara manual.

3.3.4. Membangun Suatu Solusi yang Optimal

Perhitungan secara manual dan perhitungan dengan menggunakan pseudocode order rangkaian matriks telah menentukan nilai minimal dari

perkalian skalar yang dibutuhkan untuk menghitung suatu bentuk perkalian rangkaian matriks dan menentukan nilai s[i,j]. Tetapi hasil yang diperoleh tersebut tidak langsung menunjukkan bagaimana urutan untuk mengalikan rangkaian matriks tersebut atau dengan kata lain bagaimana letak dari tanda kurung yang optimal dalam pemberian tanda kurung secara penuh pada bentuk perkalian rangkaian matriks belum terlihat. Oleh karena itu pada langkah keempatnya, algoritma pemrograman dinamik akan membentuk suatu solusi yang optimal dengan menggunakan informasi atau data-data yang dihasilkan perhitungan pada langkah ketiga.

Dari proses perhitungan scara manual telah diketahui bahwa untuk y_1 =1,4 atau untuk perkalian rangkaian matriks $(A_1.A_2.A_3.A_4)$ didapatkan nilai $s_1[1,4] = 1$. Artinya pada perkalian rangkaian matriks $(A_1.A_2.A_3.A_4)$ akan diletakkan tanda kurung pada posisi setelah matriks A_1. Sehingga diperoleh bentuk perkalian rangkaian matriks $(A_1.(A_2.A_3.A_4))$. Dari bentuk yang terakhir berarti pada tahap 2 akan dicari nilai $s_2[i_2, j_2]$ untuk $(A_2.A_3.A_4)$ dan didapatkan $s_2[2, 4]= 3$. Sehinggga akan diperoleh bentuk perkalian rangkaian matriks $((A_2.A_3).A_4)$ dan dari bentuk ini dicari nilai $s_3[2,3]$ dan diperoleh $s_3[2,3] = 2$ yang berarti membentuk $((A_2).(A_3))$. Bentuk yang terakhir biasa ditulis dengan $(A_2.A_3)$ saja. Sehingga dari keseluruhan jawaban diperoleh nilai $(s_1[i_1,j_1], s_2[i_2,j_2], s_3[i_3,j_3]) = (1, 3, 2)$ yang menjadikan bentuk perkalian rangkaian matriks $(A_1.A_2.A_3.A_4)$ dihitung menurut bentuk $(A_1.((A_2.A_3).A_4))$.

Sebagaimana dalam proses perhitungan secara manual, perhitungan dengan menggunakan bantuan komputer dalam menyelesaikan masalah perkalian

rangkaian matriks ini digunakan tabel s[1..n,1..n] untuk menentukan urutan terbaik dalam mengalikan matriks-matriks tersebut. Setiap dimasukkan s[i,j] berarti diberikan suatu nilai k sedemikian hingga pemberian tanda kurung pada $A_i.A_{i+1}...A_j$ memecah bentuk perkalian rangkaian matriks tersebut di antara A_k dan A_{k+1}. Sehinggga dengan demikian dapat diketahui bahwa perkalian matriks yang terakhir dalam perhitungan $A_{i..n}$ secara optimal adalah perkalian yang dihasilkan dari dua sub rangkaian $A_{1..s[1,n]}$ dan $A_{s[1,n]+1..n}$. Perkalian rangkaian matriks pada sub rangkaian $A_{1..s[1,n]}$ dapat dihitung secara rekursif. Hal ini disebabkan karena adanya s[1,s[1,n]] yang menentukan sub rangkaian terbesar atau menentukan perkalian matriks yang terakhir dalam perhitungan $A_{1..s[1,n]}$ tersebut.

Hal yang sama juga terjadi pada sub rangkaian $A_{s[1,n]+1..n}$, karena s[[1,n]+1,n] juga akan menentukan perkalian matriks yang terakhir dalam perhitungan $A_{s[1,n]+1..n}$. Berikut ini disajikan pseudocode perkalian rangkaian matriks untuk membentuk solusi yang optimal tersebut. Sebagai masukannya adalah matriks-matriks $A=\{A_1, A_2,... A_n\}$, tabel s yang dihasilkan dari pseudocode order rangkaian matriks, nomer urutan i dan j.

Pseudocode perkalian rangkaian matriks (A,s,i,j)

```
1. if j > i
2.     then x ← perkalian rangkaian matriks (A,s,i,s[i,j])
3.          y ← perkalian rangkaian matriks (A,s,s[i,j]+1,j)
4.          return perkalian_matrik(x,y)
5.     else return Ai
```

Dalam contoh pada tabel 3.4 pemanggilan perkalian rangkaian matriks (A,s,1,4) menghitung bentuk perkalian rangkaian matriks $A_1.A_2.A_3.A_4$ tersebut menurut peletakan tanda kurung

$(A_1.((A_2.A_3).A_4))$

3.4. Analisa Running-Time pada Perhitungan Harga Perkalian Skalar

Dalam subbab 3.4. ini akan dilakukan analisa running-time yaitu pencarian big-Oh pada prosedur order_rangkaian_matriks. Analisa dilakukan dengan mengasumsikan bahwa terjadi kondisi terburuk yaitu pada setiap dilakukan pencarian nilai k yang paling tepat, nilai k ditemukan pada iterasi yang paling akhir. Hal ini dilakukan untuk menjamin bahwa semua kemungkinan yang ada sudah tercakup dan running time yang diperoleh akan merupakan running time terbesar sebagaimana yang telah dijelaskan pada subbab 2.4. Dari pseudocode order_rangkaian_matriks dapat diketahui running time untuk setiap statemen dapat dituliskan sebagai berikut :

Prosedur order_rangkaian _matriks

		Times
1.	n ← panjang [p] – 1	1
2.	for i ← 1 to n do	$(\sum_{i=1}^{n} 1) + 1$
3.	f[i,I] ← 0	$\sum_{i=1}^{n} 1$
4.	for L← 2 to n do	$(\sum_{L=2}^{n} 1) + 1$

5.	for i ← 1 to n – L +1 do	$(\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}1)+1$
6.	j ← i + L –1	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}1$
7.	f[i,j] ← ∞	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}1$
8.	for k ← i to j – 1 do	$(\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}\sum_{k=i}^{j-1}1)+1$
9.	q ← $p_{i-1}.p_k.p_j$ + f[i,k] + f[k+1,j]	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}\sum_{k=i}^{j-1}1$
10.	if q < f[i,j]	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}\sum_{k=i}^{j-1}1$
11.	then f[i,j] ← q	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}\sum_{k=i}^{j-1}1$
12.	s[i,j] ←k	$\sum_{L=2}^{n}\sum_{i=1}^{n-L+1}\sum_{k=i}^{j-1}1$
13.	return f dan s	1

Nilai-nilai running time untuk setiap statemen pada pseudocode order_rangkaian_matriks diperoleh dengan penjelasan sebagai berikut.

1. Baris pertama merupakan statemen tunggal yang berisi penugasan pemberian nilai pada variabel n maka harga running timenya adalah 1.
2. Baris ke- 2 dan 3 merupakan loop, maka harga running timenya adalah jumlahan running time tiap satu putaran dimana dalam satu putaran setiap statemen mempunyai runing time sebesar 1. Sehingga untuk seluruh putaran harga running timenya adalah $\sum_{i=1}^{n}1$. Khusus untuk baris kedua yang merupakan counter nilai running timenya ditambah satu karena pada saat putaran terakhir

selesai dikerjakan, sistem kembali ke baris kedua dan selanjutnya menuju baris ke empat.

3. Baris empat sampai baris ke- 12 merupakan loop berkalang, sehingga proses perhitungan running timenya adalah :

   - baris ke- 4 merupakan loop terluar dan harga running timenya adalah $(\sum_{L=2}^{n} 1) + 1$
   - baris ke- 5 merupakan loop di dalam loop dan running timenya dipengaruhi oleh loop terluar sehingga nilai running timenya adalah $(\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} 1)+1$
   - baris ke- 6 dan 7 masing-masing merupakan statemen tunggal dalam loop baris ke-5 yang berarti perhitungan running timenya juga dipengaruhi oleh loop baris ke- 4 sehingga harga running timenya adalah $\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} 1$
   - baris ke- 8 merupakan counter pada loop paling dalam yang perhitungan running timenya dipengaruhi oleh dua loop sebelumnya sehingga harga running timenya adalah $(\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} \sum_{k=i}^{j-1} 1)+1$
   - baris ke- 9 sampai baris ke– 12 merupakan statemen if then dengan asumsi sebelumnya adalah terjadi worst case sehingga kedua statemen pada baris 11 dan 12 akan dieksekusi setiap putarannya dan statemen ini terdapat pada loop paling dalam sehingga harga running timenya adalah $\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} \sum_{k=i}^{j-1} 1$

4. Baris ke- 13 merupakan penugasan sehingga running timenya adalah 1.

Dari penjelasan prosedur order_rangkaian_matriks tersebut, maka running time untuk setiap baris statemen dapat ditulis sebagai berikut :

Times adalah jumlah maksimal eksekusi yang dapat dilakukan setiap baris statemen.

Misalkan c_i dengan i = 1,2,...,13 merupakan konsatanta yang menyatakan banyaknya waktu yang dibutuhkan setiap baris i untuk melakukan satu eksekusi, Untuk menghitung running-time dari prosedur order_rangkaian_matriks, maka running-time setiap baris statemen dijumlahkan sebagai berikut :

$$T(n) = c_1 + c_2.((\sum_{i=1}^{n} 1)+1) + c_3.\sum_{i=1}^{n} 1 + c_4.((\sum_{L=2}^{n} 1)+1) + c_5.((\sum_{L=2}^{n}\sum_{i=1}^{n-l+1} 1)+1) +$$

$$c_6.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1} 1 + c_7.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1} 1 + c_8.((\sum_{L=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{j-1} 1)+1) + c_9.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{j-1} 1 +$$

$$c_{10}.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{j-1} 1 + c_{11}.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{j-1} 1 + c_{12}.\sum_{L=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{j-1} 1 + c_{13}$$

dengan memperhatikan bahwa

$$\sum_{i=a}^{b} c = (b-a+1).c\ , \qquad \sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \qquad \sum_{i=1}^{n} i^2 = \frac{2n^3+3n^2+n}{6}$$

maka,

(i) $\sum_{i=1}^{n} 1 = n,$ $\qquad\qquad$ $\sum_{L=2}^{n} 1 = n-1$

(ii) $\sum_{l=2}^{n}\sum_{i=1}^{n-L+1} 1 = \frac{1}{2}n^2 - \frac{1}{2}n$ dengan penjelasan sebagai berikut.

$$\sum_{i=1}^{n-L+1} 1 = n - L + 1$$

$$\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} 1 = \sum_{L=2}^{n} (n - L + 1)$$

$$= n\sum_{L=2}^{n} 1 - \sum_{L=2}^{n} L + \sum_{L=2}^{n} 1$$

$$= n.(n-1) - \left(\frac{n(n+1)}{2} - 1 \right) + (n-1)$$

$$= \frac{1}{2}n^2 - \frac{1}{2}n$$

(iii) $\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} \sum_{k=i}^{j-1} 1 = \frac{1}{6} n^3 - \frac{1}{6} n$ dengan penjelasan sebagai berikut

$$\sum_{k=i}^{j-1} 1 = j - i$$

pada baris 6 diketahui bahwa j = i + L - 1, sehingga

$$\sum_{k=i}^{j-1} 1 = i + L - 1 - i$$

$$= L - 1$$

$$\sum_{i=1}^{n-L+1} \sum_{k=i}^{j-1} 1 = \sum_{i=1}^{n-L+1} (L - 1)$$

$$= (n - L + 1).(L - 1)$$

$$= -L^2 + L.(n + 2) - (n + 1)$$

$$\sum_{L=2}^{n} \sum_{i=1}^{n-L+1} \sum_{k=i}^{j-1} 1 = \sum_{L=2}^{n} (-L^2 + L.(n+2) - (n+1))$$

$$= -\sum_{L=2}^{n} L^2 + (n+2)\sum_{L=2}^{n} L - (n+1)\sum_{L=2}^{n} 1$$

$$= -\left(\frac{2n^3+3n^2+n}{6}-1\right)+(n+2)\left(\frac{n(n+1)}{2}\right)-(n+1)(n-1)$$

$$= \frac{1}{6}n^3 - \frac{1}{6}n$$

sehingga

$$T(n) = c_1 + c_2.(n+1) + c_3.n + c_4.n + c_5.\left(\frac{1}{2}n^2 - \frac{1}{2}n + 1\right) + (c_6 + c_7).\left(\frac{1}{2}n^2 - \frac{1}{2}n\right) +$$

$$+ c_8.\left(\frac{1}{6}n^3 - \frac{1}{6}n + 1\right) + (c_9 + c_{10} + c_{11} + c_{12}).\left(\frac{1}{6}n^3 - \frac{1}{6}n\right) + c_{13}$$

$$= \frac{1}{6}(c_8 + c_9 + c_{10} + c_{11} + c_{12}).n^3 + \frac{1}{2}(c_5 + c_6 + c_7).n^2$$

$$+ \left(c_2 + c_3 + c_4 - \frac{1}{2}(c_5 + c_6 + c_7) - \frac{1}{6}(c_8 + c_9 + c_{10} + c_{11} + c_{12})\right).n$$

$$+ (c_1 + c_2 + c_5 + c_8 + c_{13})$$

untuk konstanta a,b,c,d dengan $a = \frac{1}{6}(c_8 + c_9 + c_{10} + c_{11} + c_{12})$, $b = \frac{1}{2}(c_5 + c_6 + c_7)$,

$c = \left(c_2 + c_3 + c_4 - \frac{1}{2}(c_5 + c_6 + c_7) - \frac{1}{6}(c_8 + c_9 + c_{10} + c_{11} + c_{12})\right)$, dan

$d = (c_1 + c_2 + c_5 + c_8 + c_{13})$ maka berlaku

$T(n) = a.n^3 + b.n^2 + c.n + d$

$\approx O(n^3)$

BAB IV

KESIMPULAN

Dari pembahasan sebelumnya dapat disimpulkan bahwa algoritma pemrograman dinamik dalam menyelesaikan masalah perkalian barisan matriks berusaha untuk menemukan solusi terbaik pada setiap tahapan yang menjamin keoptimalan hasil penyelesaian secara keseluruhan. Dalam proses pengerjaannya algoritma pemrograman dinamik selalu menyimpan hasil penyelesaian untuk tiap sub masalah, sehingga jika pada suatu tahapan proses ditemukan sub masalah yang sama dengan satu tahap sebelumnya maka algoritma pemrograman dinamik tidak lagi mencari penyelesaian sub masalah yang sama pada tahap tersebut melainkan akan menggunakan hasil penyelesaian yang sudah didapatkan pada satu tahap sebelumnya.

Dalam menyelesaikan masalah perkalian barisan matriks algoritma pemrograman dinamik mempunyai running time $O(n^3)$.

DAFTAR PUSTAKA

Anton, Howard : *Aljabar Linier Elementer, edisi kelima*, alih bahasa : Pantur Silaban; I. Nyoman S., Penerbit Erlangga, Jakarta,1995

Baase, Sara : *Computer Algorithms, Introduction to Design an Analysis, second edition*, Addison-Wesley Publishing Company, New York, 1989

Cormen, Thomas H. ; Leiserson, Charles E. ; Rivest, Ronald L. : *Introduction to Algorithms*, Mc Graw-Hill Book Company, New York, 1990

Dimyati, Tjutju T. ; Dimyati, Ahmad : *Operation Research, Model-Model Pengambilan Keputusan*, Penerbit Sinar Baru, Bandung, 1989

Goldberg, Jack L. : *Matrix Theory with Applications*, Mc Graw-Hill Inc., New York, 1991

Hillier, Frederick S.; Lieberman, Gerald J.: *Pengantar Riset Operasi*, alih bahasa : Gunawan, Ellen ; Mulia, Ardi W., Penerbit Erlangga, Jakarta, 1994

Sernesi, E. : *Linear Algebra, Geometric Approach*, Chapman and Hall, London, 1993

Suryadi, M. T. : *Pengantar Analisis Algoritma*, Penerbit Gunadarma, Jakarta, 1990

Toha, Hamdy A. : *Operation Research, An introduction, third edition*, Mac Millan Publishing CO, Inc, California, 1982

Weiss, Mark A. : *Data Structures and Algorithms in C++*, Benjamin/Cumming Publishing Company Inc., Canada, 1994

LAMPIRAN KODE PROGRAM

```
{$M 65520,0,650000}
{*************************************************************************
    PROGRAM UNTUK MENYELESAIKAN MASALAH PERKALIAN RANGKAIAN MATRIKS
    DISUSUN OLEH : HELMIE ARIF WIBAWA (J 2A0 96 027)
 *************************************************************************}
program MATRIKS;
uses crt,dos;
type Pmat = ^mat;
     mat = record
        baris : byte;
        kolom : byte;
        nilai :integer;
        kanan,
        bawah : Pmat
     end;
     matrixs = record
        baris, kolom : byte;
        nilai        : array[1..50,1..50] of integer;
     end;
     DataMatrik = record
        byk      : byte;
        p1,p2    : bitel;
        matrix   : array [1..10] of matrixs;
     end;
     RecordMenu = record
        PosisiMenu          : array[1..6] of byte;
        KarakterMenu        : array[1..6] of char;
        PosisiKarakterMenu  : array[1..6] of byte;
        IsiMenu             : array[1..6] of string23;
      end;
      RecordMenuEdit = record
        PosisiMenu          : array[1..4] of byte;
        KarakterMenu        : array[1..4] of char;
        PosisiKarakterMenu  : array[1..4] of byte;
        IsiMenu             : array[1..4] of string23;
      end;
      RecordMenuProses = record
        PosisiMenu          : array[1..3] of byte;
        KarakterMenu        : array[1..3] of char;
        PosisiKarakterMenu  : array[1..3] of byte;
        IsiMenu             : array[1..3] of string23;
      end;
      pixel = record
        karakter : char;
        atribut  : byte;
      end;
     matrik  = array [1..10] of Pmat;
     tabel   = array [1..2] of Pmat;
     bite    = array[1..50,1..50]of byte;
     bitel   = array[0..50] of byte;
     string3 = string[3];
     string23 = string[23];
     mskalar = array[1..50,1..50] of longint;
     bufferlayar = array[1..25,1..80] of pixel;
```

```
const DataMenu : RecordMenu =
      (PosisiMenu : (9,10,11,12,13,14);
       KarakterMenu : 'IMBSPe';
       PosisiKarakterMenu : (28,28,28,28,28,29);
       IsiMenu :(' Informasi Menu          ',' Masukan Data            ',
                ' Buka File                ',' Simpan ke File          ',
                ' Proses                   ',' Keluar                  '));
      DataMenuProses : RecordMenuProses =
      (PosisiMenu : (9,10,11);
       KarakterMenu : 'TDe';
       PosisiKarakterMenu : (28,28,29);
       IsiMenu :(' Tanpa Algoritma DP  ',' Dengan Algoritma DP ',
                ' Ke Menu Utama        '));
var c,mat_kali : Pmat;
    h          : matrik;
    jawab      : char;
    bar,kol,i,ii,jj,n : byte;
    nla      : integer;
    r,r1,r2 : bite1;
    tt       : bite;
    byk      : byte;
    benar : boolean;
    Layar : BufferLayar absolute $B800:$0000;
    LayarMenuUtama,LayarMasukan,LayarMenuProses : BufferLayar;
    h1,h2,m1,m2,s1,s2,hund1,hund2 :word;


{prosedur untuk membuat simpul baru dari matriks }
procedure buat_simpul(var Baru : Pmat; b,k : byte; n : integer);
begin
   new(Baru);
   with Baru^ do
     begin
       baris := b;
       kolom := k;
       nilai := n;
       kanan := nil;
       bawah := nil;
     end
end;

{prosedur untuk mencari informasi apakah suatu baris terdapat dalam matriks}
procedure ada_baris(var nb : Pmat ; var ada : boolean; awal : Pmat; b : byte);
begin
  ada := false;
  nb := awal^.bawah;
  repeat
    if nb^.baris = b then
       ada := true
    else
       nb := nb^.bawah
  until ada or (nb = awal) or (nb^.baris > b)
end;
```

```
{prosedur untuk mencari informasi apakah suatu kolom terdapat dalam matriks}
procedure ada_kolom(var nk : Pmat; var ada : boolean; awal_ : Pmat; k : byte);
begin
  ada := false;
  nk := awal^.kanan;
  repeat
    if nk^.kolom = k then
      ada := true
    else
      nk := nk^.kanan
  until ada or (nk = awal) or (nk^.kolom > k)
end;

{prosedur untuk memberikan informasi tentang langkah yang harus dilakukan
pengguna pada saat mengoperasikan sistem}
procedure Petunjuk(P:string);
var x,y,n:integer;
begin
  textcolor(15);
  textbackground(8);
  gotoxy(1,24);write(P);
  x:=wherex;
  y:=wherey;
  n:=81-x;
  gotoxy(x,y);write(' ':n);
  textcolor(0);
  textbackground(15);
end;

{prosedur untuk menampilkan informasi pada pengguna tentang kesalahan yang
terjadi dan saran pembetulannya}
procedure Peringatan(P:string);
var x,y,n:integer;
begin
  textcolor(15);
  textbackground(4);write(#7);
  gotoxy(1,24);write(P);
  x:=wherex;
  y:=wherey;
  n:=81-x;
  gotoxy(x,y);write(' ':n);
  textbackground(15);
  textcolor(0);
end;

{prosedur untuk menempatkan suatu elemen matriks pada suatu matriks sesuai
dengan baris dan kolomnya}
procedure sisip_elemen(var awal : Pmat;b,k : byte;n : integer);
var Baru,sb,sk,n_baris,n_kolom,bb : Pmat;
    ada : boolean;
begin
  buat_simpul(Baru,b,k,n);
  ada_baris(n_baris,ada,awal,b);
  if not (ada) then
    begin
      buat_simpul(sb,b,0,0);
```

```
        sb^.kanan := sb;
        sb^.bawah := sb;
        bb := awal;
        while (bb^.bawah^.baris < b) and(bb^.bawah <> awal) do
           bb := bb^.bawah;
        if bb^.bawah = awal then
           sb^.bawah := awal
        else
           sb^.bawah := bb^.bawah;
        bb^.bawah := sb;
        n_baris := sb;
      end;
    ada_kolom(n_kolom,ada,awal,k);
    if not (ada) then
      begin
        buat_simpul(sk,0,k,0);
        sk^.kanan := sk;
        sk^.bawah := sk;
        bb := awal;
        while (bb^.kanan^.kolom < k) and(bb^.kanan <> awal) do
           bb := bb^.kanan;
        if bb^.kanan = awal then
           sk^.kanan := awal
        else
           sk^.kanan := bb^.kanan;
        bb^.kanan := sk;
        n_kolom := sk;
      end;
    sb := n_baris;
    while (sb^.kanan^.kolom < k) and(sb^.kanan^.kolom <> 0) do
       sb := sb^.kanan;
    if sb^.kanan^.kolom = 0 then
       Baru^.kanan := n_baris
    else
       Baru^.kanan := sb^.kanan;
    sb^.kanan := Baru;
    sk := n_kolom;
    while (sk^.bawah^.baris < b) and(sk^.bawah^.baris <> 0) do
       sk := sk^.bawah;
    if sk^.bawah^.baris = 0 then
       Baru^.bawah := n_kolom
    else
       Baru^.bawah := sk^.bawah;
    sk^.bawah := Baru
end;

{prosedur untuk mencari elemen matriks tertentu pada suatu matriks dalam
rangkaian dan menempatkan alamat simpul pada posisi tersebut}
function CariMatrik(M : Pmat; i,j : byte):Pmat;
var Caribrs,cariklm : Pmat;
begin
 if M <> nil then
 begin
  caribrs := M^.bawah;
  if caribrs <> nil then
   begin
```

```
    while (caribrs <> nil) and (caribrs^.baris <> i) do
     caribrs := caribrs^.bawah;
    cariklm := caribrs^.kanan;
    while (cariklm <> nil) and (cariklm^.kolom <> j) do
     cariklm := cariklm^.kanan;
    CariMatrik := cariklm
   end
  else
   CariMatrik := caribrs;
 end
 else
 CariMatrik := M;
end;

{prosedur untuk mengalikan dua matriks x dan matriks y dan menghasilkan matriks
hasil yaitu kali yang akan dikembalikan ke sistem}
procedure kali_matriks(x,y : Pmat; var kali : Pmat);
var N,S,M : tabel;
    I,J,k,brs,klm : byte ;
    jml : integer;
begin
  buat_simpul(kali,0,0,0);
  kali^.kanan := kali;
  kali^.bawah := kali;
  N[1] := x^.bawah;
  N[2]:= y^.kanan;
  repeat
    repeat;
      jml := 0;
      S[1] := N[1]^.kanan;
      S[2] := N[2]^.bawah;
      repeat
        if S[1]^.kolom <> S[2]^.baris then
          begin
             jml := jml;
          if S[1]^.kolom > S[2]^.baris then
             S[2] := S[2]^.bawah
          else
             S[1] := S[1]^.kanan;
          end
        else
          begin
             jml := jml + S[1]^.nilai * S[2]^.nilai;
             S[2] := S[2]^.bawah;
             S[1] := S[1]^.kanan;
          end;
      until (S[1] = N[1]) or (S[2] = N[2]);
      brs := S[1]^.baris;
      klm := S[2]^.kolom;
      if (brs <> 0) and (klm <> 0) then
      sisip_elemen(kali,brs,klm,jml);
      N[2] := N[2]^.kanan;
      until N[2] = y;
      N[1] := N[1]^.bawah;
    until N[1] = x;
end;
```

```
{prosedur untuk mencetak elemen matriks dalam bentuk matriks (segi empat)}
procedure cetak_matriks(awal : Pmat);
var bb,bb1 : Pmat;
    jk,jb,i,j,n_bar,n_kol,uk,baris : integer;
begin
  baris := 0;
  bb := awal^.kanan;
  while bb^.kanan^.kolom <> 0 do
     bb := bb^.kanan;
  jk := bb^.kolom;
  bb := awal^.bawah;
  while bb^.bawah^.baris <> 0 do
     bb := bb^.bawah;
  jb := bb^.baris;
  uk := jb;
  if (jb <> jk) then
    if jb < jk then
       uk := jk
    else
       uk := jb;
  bb := awal^.bawah;
  n_bar := 0;
  repeat
    if (bb^.baris - N_bar) <> 1 then
      for i := n_bar + 1 to bb^.baris - 1 do
         begin
           for j := 1 to uk do
             write(0:8);
           writeln
         end;
    bb1 := bb^.kanan;
    if (bb1^.kolom - bb^.kolom) <> 1 then
      for i := 1 to bb1^.kolom - 1 do
         write(0:8);
    repeat
      baris := baris + 1;
      write('Elemen baris ',bb1^.baris,' kolom ',bb1^.kolom,' = ');
      writeln(bb1^.nilai:8);
      if baris = 20 then
      begin
        baris := 0;readln;
        clrscr;
      end;
      n_kol := bb1^.kolom;
      bb1 := bb1^.kanan;
      if bb1^.kolom <> 0 then
         if (bb1^.kolom - n_kol) <> 1 then
           for i := 1 to bb1^.kolom-n_kol-1 do
              write(0:8);
    until bb1^.kolom = 0;
    writeln;
    n_bar := bb^.baris;
    bb := bb^.bawah
  until bb^.baris = 0;
end;
```

```
{prosedur untuk mencetak elemen matriks tidak dalam bentuk matriks}
procedure cetak_matriks2(awal : Pmat);
var bb,bb1 : Pmat;
    jk,jb,i,j,n_bar,n_kol,uk,baris : integer;
begin
  baris := 0;
  bb := awal^.kanan;
  while bb^.kanan^.kolom <> 0 do
     bb := bb^.kanan;
  jk := bb^.kolom;
  bb := awal^.bawah;
  while bb^.bawah^.baris <> 0 do
     bb := bb^.bawah;
  jb := bb^.baris;
  uk := jb;
  if (jb <> jk) then
    if jb < jk then
       uk := jk
    else
       uk := jb;
  bb := awal^.bawah;
  n_bar := 0;
  repeat
    if (bb^.baris - N_bar) <> 1 then
      for i := n_bar + 1 to bb^.baris - 1 do
         begin
           for j := 1 to uk do
             write(0:8);
           writeln
         end;
    bb1 := bb^.kanan;
    if (bb1^.kolom - bb^.kolom) <> 1 then
      for i := 1 to bb1^.kolom - 1 do
         write(0:8);
    repeat
      write(bb1^.nilai:8);
      n_kol := bb1^.kolom;
      bb1 := bb1^.kanan;
      if bb1^.kolom <> 0 then
         if (bb1^.kolom - n_kol) <> 1 then
           for i := 1 to bb1^.kolom-n_kol-1 do
              write(0:8);
    until bb1^.kolom = 0;
    writeln;
    n_bar := bb^.baris;
    bb := bb^.bawah
  until bb^.baris = 0;
end;

{prosedur untuk delay sampai waktu tunggu selesai atau sampai suatu tombol
ditekan}
procedure TungguTekan(Tunda : integer);
var i,j,x : integer;
    T,tekan : char;
begin
  i := 0;
```

```
    x := Tunda*100;
    repeat
      i := i+1;
      delay(10);
    until (keypressed) or (i > x);
    if keypressed then
    begin
      tekan := readkey;
      write(#7);
    end;
  end;

  {prosedur untuk menampilkan isi informasi tentang fungsi dari bagian-bagian yang
  terdapat pada menu utama}
  procedure InformasiMenuUtama;
  var i : integer;
  begin
    textbackground(15);clrscr;
    textcolor(4);gotoxy(1,2);
    write('INFORMASI MENU');
    textcolor(1);
    gotoxy(15,2);write('============================================================');
    gotoxy(1,4);write(DataMenu.IsiMenu[2]);
    gotoxy(1,7);write(DataMenu.IsiMenu[3]);
    gotoxy(1,12);write(DataMenu.IsiMenu[4]);
    gotoxy(1,16);write(DataMenu.IsiMenu[5]);
    gotoxy(1,20);write(DataMenu.IsiMenu[6]);
    textcolor(0);
    gotoxy(3,5);write('Memberi masukan matriks dari papan ketik');
    gotoxy(3,8);write('Memberi masukan matriks dengan membuka file yang menyimpan
                       data matriks');
    gotoxy(3,9);write('nama file dan posisi awal serta akhir rangkaian diketikkan
                       pada tempat ');
    gotoxy(3,10);write('yang disediakan');
    gotoxy(3,13);write('Menyimpan rangkaian matriks yang ada di memori ke media
                        penyimpanan');
    gotoxy(3,14);write('disket atau hard disk');
    gotoxy(3,17);write('Melakukan proses dimana dapat dipilih menggunakan
                        Algoritma Pemrograman');
    gotoxy(3,18);write('Dinamik atau tidak dan menampilkan hasilnya beserta lama
                        eksekusinya');
    gotoxy(3,21);write('Kembali ke DOS');
    gotoxy(3,22);write('data rangkaian matriks otomatis hilang jika tidak disimpan
                        ke file ');
    gotoxy(1,24);textcolor(1);
    write('============================================================');
    gotoxy(10,25);textcolor(4);write('Tekan <enter>');readln;
  end;

  {prosedur untuk menampilkan kover program pada layar}
  procedure Cover;
  begin
    textbackground(15);
    textcolor(0);clrscr;
    gotoxy(1,4);
    writeln('':8,'╔═══════════════════════════════════════════════════════════╗');
```

```
    writeln('':8,'╔═══════════════════════════════════════════════════════╗');
    writeln('':8,'║ Aplikasi Penyelesaian Masalah Perkalian Rangkaian Matriks ');
    writeln('':8,'╠═══════════════════════════════════════════════════════╣');
    writeln('':8,'║                    Disusun oleh :                     ');
    writeln('':8,'║                  Helmie ARif Wibawa                   ');
    writeln('':8,'║                   ( J2A 096 060 )                     ');
    writeln('':8,'╠═══════════════════════════════════════════════════════╣');
    writeln('':8,'║                  Jurusan Matematika                   ');
    writeln('':8,'║       Fakultas Matematika dan Ilmu Pengetahuan Alam   ');
    writeln('':8,'║                  Universitas Diponegoro               ');
    writeln('':8,'║                      Semarang                         ');
    writeln('':8,'║                       2001                            ');
    writeln('':8,'╚═══════════════════════════════════════════════════════╝');
    TungguTekan(5);
  end;

  {prosedur untuk mendapatkan masukan nama file yang akan digunakan dan drive
  tempat penyimpanan}
  procedure FormBSFile(var NamaFile:string);
  begin
    textbackground(0);
    textcolor(15);
    gotoxy(20,13);write('                                          ');
    gotoxy(20,14);write(' Masukan nama file yang akan di gunakan ');
    gotoxy(20,15);write('  dengan nama drive tempat penyimpanan  ');
    gotoxy(20,16);write('      Nama file  :                       ');
    gotoxy(20,17);write('                                         ');
    gotoxy(38,16);readln(NamaFile);
    gotoxy(38,16);write('                  ');
    textbackground(15);
  end;

  {prosedur untuk menampilkan efek sorotan pada salah satu pilihan yang terdapat
  pada layar menu}
  procedure ChAtributLayar(x1,y1,x2,y2,Atribut : byte);
  var posisi : word;
      i,j    : byte;
  begin
    for i := y1 to y2 do
    begin
      for j := x1 to x2 do
      begin
        posisi := (((i-1)*80+j)-1)*2;
        Mem[$B800:$0000+posisi+1] := Atribut;
      end;
    end;
  end;

  {prosedur untuk mengembalikan layar sesuai dengan layar yang diinginkan pada
  suatu posisi tertentu}
  procedure KembalikanLayar(x1,y1,x2,y2:byte;var LayarSimpan : BufferLayar);
  var posisi : word;
      i,j    : byte;
  begin
    for i := y1 to y2 do
```

```
  begin
    for j := x1 to x2 do
    with LayarSimpan[i,j] do
    begin
      posisi := (((i-1)*80+j)-1)*2;
      Mem[$B800:$0000+posisi] := ord(karakter);
      Mem[$B800:$0000+posisi+1] := Atribut;
    end;
  end;
end;

{prosedur untuk menyimpan data ke dalam file dengan nama dan drive yang
dimasukkan}
procedure SimpanKeFile(var SS:Matrik; p1,p2 :bite1; n:byte);
var Berkas     : text;
    NamaFile   : string;
    S          : Matrik;
    i          : byte;
    SudahAda   : boolean;
    F,G        : Pmat;
begin
  S:=SS;
  FormBSFile(NamaFile);
  Assign(Berkas,NamaFile);
  {$I-}
  reset(Berkas);
  {$I+}
  SudahAda := (IOResult=0);
  if SudahAda then
  begin
   jawab := ' ';
   while Not((jawab='Y') or (jawab='T')) do
   begin
    KembalikanLayar(20,13,85,17,LayarMenuUtama);
    write(chr(7));textcolor(4);
    gotoxy(18,20);write('File sudah ada, Tumpangi !!! (Y / T) ? ');
    readln(jawab);textcolor(0);
    jawab := upcase(jawab);
   end;
   if jawab = 'T' then exit;
  end;
  rewrite(Berkas);
  writeln(berkas,n:3);
  for i := 1 to n do
  begin
   writeln(berkas,p1[i]:3,p2[i]:3);
   G := S[i]^.bawah;
   repeat
    F := G^.kanan;
    repeat
      textbackground(0);textcolor(15);
      gotoxy(20,14);write('                                    ');
      gotoxy(20,15);write(' Tunggu sedang menyimpan ..... ');
      gotoxy(20,16);write('                                    ');
      textbackground(15);textcolor(0);
      writeln(berkas,F^.nilai:5);
```

```
        F := F^.kanan;
      until F = G;
      G := G^.bawah;
      until G = S[i];
    end;
    close(Berkas);
  end;

  {prosedur untuk menampilkan form untuk memberikan masukan melalui papan ketik
  pada sistem}
  procedure FormMasukan;
  var i:byte;
  begin
    textbackground(15);clrscr;
    gotoxy(33,1);write('MASUKAN DATA');
    gotoxy(3,2);write('┌──────────────────────────────────────────────────────────┐');
    gotoxy(3,3);write('│ Banyak matrik dalam rangkaian :                          │');
    gotoxy(3,4);write('└──────────────────────────────────────────────────────────┘');
    gotoxy(3,7);write('┌── UKURAN MATRIKS KE- ────────────────────────────────────┐');
    gotoxy(3,8);write('│ Banyaknya baris :                                        │');
    gotoxy(3,9);write('│ Banyaknya kolom :                                        │');
    gotoxy(3,10);write('└──────────────────────────────────────────────────────────┘');
    gotoxy(3,12);write('┌── ELEMEN MATRIKS KE- ────────────────────────────────────┐');
    gotoxy(3,13);write('│                                                          │');
    gotoxy(3,14);write('│ Baris ke :                                               │');
    gotoxy(3,15);write('│ Kolom ke :                                               │');
    gotoxy(3,16);write('│ Nilai elemen :                                           │');
    gotoxy(3,17);write('└──────────────────────────────────────────────────────────┘');
    gotoxy(3,19);write('┌── AWAL DAN AKHIR ────────────────────────────────────────┐');
    gotoxy(3,20);write('│ Awal rangkaian terletak pada matriks ke  :               │');
    gotoxy(3,21);write('│ Akhir rangkaian terletak pada matriks ke :               │');
    gotoxy(3,22);write('└──────────────────────────────────────────────────────────┘');
    move(Layar,LayarMasukan,4000);
  end;

  {prosedur untuk mendapatkan masukan berupa banyaknya matriks yang terdapat dalam
  rangkaian}
  procedure MasukanBanyakMatriks(var n: byte);
  var IOn : byte;
  begin
    petunjuk('Masukan banyaknya matriks yang ada dalam rangkaian ');
    {$I-}
    repeat
      gotoxy(38,3);readln(n);
      IOn := IOresult;
      if IOn <> 0 then
      begin
        peringatan(' tipe data tidak valid');
        KembalikanLayar(38,3,40,3,LayarMasukan);
      end;
    until IOn = 0;
    {$I+}
  end;
```

```
{prosedur untuk mendapatkan masukan berupa elemen suatu matriks dalam rangkaian
sesuai dengan baris dan kolom yang dimasukkan}
procedure MasukanElemenMatriks(var nilai :integer;i,j: byte);
var IOnilai : byte;
begin
  petunjuk('Masukan nilai elemen untuk baris dan kolom yang sesuai ');
  gotoxy(16,14);write(i);
  gotoxy(16,15);write(j);
  {$I-}
  repeat
    gotoxy(20,16);readln(nilai);
    IOnilai := IOresult;
    if IOnilai <> 0 then
      begin
      KembalikanLayar(20,16,25,16,LayarMasukan);
      peringatan(' Tipe nilai yang dimasukkan salah, seharusnya bertipe
                 integer');
      end;
  until IOnilai = 0;
  {$I+}
end;

{prosedur untuk mendapatkan masukan berupa posisi awal rangkaian yaitu posisi
matriks pertama dalam rangkaian}
procedure MasukanAwal(var i : byte);
var IOawal : byte;
begin
  petunjuk('Masukan posisi matriks yang berada di awal rangkaian ');
  {$I-}
  repeat
   gotoxy(48,20);readln(i);
   IOawal := IOresult;
   if IOawal <> 0 then
   begin
    KembalikanLayar(48,20,50,20,LayarMasukan);
    Peringatan(' Tipe data awal rangkaian salah, seharusnya bertipe integer');
   end;
  until IOawal = 0;
  {$I+}
end;

{prosedur untuk mendapatkan masukan berupa posisi akhir rangkaian yaitu berupa
posisi matriks terakhir dalam rangkaian}
procedure MasukanAkhir(var j : byte);
var IOakhir : byte;
begin
  petunjuk('Masukan posisi matriks yang berada di akhir rangkaian ');
  {$I-}
  repeat
  gotoxy(48,21);readln(j);
  IOakhir := IOresult;
  if IOakhir <> 0 then
  begin
   KembalikanLayar(48,21,50,21,LayarMasukan);
   Peringatan(' Tipe data akhir rangkaian salah, harusnya bertipe integer');
  end;
```

```
  until IOakhir = 0;
  {$I+}
end;

{prosedur untuk mendapatkan masukan berupa banyaknya baris suatu matriks}
procedure MasukanBaris(var i : byte);
var IObaris : byte;
begin
  petunjuk(' Masukkan banyaknya baris pada matriks !');
  {$I-}
  repeat
  gotoxy(23,8);readln(i);
  IObaris := IOresult;
  if IObaris <> 0 then
  begin
   KembalikanLayar(23,8,25,8,LayarMasukan);
   Peringatan(' Tipe data banyak baris salah, harusnya bertipe integer');
  end;
  until IObaris = 0;
  {$I+}
end;

{prosedur untuk mendapatkan masukan berupa banyaknya kolom suatu matriks}
procedure MasukanKolom(var j : byte);
var IOkolom : byte;
begin
  petunjuk(' Masukkan banyaknya kolom pada matriks ! ');
  {$I-}
  repeat
  gotoxy(23,9);readln(j);
  IOkolom := IOresult;
  if IOkolom <> 0 then
  begin
   KembalikanLayar(23,9,25,9,LayarMasukan);
   Peringatan(' Tipe data banyak kolom salah, harusnya bertipe integer');
  end;
  until IOkolom = 0;
  {$I+}
end;

{prosedure untuk memasukkan posisi baris suatu elemen matriks pada saat akan
dilakukan pembetulan elemen matriks}
procedure Gantibaris(var i : byte);
var IObaris : byte;
begin
  {$I-}
  repeat
  textbackground(1);textcolor(15);
  gotoxy(19,20);write('       ');
  gotoxy(19,20);readln(i);
  IObaris := IOresult;
  if IObaris <> 0 then
  begin
   Peringatan(' Tipe data baris salah, harusnya bertipe integer');
  end;
  until IObaris = 0;
```

```
    {$I+}
    textbackground(15);
    textcolor(0);
  end;

  {prosedure untuk memasukkan posisi kolom suatu elemen matriks yang akan diganti
  pada saat dilakukan pembetulan elemen matriks}
  procedure Gantikolom(var j : byte);
  var IOkolom : byte;
  begin
    {$I-}
    repeat
    textbackground(1);
    textcolor(15);
    gotoxy(44,20);write('       ');
    gotoxy(44,20);readln(j);
    IOkolom := IOresult;
    if IOkolom <> 0 then
    begin
     Peringatan(' Tipe data kolom salah, harusnya bertipe integer');
    end;
    until IOkolom = 0;
    {$I+}
    textbackground(15);textcolor(0)
  end;

  {prosedur untuk memasukkan elemen pengganti pada saat akan dilakukan pembetulan
  elemen matriks}
  procedure GantiNilai(var i : integer);
  var IOnilai : byte;
  begin
    {$I-}
    repeat
    textbackground(1);textcolor(15);
    gotoxy(29,21);readln(i);
    IOnilai := IOresult;
    if IOnilai <> 0 then
    begin
     gotoxy(29,21);write('     ');
     Peringatan(' Tipe data untuk elemen salah, harusnya bertipe integer');
    end;
    until IOnilai = 0;
    {$I+}
    textbackground(15);textcolor(0);
  end;

  {Prosedur untuk mencetak nama-nama file pada suatu direktori ke layar pada saat
  dilakukan proses buka file}
  procedure TulisFile;
  var  DirCari : string[139];
       hasil   : SearchRec;
  begin
    clrscr;
    gotoxy(3,3);write('File : ');
    gotoxy(10,3); readln(DirCari);writeln;
    FindFirst(DirCari,Archive,hasil);
```

```
  while DosError = 0 do
  begin
   write(hasil.name:16);
   FindNext(Hasil);
  end;
end;

{prosedure untuk memberikan pesan kesalahan yang terjadi pada proses pemasukan
posisi awal rangkaian}
procedure AwalSalah(i:byte);
begin
  textbackground(4);
  textcolor(7);
  gotoxy(10,15);write('                                                  ');
  gotoxy(10,16);write('       POSISI MATRIKS PERTAMA SALAH               ');
  gotoxy(10,17);write('   SEHARUSNYA BERADA DI ANTARA 1 DAN ',i,'        ');
  gotoxy(10,18);write('            BETULKAN DAHULU !.                    ');
  gotoxy(10,19);write('                                                  ');
  textcolor(0);
  textbackground(7);
  readln;
end;

{prosedur untuk menampilkan pesan kesalahan pada saat proses memasukkan posisi
akhir rangkaian}
procedure AkhirSalah(j:byte);
begin
  textbackground(4);
  textcolor(7);
  gotoxy(10,15);write('                                                  ');
  gotoxy(10,16);write('          POSISI MATRIKS TERAKHIR SALAH           ');
  gotoxy(10,17);write('             TIDAK BOLEH LEBIH DARI ',j,'         ');
  gotoxy(10,18);write('                BETULKAN DAHULU !                 ');
  gotoxy(10,19);write('                                                  ');
  textcolor(0);
  textbackground(15);
  readln;
end;

{prosedur untuk menampilkan pesan kesalahan pada saat terjadi kesalahan pada
memasukkan banyaknya baris ( baris ke i ≠ kolom ke i-1)}
procedure MasukanSalah(k,p :byte);
begin
  textbackground(4);
  textcolor(7);
  gotoxy(10,15);write('                                                  ');
  gotoxy(10,16);write('       KOLOM MATRIKS ',k-1,' DAN BARIS MATRIKS ',k,' ');
  gotoxy(10,17);write('                TIDAK  SESUAI !!!                 ');
  gotoxy(10,18);write(' SEHARUSNYA BANYAKNYA BARIS MATRIKS ',k,' ADALAH',p,'');
  gotoxy(10,19);write('                BETULKAN DAHULU !                 ');
  gotoxy(10,20);write('                                                  ');
  textcolor(0);textbackground(15);
  readln;
end;
```

```
{prosedure induk untuk melakukan proses masukan data rangkaian matriks}
procedure input(var p,p1,p2:bite1; var matrix :matrik; var n,ii,jj : byte);
var i,j,k,bar,kol,brs,klm,bantu : byte;
    nla : integer;
    benar : boolean;
    edit,jawab : char;
    M,MM,MA,Mbawah,Mkolom : Pmat;
    IOn : integer;
begin
  KembalikanLayar(1,1,80,25,LayarMasukan);
  MasukanBanyakMatriks(n);
  bantu := 0;
  for k := 1 to n do
     begin
       buat_simpul(matrix[k],0,0,0);
       matrix[k]^.bawah := matrix[k];
       matrix[k]^.kanan := matrix[k];
       bantu := bantu + 1;
       gotoxy(26,7);write(k);
       gotoxy(26,12);write(k);
       MasukanBaris(p1[k]);
       repeat
         benar := true;
         petunjuk('Masukan banyaknya baris pada matriks ');
         if (k > 1) and (p1[k] <> p2[k-1]) then
           begin
            MasukanSalah(k,p2[k-1]);
            KembalikanLayar(10,15,59,20,LayarMasukan);
            KembalikanLayar(22,8,26,9,LayarMasukan);
            MasukanBaris(p1[k]);
            benar := false;
           end;
       until benar;
       petunjuk(' Masukkan banyaknya kolom pada matriks ');
       MasukanKolom(p2[k]);
       p[k] := p2[k];
       for i := 1 to p1[k] do
        begin
          bar := i;
          for j := 1 to p2[k] do
          begin
            kol := j;
            MasukanElemenMatriks(nla,i,j);
            sisip_elemen(matrix[k],bar,kol,nla);
            if bantu <> n then
              KembalikanLayar(15,14,20,15,LayarMasukan);
            if (bantu <> n) and (k <> n) and (p2[k] <> p2[n]) then
              KembalikanLayar(18,16,22,16,LayarMasukan);
          end;
        end;
        repeat
        edit :=' ';
        while not ((edit = 'Y') or (edit = 'T')) do
        begin
         clrscr;
         gotoxy(5,5);writeln('matriks ke- ',k,' yang anda masukkan adalah
```

```
                sebagai berikut :');writeln;
            cetak_matriks2(matrix[k]);
            textcolor(1);
            gotoxy(15,20);write(' Apakah ada data yang salah (Y/T) ? ');read(edit);
            edit := upcase(edit); textcolor(0);
            KembalikanLayar(15,20,40,20,LayarMasukan);
          end;
          if edit = 'Y' then
          begin
            textbackground(1);
            textcolor(15);
            gotoxy(3,18);write('                                                    ');
            gotoxy(3,19);write(' Masukkan posisi elemen dan nilai elemen yang akan
                               diganti !       ');
            gotoxy(3,20);write(' Posisi baris :                    kolom :      ');
            gotoxy(3,21);write(' Nilai elemen yang baru :                       ');
            gotoxy(3,22);write('                                                ');
            petunjuk('Masukan posisi baris dari elemen yang akan diganti ');
            repeat
             benar := true;
             GantiBaris(bar);
             if (1 > bar) or (bar > p1[k]) then
             begin
              peringatan(' Posisi baris untuk elemen harus disesuaikan ukuran
                           barisnya');
              benar := false;
             end;
            until benar;
             petunjuk('Masukan posisi kolom dari elemen yang akan diganti ');
            repeat
             benar := true;
             GantiKolom(kol);
             if (1 > kol) or (kol > p2[k]) then
             begin
              peringatan(' Posisi kolom untuk elemen yang akan diganti salah,
                           harus sesuai ukuran kolom');
              benar := false;
             end;
            until benar;
            petunjuk(' Masukkan elemen matriks yang baru ! ');
            GantiNilai(nla);
            M := CariMatrik(matrix[k],bar,kol);
            M^.nilai := nla;
            textcolor(0);textbackground(15);
        end;
        until edit = 'T';
         KembalikanLayar(1,1,80,25,LayarMasukan);
         if bantu <> n then
            KembalikanLayar(22,8,26,9,LayarMasukan);
    end;
p[0] := p1[1];
repeat
  benar := true;
  MasukanAwal(ii);
  if (ii < 1) or (ii > n) then
     begin
```

```
          AwalSalah(n);
          KembalikanLayar(10,15,59,19,LayarMasukan);
          gotoxy(46,20);write('          ');
          benar := false;
        end;
    until benar;
    repeat
      benar := true;
      MasukanAkhir(jj);
      if (jj < 1) or (jj > n) then
        begin
          AkhirSalah(n);
          KembalikanLayar(10,15,65,19,LayarMasukan);
          gotoxy(46,21);write('        ');
          benar := false;
        end;
    until benar;
  end;

  {prosedure untuk menampilkan form pengisian posisi awal dan akhir rangkaian }
  Procedure FormAwalAkhir(var i,j : byte;n :byte);
  begin
    textbackground(1);
    textcolor(15);
    gotoxy(3,19);write('┌── MASUKAN POSISI AWAL DAN AKHIR RANGKAIAN ──────────┐');
    gotoxy(3,20);write('| Terdapat ',n,' matriks dalam rangkaian              |');
    gotoxy(3,21);write('| Awal barisan terletak pada matriks ke  :            |');
    gotoxy(3,22);write('| Akhir barisan terletak pada matriks ke :            |');
    gotoxy(3,23);write('└─────────────────────────────────────────────────────┘');
    gotoxy(46,21);readln(i);
    gotoxy(46,22);readln(j);
    textcolor(0);
    textbackground(15);
  end;

  {prosedure untuk menampilkan form pertanyaan untuk masukan posisi pemecahan
  rangkaian pada proses penyelesaian tanpa menggunakan algoritma DP}
  procedure FormTanya;
  begin
     textbackground(1);
     textcolor(15);
     gotoxy(3,19);write('┌──LETAK PEMECAHAN BARISAN ──────────────────────────┐');
     gotoxy(3,20);write('| Awal barisan :                    Akhir barisan : |');
     gotoxy(3,21);write('| Pemecahan dilakukan pada posisi setelah matriks ke:|');
     gotoxy(3,22);write('└─────────────────────────────────────────────────────┘');
     textcolor(0);
     textbackground(15);
  end;

  {prosedure untuk membuka suatu file pada saat pemasukan melalui open file}
  procedure BukaFile(var matrix: matrik; var p : bitel;var n : byte);
  var p1,p2          : bitel;
      brs,klm,i,k,j  : byte;
      nilai          : integer;
      NamaFile       : string;
      Berkas         : text;
```

```
begin
  FormBSFile(NamaFile);
  Assign(Berkas,NamaFile);
  {$I-}
  reset(Berkas);
  {$I+}
  if IOresult <> 0 then
  begin
    clrscr;
    Peringatan(' File tidak ditemukan ( tekan <enter> untuk ke menu utama  ! ');
    textbackground(4); textcolor(15);
    gotoxy(20,13);write('                                    ');
    gotoxy(20,14);write(' File tidak ditemukan ...    ');
    gotoxy(20,15);write('                                    ');
    gotoxy(48,14);readln;textbackground(15); textcolor(0);
  end
  else
  begin
    while not Eof(Berkas) do
    begin
     readln(Berkas,n);
     for k := 1 to n do
     begin
      buat_simpul(matrix[k],0,0,0);
      matrix[k]^.bawah := matrix[k];
      matrix[k]^.kanan := matrix[k];
      readln(Berkas,p1[k],p2[k]);
      for i := 1 to p1[k] do
       begin
        brs := i;
        for j := 1 to p2[k] do
         begin
          klm := j;
          readln(Berkas,nilai);
          sisip_elemen(matrix[k],brs,klm,nilai);
         end;
       end;
      p[k] := p2[k];
     end;
     p[0] := p1[1];
    end;
    close(Berkas);
    FormAwalAkhir(ii,jj,byk);
  end;
end;

{prosedure untuk memasukkan posisi pemecahan rangkaian}
procedure tanya(var s : bite;i,j : byte);
begin
  if j > i then
   begin
    gotoxy(20,20);write(i);
    gotoxy(56,20);write(j);
    gotoxy(58,21);readln(s[i,j]);
    tanya(s,i,s[i,j]);
    tanya(s,s[i,j]+1,j);
```

```
    end;
end;

{prosedure untuk menentukan posisi pemecahan rangkaian dengan menggunakan
algoritma DP}
procedure order( var s:bite;n :byte;p :bite1);
var i,j,k,l : byte;
    m:mskalar;
    q : longint;
begin
  for i:= 1 to n do
  m[i,i] :=0;
  for l := 2 to n do
    for i:= 1 to n-l+1 do
    begin
      j:=i+l-1;
      m[i,j]:= 2147483647;
      for k:=i to j-1 do
      begin
        q:=m[i,k]+m[k+1,j]+p[i-1]*p[k]*p[j];
        if (q > 0) and (q < m[i,j]) then
        begin
          m[i,j]:= q;
          s[i,j]:= k;
        end;
      end;
   end;
end;

{prosedur untuk proses mengalikan rangkaian matriks}
procedure barisan(matrix: matrik; s: bite;i,j:byte ;var c:Pmat);
var x,y : Pmat;
begin
   buat_simpul(c,0,0,0);
   if j > i then
   begin
     barisan(matrix,s,i,s[i,j],x);
     barisan(matrix,s,s[i,j]+1,j,y);
     kali_matriks(x,y,c);
   end
else
   c := h[i];
end;

{prosedur untuk menampilkan rangkaian matriks lengkap dengan letak tanda
kurungnya}
Procedure BentukPerkalian(i,j :byte;s:bite);
var k: byte;
begin
  if i = j then write('A',i)
  else
   begin
    k := s[i,j];
    write('(');
    BentukPerkalian(i,k,s);
    write('*');
```

```
      BentukPerkalian(k+1,j,s);
      write(')');
    end;
  end;

  {prosedur untuk menampilkan form menu utama}
  procedure FormMenuUtama;
  var i : byte;
  begin
    textbackground(15);
    clrscr;
    Petunjuk('Gerakkan kursor '+chr(24)+' atau '+chr(25)+' atau tekan huruf dengan
              warna berbeda ');
    textcolor(1);
    gotoxy(12,2);write('APLIKASI PENYELESAIAN MASALAH PERKALIAN RANGKAIAN
                        MATRIKS');
    gotoxy(20,3);write('DENGAN ALGORITMA PEMROGRAMAN DINAMIK');
    textcolor(4);
    gotoxy(22,7);write('▒▒▒▒▒▒▒▒ MENU PENGERJAAN ▒▒▒▒▒▒▒▒');
    gotoxy(22,8);write('▒');
    gotoxy(53,8);write('▒');
    for i:= 1 to 6 do
    begin
      textcolor(4);
      gotoxy(22,i+8);write('▒');
      gotoxy(53,i+8);write('▒');
      with DataMenu do
      begin
        textcolor(4);textbackground(15);
        gotoxy(27,posisiMenu[i]);textbackground(1);textcolor(7);write(IsiMenu[i]);
        gotoxy(posisiKarakterMenu[i],i+8);textcolor(14);write(KarakterMenu[i]);
      end;
    end;
    textcolor(4);gotoxy(22,i+10);write('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');
    gotoxy(22,i+9);write('▒');
    gotoxy(53,i+9);write('▒');
    move(layar,LayarMenuUtama,4000);
  end;

  {prosedure untuk menampilkan form menu proses}
  procedure FormMenuProses;
  var i : byte;
  begin
    textbackground(15);textcolor(4);
    clrscr;
    Petunjuk('Gerakkan kursor '+chr(24)+' atau '+chr(25)+' atau tekan huruf dengan
              warna berbeda ');
    gotoxy(12,2);write('APLIKASI PENYELESAIAN MASALAH PERKALIAN RANGKAIAN
                        MATRIKS');
    gotoxy(22,7);write('▒▒▒▒▒▒▒▒▒▒▒▒   PILIHAN  ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');
    gotoxy(22,8);write('▒');
    gotoxy(52,8);write('▒');
    for i:= 1 to 3 do
    begin
      textcolor(4);textbackground(15);
      gotoxy(22,i+8);write('▒');
```

```
      gotoxy(52,i+8);write('▒');
      with DataMenuProses do
      begin
        textcolor(0);
        gotoxy(27,posisiMenu[i]);textbackground(1);textcolor(7);write(IsiMenu[i]);
        gotoxy(posisiKarakterMenu[1],i+8);textcolor(14);write(KarakterMenu[i]);
      end;
    end;
    textcolor(4);gotoxy(22,i+10);write('▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒');
    gotoxy(22,i+9);write('▒');
    gotoxy(52,i+9);write('▒');
    move(layar,LayarMenuProses,4000);
  end;

  {prosedur untuk melakukan proses perkalian (dengan algoritma DP atau tanpa DP),
  mencetak hasilnya, dan mendapatkan lama eksekusi}
  procedure Proses(var time : word);
  begin
     GetTime(h1,m1,s1,hund1);
     barisan(h,tt,ii,jj,c);
     GetTime(h2,m2,s2,hund2);
     time := (h2-h1)*3600*100 + (m2-m1)*60*100 + (s2-s1)*100 + (hund2-hund1);
     cetak_matriks(c);
  end;

  {prosedur induk untuk melakukan proses perkalian rangkaian matriks}
  procedure MenuProses;
  var i,j,pilih,pilihprev,
      baris1      :byte;
      TekanMenu,selesai,t   : char;
      h3,h4,m3,m4,s3,s4,hund3,hund4,time,time1,waktu : word;
  begin
    KembalikanLayar(1,1,80,25,LayarMenuProses);
    pilih:=1;
    baris1:=DataMenuProses.PosisiMenu[pilih];
    ChAtributLayar(27,baris1,47,baris1,$62);
    repeat
      TekanMenu :=' ';
      PilihPrev := Pilih;
      gotoxy(28,baris1);
      TekanMenu := readkey;
      if TekanMenu <> #0 then
      begin
       case upcase(TekanMenu) of
        'T': pilih:=1;
        'D': pilih:=2;
        'E': pilih:=3;
        #13: pilih:=PilihPrev;
        else
         write(#7);
       end;
      end;
      if TekanMenu = #0 then
      begin
        TekanMenu := readkey;
        case TekanMenu of
```

```
      #72 : begin
              if pilih = 1 then
                 pilih := 3
              else
                 pilih := pilih-1;
            end;
      #80 : begin
              if pilih = 3 then
                 pilih := 1
              else
                 pilih := pilih+1;
            end;
      else
        write(#7);
    end;
  end;
  if Pilih <> PilihPrev then
  begin
    baris1 := DataMenuProses.PosisiMenu[PilihPrev];
    KembalikanLayar(27,baris1,50,baris1,LayarMenuProses);
    baris1 := DataMenuProses.PosisiMenu[pilih];
    ChAtributLayar(27,baris1,47,baris1,$62);
  end;
  if (upcase(TekanMenu) = 'E')or((pilih = 3)and(TekanMenu = #13)) then
  exit;
  KembalikanLayar(30,20,65,20,LayarMenuProses);
  if TekanMenu = #13 then
  begin
    if pilih = 1 then
    begin
      FormTanya;
      tanya(tt,ii,jj);
    end;
    if pilih = 2 then
    begin
      GetTime(h3,m3,s3,hund3);
      order(tt,byk,r);
      GetTime(h4,m4,s4,hund4);
      time := (h4-h3)*3600*100 + (m4-m3)*60*100 + (s4-s3)*100 + hund4 - hund3;
    end;
  clrscr;
  writeln('Rangkaian matrik akan dikalikan menurut peletakan tanda kurung :');
  writeln;BentukPerkalian(ii,jj,tt);
  writeln;writeln;
  writeln('Hasil akhir perkalian adalah sebagai berikut : ');
  writeln;
  Proses(time1);
  if pilih = 2 then
    waktu := time1 + time
  else
    waktu := time1;
  writeln('Waktu mulai : ',h1,':',m1,':',s1,'.',hund1);
  writeln('Waktu selesai : ',h2,':',m2,':',s2,'.',hund2);
  writeln('Lama eksekusi : ',waktu,' per 100 detik');
  readln;
  KembalikanLayar(1,1,80,25,LayarMenuProses);
```

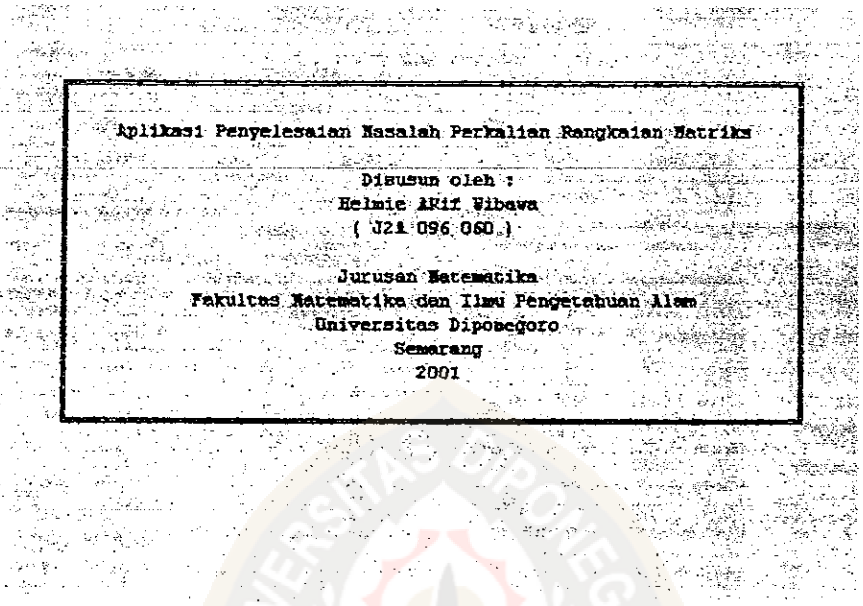
```
    end;
  until false;
end;

{prosedur untuk melakukan proses menu utama}
procedure MenuUtama;
var i,j,pilih,pilihprev,
    baris1      :byte;
    TekanMenu,selesai,t   : char;
begin
  KembalikanLayar(1,1,80,25,LayarMenuUtama);
  pilih:=1;
  baris1:=DataMenu.PosisiMenu[pilih];
  ChAtributLayar(27,baris1,49,baris1,$62);
  repeat
    TekanMenu :=' ';
    PilihPrev := Pilih;
    gotoxy(28,baris1);
    TekanMenu := readkey;
    if TekanMenu <> #0 then
    begin
     case upcase(TekanMenu) of
      'I': pilih:=1;
      'M': pilih:=2;
      'B': pilih:=3;
      'S': pilih:=4;
      'P': pilih:=5;
      'E': pilih:=6;
      #13: pilih:=PilihPrev;
      else
       write(#7);
     end;
    end;
    if TekanMenu = #0 then
    begin
      TekanMenu := readkey;
      case TekanMenu of
        #72 : begin
                if pilih = 1 then
                   pilih := 6
                else
                   pilih := pilih-1;
              end;
        #80 : begin
                if pilih = 6 then
                   pilih := 1
                else
                   pilih := pilih+1;
              end;
        else
          write(#7);
      end;
    end;
    if Pilih <> PilihPrev then
    begin
      baris1 := DataMenu.PosisiMenu[PilihPrev];
```

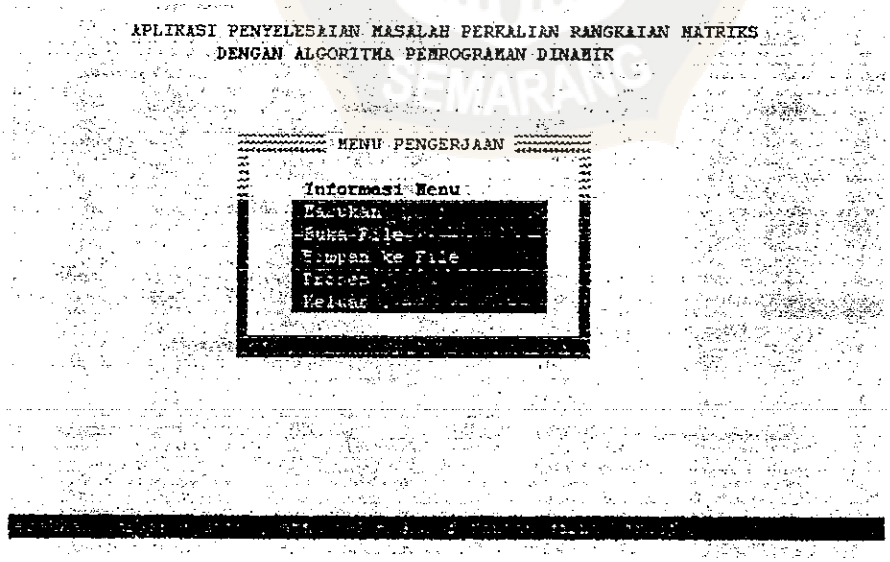
```
        KembalikanLayar(27,baris1,50,baris1,LayarMenuUtama);
        baris1 := DataMenu.PosisiMenu[pilih];
        ChAtributLayar(27,baris1,49,baris1,$62);
      end;
      if (upcase(TekanMenu) = 'E')or((pilih = 6)and(TekanMenu = #13)) then
      begin
        selesai :=' ';
        while not ((selesai = 'Y') or (selesai = 'T')) do
        begin
          textcolor(1);
          gotoxy(28,20);write(' Anda yakin (Y/T) ? ');read(selesai);
          selesai := upcase(selesai); textcolor(0);
        end;
        if selesai = 'Y' then halt;
        KembalikanLayar(30,20,65,20,LayarMenuUtama);
      end;
      if TekanMenu = #13 then
      begin
        if pilih = 1 then
        begin
          InformasiMenuUtama;
          write('belom dibuat');
        end;
        if pilih = 2 then
        begin
          Input(r,r1,r2,h,byk,ii,jj);
        end;
        if pilih = 3 then
        begin
          TulisFile;
          BukaFile(h,r,byk);
        end;
        if pilih = 4 then
        begin
          SimpanKeFile(h,r1,r2,byk);
        end;
        if pilih = 5 then
        begin
           MenuProses;
        end;
        KembalikanLayar(1,1,80,25,LayarMenuUtama);
      end;
    until false;
  end;

  {program utama}
  begin
     clrscr;
     FormMenuUtama;
     FormMenuProses;
     FormMasukan;
     Cover;
     MenuUtama;
  end.
```

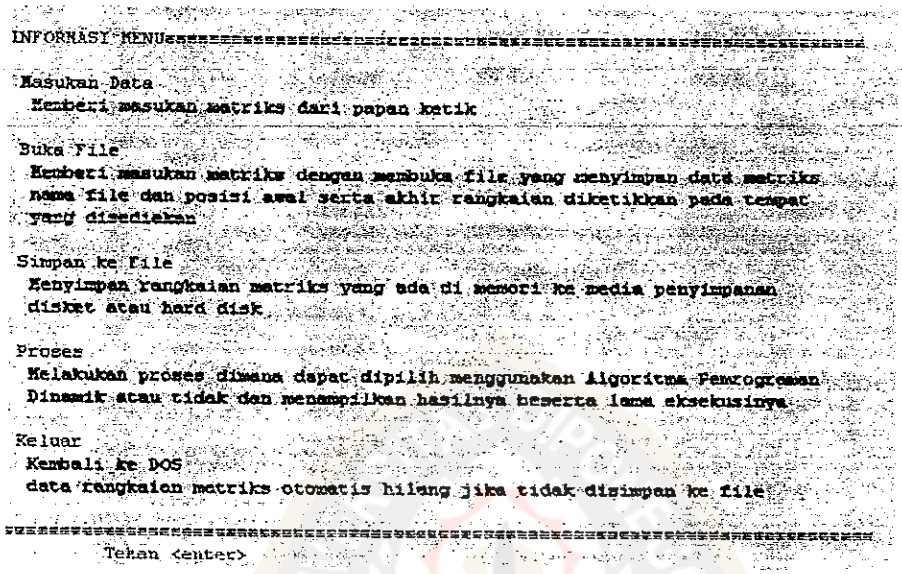
LAMPIRAN OUTPUT PROGRAM



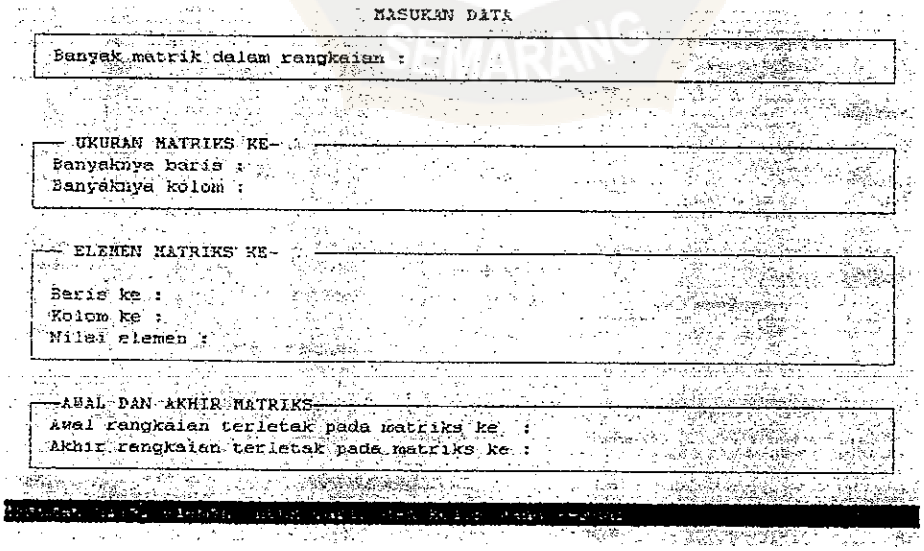
Tampilan pembukaan program



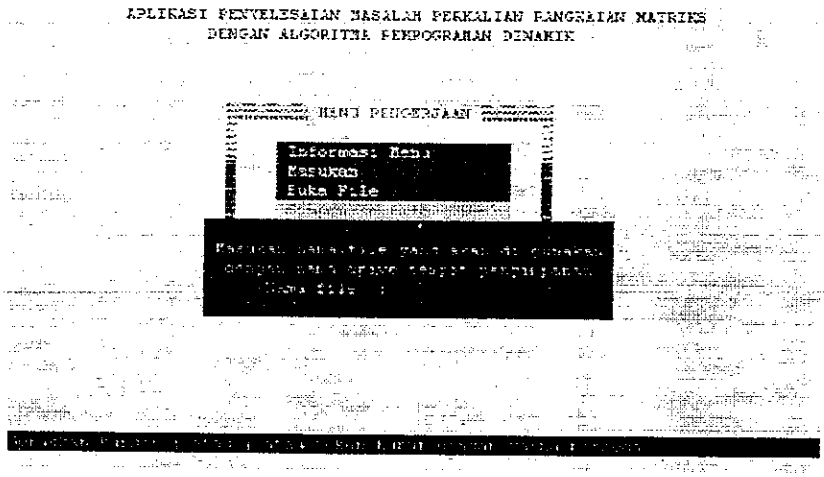
Tampilan Menu Utama



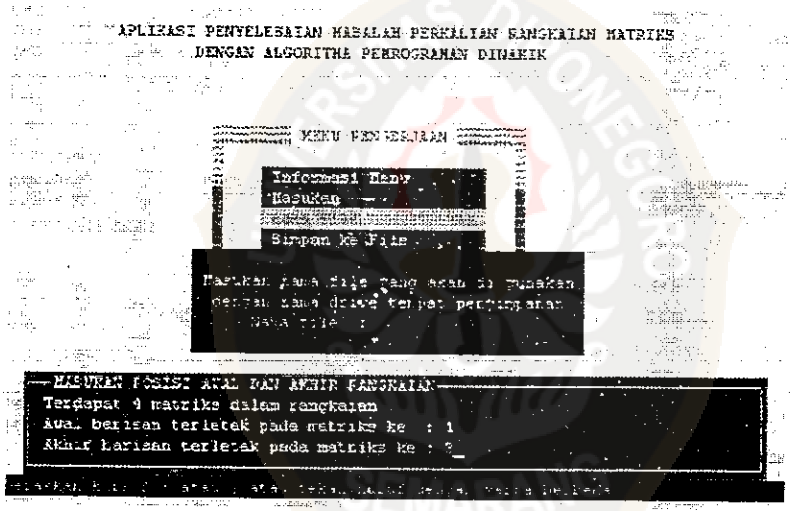
Tampilan Informasi Menu



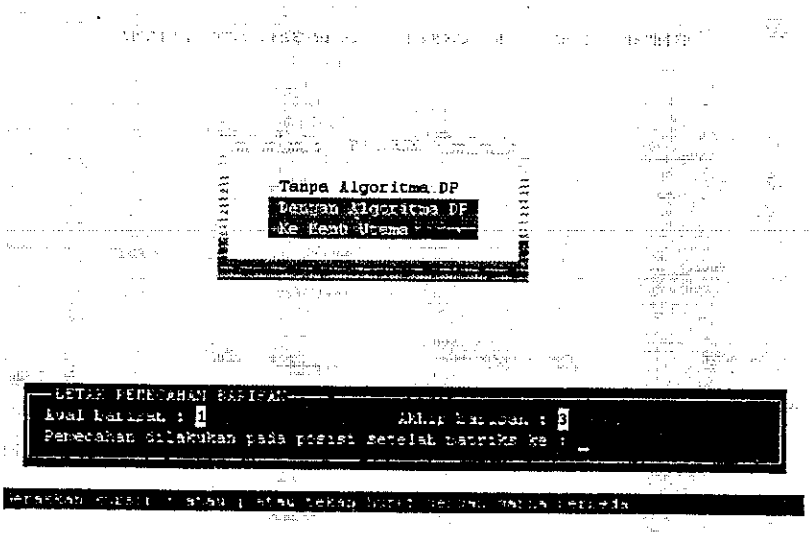
Tampilan Masukan



Tampilan Simpan ke File



Tampilan Buka File



Tampilan Proses Tanpa DP

```
((A1*A2)*A3)*A4)

Hasil akhir perkalian adalah sebagai berikut :

Elemen baris 1 kolom 1 =      400
Elemen baris 1 kolom 2 =      400
Elemen baris 1 kolom 3 =      400
Elemen baris 1 kolom 4 =      400
Elemen baris 1 kolom 5 =      400
Elemen baris 1 kolom 6 =      400
Elemen baris 1 kolom 7 =      400
Elemen baris 1 kolom 8 =      400
Elemen baris 1 kolom 9 =      400
Elemen baris 1 kolom 10 =     400
Elemen baris 1 kolom 11 =     400
Elemen baris 1 kolom 12 =     400
Elemen baris 1 kolom 13 =     400
Elemen baris 1 kolom 14 =     400
Elemen baris 1 kolom 15 =     400
Elemen baris 1 kolom 16 =     400
Elemen baris 1 kolom 17 =     400
Elemen baris 1 kolom 18 =     400
Elemen baris 1 kolom 19 =     400
Elemen baris 1 kolom 20 =     400

Elemen baris 30 kolom 16 =    400
Elemen baris 30 kolom 17 =    400
Elemen baris 30 kolom 18 =    400
Elemen baris 30 kolom 19 =    400
Elemen baris 30 kolom 20 =    400
Elemen baris 30 kolom 21 =    400
Elemen baris 30 kolom 22 =    400
Elemen baris 30 kolom 23 =    400
Elemen baris 30 kolom 24 =    400
Elemen baris 30 kolom 25 =    400

Waktu mulai : 5:16:51.80
Waktu selesai : 5:16:51.91
Lama eksekusi : 11 milidetik
```

Tampilan Sebagian Hasil Akhir

LAMPIRAN PERBANDINGAN WAKTU EKSEKUSI

Data :

Eksekusi dilakukan pada komputer dengan spesifikasi :

- Prosessor = pentium 133
- Ram = 16 MHz
- Harddisk = 1,2 GB

Banyaknya matriks dalam rangkaian adalah 4 matriks dengan elemen pada setiap baris dan kolom setiap matriks adalah 1.

Ukuran matriks

A_1	30 x 1	A_3	40 x 10
A_2	1 x 40	A_4	10 x 25

Hasil

- Letak tanda kurung : $(A_1.(A_2.(A_3.A_4)))$

  Lama eksekusi : 5 milidetik

- Letak tanda kurung : $(A_1.((A_2.A_3).A_4))$

  Lama eksekusi : 0 milidetik

- Letak tanda kurung : $((A_1.A_2).(A_3.A_4))$

  Lama eksekusi : 11 milidetik

- Letak tanda kurung : $((A_1.(A_2.A_3)).A_4)$

  Lama eksekusi : 6 milidetik

- Letak tanda kurung : $(((A_1.A_2).A_3).A_4)$

  Lama eksekusi : 6 milidetik