Penggunaan Dynamic Programming Untuk Menyelesaikan Permasalahan Optimasi Matrix Chain Multiplication

Satria Ady Pradana - 13510030
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
satria@arc.itb.ac.id, master@xathrya.web.id

***Abstrak*—Perkalian matriks merupakan salah satu operasi umum yang terjadi pada bidang komputasi. Suatu komputasi seringkali melibatkan perkalian-berantai matriks.**

Makalah ini akan membahas penggunaan algoritma dynamic programming untuk mencari solusi optimum dalam menentukan urutan perkalian.

***Kata kunci*—matriks, perkalian berantai, optimasi, *dynamic programming*.**

I. Matrix dan Operasinya

Mengacu pada istilah matematika, matriks adalah kumpulan bilangan, simbol, atau ekspresi yang disusun membentuk persegi panjang atau tabel menurut baris dan kolom.

Sebuah matriks dinotasikan dengan huruf kapital, seperti: A, M, K, dsb. Ukuran suatu matriks, disebut pula sebagai ordo matriks, ditentukan oleh panjang dan lebar matriks tersebut. Secara umum, matriks dituliskan sebagai berikut:

$$A_{mxn} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Gambar 1: Matriks A berukuran m x n

Pada gambar di atas, sebuah matriks bernama A definisikan sebagai matriks dengan ukuran *m* baris dan *n* kolom dengan m,n > 0. Terdapat m x n buah elemen yakni a_{11}, a_{12}, ..., a_{mn} yang tersusun membentuk tabel. Elemen-elemen tersebut tersusun berdasarkan baris dan kolom.

Berikut ini adalah contoh lain matriks.

$$A_{3x2} = \begin{bmatrix} 135 & 132 \\ 182 & 180 \\ 235 & 181 \end{bmatrix}$$

Gambar 2: Matriks A berukuran 3x2

Pada gambar 2 terlihat bahwa sebuah matriks A didefinisikan sebagai matriks berukuran 3 x 2 atau 3 baris dan 2 kolom.

Setiap nilai / simbol / ekspresi yang membentuk matriks disebut sebagai elemen matriks. Elemen matriks diidentifikasi sesuai dengan letak sel, yakni koordinat pertemuan baris dan kolom. Dengan mengambil contoh pada gambar 1, elemen a_{23} adalah elemen matriks A pada baris ke-2 dan kolom ke-3. Contoh lain adalah nilai 135 merupakan elemen matriks pada baris pertama kolom pertama sementara nilai 182 adalah elemen matriks pada baris kedua kolom pertama.

Secara luas matriks dimanfaatkan dalam berbagai jenis komputasi. Beberapa contoh yang dapat diambil antara lain: menemukan solusi sistem persamaan linear, transformasi linear (translasi, dilatasi, rotasi suatu objek), perhitungan pada bidang *engineering*, dll.

Berdasarkan ordonya matriks dapat diklasifikasikan sebagai: matriks bujur sangkar/persegi, matriks baris, matriks kolom, matriks tegak, matriks datar.

Matriks bujur sangkar atau matriks persegi adalah matriks yang memiliki ordo n x n. Dengan kata lain matriks ini memiliki jumlah baris yang sama dengan jumlah kolomnya. Matriks ini sering pula disebut sebagai matriks berordo n.

$$A_{2x2} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Gambar 3: Matriks persegi 2x2

Matriks baris adalah matriks dengan ordo 1 x n. Dengan kata lain matriks ini hanya memiliki satu buah baris saja.

$$A_{1x3} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Gambar 4: Matriks Baris ordo 1x3

Matriks kolom adalah kebalikan dari matriks baris. Matriks ini adalah matriks dengan ordo n x 1 atau

memiliki n buah baris dengan hanya sebuah kolom yang menjadi penyusunnya.

Matriks tegak adalah matriks berukuran m x n dengan m .> n. Jika m < n maka matriks tersebut disebut sebagai matriks datar. Gambar 2 merupakan salah satu contoh matriks tegak. Sebuah matriks baris dapat dikatakan sebagai matriks datar, begitu juga matriks kolom dapat dikatakan sebagai matriks tegak karena memenuhi karakteristik.

Berdasarkan elemen penyusunnya, matriks dapat diklasifikasikan sebagai: matriks nol, matriks diagonal, matriks identitas, matriks skalar, matriks simetri, matriks segitiga atas, dan matriks segitiga bawah, .

Matriks nol adalah matriks berukuran m x n dengan seluruh elemen penyusunnya adalah 0 (nol).

Matriks diagonal adalah matriks persegi dengan semua elemen di atas dan di bawah diagonal bernilai nol. Secara formal didefinisikan bahwa untuk elemen matriks pada baris i dan j dengan $i \neq j$ bernilai 0.

Matriks identitas atau matriks satuan adalah matriks bentuk khusus matriks diagonal dengan semua elemen pembentuk diagonal memiliki nilai 1 (satu). Matriks ini adalah matriks khusus uang disimbolkan dengan *I* dari kata identity..

Matriks skalar adalah bentuk khusus matriks diagonal dengan semua elemen pada diagonal bernilai sama. Matriks identitas adalah matriks bentuk khusus dengan setiap elemen diagonal bernilai satu.

Matriks simetri adalah matriks persegi dengan nilai elemen a_{ij} sama dengan nilai elemen a_{ji}.

$$A_{3x3} = \begin{bmatrix} 1 & 5 & 6 \\ 5 & 2 & 3 \\ 6 & 3 & 3 \end{bmatrix}$$

Gambar 5: Matriks Simetri 3x3

Matriks segitiga atas adalah bentuk khusus matriks persegi dengan elemen pada matriks di bawah diagonal bernilai 0. Dengan kata lain, setiap elemen a_{ij} dengan i > j bernilai 0.

Matriks segitiga bawah adalah lawan dari matriks segitiga atas. Pada matriks segitiga bawah, setiap elemen yang berada di atas diagonal bernilai 0. Secara formal berlaku $a_{ij} = 0$ dengan i < j.

Matriks seperti halnya bilangan biasa memiliki operasi-operasi spesifik terhadapnya. Beberapa operasi dasar pada matriks antara lain: transpos, invers, penjumlahan, pengurangan, perkalian, dan pembagian. Namun dalam makalah ini hanya akan dibahas mengenai operasi perkalian saja, terutama tentang perkalian matriks dengan matriks.

Perkalian matriks adalah suatu operasi biner yang melibatkan dua buah matriks. Suatu operasi perkalian terhadap matriks A dan matriks B hanya dapat dilakukan jika jumlah kolom A sama dengan jumlah baris B.

Misal:

$$A_{2x3} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 1 \end{bmatrix}$$

dan

$$B_{3x4} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 5 & 6 \\ 3 & 4 & 5 & 6 \end{bmatrix}$$

Operasi perkalian antara A dan B akan menghasilkan sebuah matriks pula. Misal C adalah matriks hasil perkalian A dan B sesuai dengna contoh di atas. Matriks C didefinisikan sebagai sebuah matriks ordo 2x4.

Secara umum perkalian antara A_{mxn} dengan B_{nxr} akan menghasilkan matriks baru dengan ordo C_{mxr}. Nilai elemen matriks C didapatkan dari hasil penjumlahan perkalian elemen baris i pada matriks A dan elemen kolom j pada matriks B. Dalam persamaan matematika, definisi tersebut dapat dituliskan sebagai:

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}$$

Persamaan 1: Nilai elemen matriks C

Dengan menggunakan definisi tersebut, matriks C bernilai sebagai berikut:

$$A \times B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 5 & 6 \\ 3 & 4 & 5 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 1+2+9 & 2+4+12 & 3+10+15 & 4+12+18 \\ 4+5+3 & 8+10+4 & 12+25+5 & 16+30+6 \end{bmatrix}$$

$$= \begin{bmatrix} 12 & 28 & 28 & 34 \\ 12 & 32 & 42 & 52 \end{bmatrix}$$

Perkalian matris bersifat asosiatif. Hal ini berarti kedua perkalian berikut akan menghasilkan nilai yang sama:

$$(AB)C = A(BC)$$

Persamaan 2: Perkalian matriks bersifat asosiatif

Namun perkalian pada matriks tidak bersifat komutatif. Sehingga:

$$AB \neq BA$$

Persamaan 3: Perkalian matriks tidak bersifat komutatif

Hal ini bisa dibuktikan sebagai berikut:

$$A_{2x2} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B_{2x2} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$A \,.\, B = \begin{bmatrix} 3 & 3 \\ 7 & 7 \end{bmatrix}$$

$$B \,.\, A = \begin{bmatrix} 4 & 6 \\ 4 & 6 \end{bmatrix}$$

Gambar 6: Perkalian matriks tidak bersifat komutatif

Namun sebuah pengecualian apabila sebuah matriks dikalikan dengan matriks identitas maka hasilnya adalah matriks itu sendiri. Sehingga kedua persamaan di bawah ini benar.

$$M \,.\, I = M$$
$$I \,.\, M = M$$

Persamaan 4: Perkalian dengan matriks identitas

II. PERKALIAN BERANTAI

Diberikan beberapa objek (dapat berupa nilai / simbol / ekspresi) yang disusun berderet, perkalian berantai adalah operasi berantai yang terjadi pada deretan untuk menghasilkan suatu nilai akhir.

Pada perkalian berantai, nilai pada suku ke-i akan dikalikan dengan nilai pada suku (i+1). Hasil perkalian ini, misal x, akan menjadi operan bagi operasi perkalian selanjutnya. Operasi ini akan terus berlanjut hingga mencapai suku ke-n.

Lebih lanjut, jika terdapat n buah suku, maka akan terjadi (n-1) buah perkalian.

$$X = a_1 a_2 a_3 \ldots a_n$$

Gambar 7: Perkalian berantai

Pada perkalian aljabar biasa berlaku sifat komutatif sehingga deret suku dapat dipermutasi sehingga menjadi suatu urutan tertentu tanpa mengubah hasil akhir operasi.

Operasi perkalian berantai pada matrix serupa dengan operasi perkalian di atas. Namun perkalian matriks tidak bersifat komutatif sehingga perkalian matriks yang dikerjakan akan memiliki satu buah variasi penempatan saja.

Perkalian berantai dapat juga dikelompokkan ke dalam kelompok-kelompok kecil yang dipisahkan dengan tanda kurung (). Sesuai dengan urutan operator, ekspresi yang berada di dalam tanda kurung akan dikerjakan terlebih dahulu. Hal ini dalam beberapa kondisi dapat memperingkas perhitungan.

$$a.b.c.d = (a.b).(c.d)$$

Gambar 8: pengelompokan suku

Hal yang sama juga berlaku pada perkalian berantai dengan matriks sebagai operan.

III. DYNAMIC PROGRAMMING

Dynamic Programming atau pemrograman dinamis adalah salah satu strategi algoritma atau desain algoritma untuk menyelesaikan persoalan. Karakteristik permasalahan yang dapat diselesaikan adalah masalah yang dapat dilihat sebagai hasil dari urutan pemilihan *(sequence of decisions*).

Dynamic Programming memecah masalah ke dalam beberapa upamasalah yang lebih sederhana. Secara luas dynamic programming digunakan apabila permasalahan tersusun dari upamasalah yang tumpang tindih (*overlapping subsequence*).

Ide utama di balik dynamic programming sangat sederhana. Untuk dapat menyelesaikan permasalahan, bagian permasalahan yang berbeda (upamasalah) harus diselesaikan terlebih dahulu. Solusi didapatkan dari hasil penggabungan upamasalah-upamasalah untuk mencapai solusi global.

Sekilas teknik ini mirip dengan teknik *Divide and Conquer* karena sifatnya yang membagi masalah menjadi permasalahan kecil. Namun perbedaan utama antara *dynamic programming* dan *divide and conquer* terletak pada definisi upamasalah serta strategi pemecahannya.

Pada *divide and conquer*, upamasalah merupakan bagian kecil dari masalah yang memiliki karakteristik sama dengan persoalan awal. Upa masalah pada ukuran paling kecil dapat diselesaikan dengan mudah. Setelah penyelesaian, upamasalah akan digabungkan dengan upamasalah lain untuk mendapat solusi yang lebih besar.

Lain halnya dengan *dynamic programming*. Upamasalah merupakan masalah yang tumpang tindih, artinya solusi suatu upamasalah dapat menjadi input untuk menyelesaikan upamasalah lain. Hasil dari pemecahan upamasalah akan disimpan atau dimemoisasi. Ketika solusi yang sama dibutuhkan upamasalah tak perlu dikomputasi ulang, cukup mengambil nilai solusi dari tabel solusi.

Desain *dynamic programming* sangat berguna terutama ketika jumlah upamasalah yang tumpang tindih tumbuh secara eksponensial sebagai fungsi dari input.

IV. PERSOALAN MATRIX CHAIN MULTIPLICATION

Berdasarkan definisi yang telah disebutkan sebelumnya, perkalian-berantai matriks atau *matrix chain multiplication* adalah operasi perkalian yang terjadi secara terus-menerus.

Persoalan perkalian berantai merupakan persoalan optimisasi yang dapat diselesaikan menggunakan dynamic programming.

Persoalan utama bukan pada bagaimana melakukan perkalian (mendapatkan nilai dari operasi) namun bagaimana menentukan urutan perkalian yang dilakukan.

Untuk menghitung hasil dari perkalian berantai, terdapat berbagai cara atau pilihan. Berdasarkan sifat asosiatif, suatu ekspresi perkalian berantai dapat dikelompokkan ke dalam partisi-partisi kecil di dalam tanda kurung () yang berisi perkalian.

Pengelompokan / partisi ini dapat menghasilkan bermacam-macam variasi / cara. Namun nilai yang dihasilkan untuk setiap pilihan adalah sama.

Pada aljabar biasa, sebuah ekspresi perkalian berantai dapat dipecah sebagai berikut:

$$abcd = (ab)(cd) = (ac)(bd) = (ad)(bc) = a(bc)d$$

Gambar 9: partisi tanda kurung

Dapat dilihat bahwa penentuan partisi dapat ditentukan selain dari letak tanda kurung (elemen-elemen yang dikelompokkan bersama) juga ditentukan oleh permutasi anggota yang dioperasikan. Dalam kasus di atas, Dua tanda kurung digunakan untuk mengelompokkan a dan b serta c dan d. Namun solusi lain yang didapat adalah dengan mengelompokkan a dan c sedangkan b dan d dikelompokkan ke dalam partisi lain.

Namun pendekatan ini tidak berlaku pada perkalian matriks karena perkalian matriks tidak bersifat komutatif. Keuntungannya adalah bahwa kita dapat mengabaikan kondisi permutasi untuk memikirkan solusi. Fokus utama hanya terdapat pada partisi objek.

Pada *Matrix Chain Multiplication*, bermacam-macam variasi solusi yang dapat dihasilkan uuntuk sebuah ekspresi. Namun semua kemungkinan akan menghasilkan hasil yang sama. Perbedaan utama terletak pada biaya yang diperlukan. Biaya yang dimaksud adalah banyaknya perkalian dan penjumlahan yang harus dilakukan.

Misal terdapat empat buah matriks: A, B, C, dan D. Berikut adalah partisi yang dapat dihasilkan dengan nilai akhir yang sama:

$$(ABC)D = (AB)(CD) = A(BCD) = A(BC)D = \cdots$$

Gambar 10: Partisi

Partisi akan sangat memengaruhi jumlah operasi aritmatika yang dilakukan. Khususnya bagi operasi perkalian dan operasi penjumlahan.

Misal A adalah matriks 10x30, B adalah matriks 30x5, dan C adalah matriks 5x60. Banyaknya operasi aritmatika yang terjadi untuk setiap kemungkinan perkalian matriks adalah sebagai berikut:

$$(AB)C = (10x30x5) + (10x5x60) = 1500 + 3000 = 4500$$
$$A(BC) = (30x5x60) + (10x30x60) = 9000 + 18000 = 27000$$

Gambar 11: Perbandingan operasi

Dapat kita lihat pemilihan urutan perkalian yang berbeda akan menghasilkan jumlah operasi yang berbeda pula. Dalam hal ini dengan mengalikan A dengna B kemudian mengalikannya dengan C akan memerlukan 4500 operasi. Sementara jika A dikalikan dengan B dan C yang telah dikalikan terlebih dahulu, operasi yang dibutuhkan adalah 27000. Perbedaan keduanya cukup besar, lebih dari 6 kali lipat jumlah operasi pertama.

Kini telah jelas permasalahan utama yang dihadapi. Bagaimana cara menentukan urutan perkalian agar memerlukan operasi minimal?

Dengan pendekatan *dynamic programming*, persoalan ini dapat dilihat sebagai penentuan sebuah struktur. Dalam hal ini adalah partisi tanda kurung ().

Masalah dipecah menjadi beberapa upamasalah. Solusi didapat dari penggabungan solusi upamasalah. Sifat perkalian matriks yang tidak komutatif memberi kita manfaat seperti yang telah dijelaskan sebelumnya. Permasalahan ini kemudian dapat dimodelkan sebagai sebuah string / deretan matriks.

Misal $A_{i..j}$ merupakan hasil dari perkalian matriks i hingga j.Dapat dilihat bahwa Ai..j merupakan hasil dari perkalian matriks p_{i-1} dan p_j dengan p adalah input nilai ukuran matriks. Untuk melakukan perkalian secara optimal, diperlukan pemilihan terdapat banyak solusi yang mungkin.

Misal untuk kasus pada gambar 11, nilai larik p adalah

$$p[\ \ ] = \{ 10, 30, 5, 60 \}$$

Persamaan 5: Nilai larik p

Langkah pertama adalah mencari struktur yang dapat menggambarkan keadaan seluruh upamasalah. Di level ini kita berasumsi bahwa perkalian terjadi merupakan operasi biner sehingga untuk setiap k, $1 \leq k \leq n-1$,

$$A_{1..n} = A_{1..k} \cdot A_{k+1..n}$$

Persamaan 6: Definisi perkalian matriks

Indeks k adalah posisi pemisahan deret menjadi dua upamasalah berbeda. Untuk setiap upamasalah, akan diselesaikan secara rekursif.

Solusi upamasalah akan disimpan ke dalam suatu tabel.

Pembangunan tabel dapat dilakukan dengan cara *top-down* maupun *bottom-up*, namun pada makalah ini akan digunakan pendekatan *bottom-up*.

Untuk 1 ≤ i ≤ j ≤ n, misal m[i,j] adalah jumlah perkalian minimal yang diperlukan untuk menghitung $A_{i..j}$. Harga optimum dapat dijabarkan melalui fungsi rekursif berikut:

Basis

Jika i = j maka deret hanya berisi satu matriks sehingga cost bernilai 0. Nilai m[i,j] akan bernilai 0.

Rekurens

Jika i < j, maka diperlukan hasil untuk $A_{i..j}$. Nilai ini bisa didapatkan dengan mempartisi $A_{i..j}$ menjadi $A_{i..k}$ dan $A_{k+1..j}$ untuk i ≤ k < j. Dengan demikian nilai optimum untuk $A_{i..k}$ dan $A_{k+1..j}$ didefinisikan sebagai m[i,k] dan m[k+1, j].

Nilai optimum didapat dari penelusuran m[i,j] yang menghasilkan nilai optimum.

Berikut adalah *pseudocode* solusi dengan pendekatan iteratif:

```
function MatrixChain( p : real[1..n] ) -> tuple
KAMUS LOKAL
    s      : real[1..n-1][2..n]
    i,j,k,L : integer
    q      : real
ALGORITMA
    for i <- 1 to n do
        m[i][i] <- 0
    for L <- 2 to n do
        for i <- to n-L+1 do
            j <- i + L - 1
            m[i][j] <- INFINITY
            for k <- i to j-1 do
                q <- m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j]
                if q < m[i,j]
                    m[i][j] <- q
                    s[i][j] <- k
    -> { m[1][n] s }
```

Kode 1: Pseudocode iteratif

Pendekatan rekursif sebagai berikut:

```
function MatrixChain2( p : integer[]; i,j: integer ) -> integer
KAMUS LOKAL
    k,min,count : integer
ALGORITMA
    if( i == j )
        -> 0
    min <- INFINITY
    for k <- i to j-1 do
        count <-  MatrixChain2(p, i, k)  +
                  MatrixChain(p,K+1,j) +
                  p[i-1]*p[k]*p[j]
        if count < min
            min <- count
    -> min
```

Kode 2: Pseudocode rekursif

V. KESIMPULAN

Perkalian berantai matriks atau matrix chain multiplication adalah salah satu permasalahan klasik di dalam bidang penentuan keputusan menggunakan *dynamic programming*. Solusi yang didapat menggunakan teknik ini cukup efisien dibandingkan harus menghitung segala kemungkinan yang ada.

REFERENCES

[1] E. Horowitz, “Computer Algorithms”, 2nd ed, Silicon Press, 2008

[2] http://www.columbia.edu/~cs2035/courses/csor4231.F11/matrix-chain.pdf

[3] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Dynamic/chainMatrixMult.htm, tanggal akses 20 Desember 2012

[4] http://www.cs.auckland.ac.nz/~jmor159/PLDS210/mat_chain.html, tanggal akses 20 Desember 2012

[5] http://www.cse.ust.hk/~dekai/271/notes/L12/L12.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21 Desember 2012

ttd

Satria Ady Pradana