Penerapan Program Dinamis Dalam Pencarian Solusi Perkalian Matriks Berantai

Shinta Marino[1], Indriani Noor Hapsari[2], Aye Aye Min[3]

Program Studi Informatika
Sekolah Teknik Elektro Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14130@students.if.itb.ac.id [1], if14150@students.if.itb.ac.id[2], if14163@students.if.itb.ac.id[3]

Abstrak

Program dinamis, adalah salah satu metode pemecahan masalah yang cukup mangkus dalam pencarian solusi berbagai masalah. Program dinamis bekerja dengan cara menguraikan solusi menjadi sekumpulan langkah yang saling berhubungan, solusi dari setiap langkah didapatkan dengan mengguakan solusi dari langkah sebelumnya dan solusi setiap langkah akan dipakai untuk menemukan solusi langkah beikutnya, sampai akhirnya tiba pada langkah terakhir. Pada langkah terakhir, solusi yang dihasilkan akan menjadi solusi untuk keseluruhan masalah yang optimal. Salah satu permasalahan yang dapat ditemukan solusinya dengan menggunakan program dinamis adalah perkalian matriks berantai (*chain matrix multiplication*). Dalam masalah perkalian matriks berantai ini solusi yang dibutuhkan bukanlah hasil akhir dari perkalian seluruh rangkaian matriks yang diberikan, akan tetapi solusi yang memberikrahukan bagaimana urutan perkalian matriks tersebut agar hasil akhir pekalian dapat dicapai dengan usaha seminimal mungkin.

***Kata kunci**: program dinamis, matriks.*

1. Pendahuluan

Ilmu informatika adalah ilmu yang mempunyai bidang penerapan sangat luas. Hampir semua permasalahan pemecahannya bisa direpresentasikan secara algoritmik. Ilmu informatika juga sangat berkaitan erat dengan matematika, ilmu yang menjadi dasar dari ilmu-ilmu lainnya. Bahkan bisa dikatakan bahwa matematika adalah ilmu utama yang mendasari ilmu informatika.

Dalam tulisan ini permasalahan yang akan dibahas dan dicari solusinya adalah permasalahan di bidang matematika, yaitu perkalian matriks berantai (*Chain Matriks Multiplication*). Perkalian matriks berantai, sesuai dengan namanya, adalah perkalian dari serangkaian matriks.

Yang harus dicari jalan keluarnya dalam hal ini adalah jika kita mengalikan matriks-matriks tersebut sesuai urutannya, proses yang dilakukan sering kali tidak efektif dan memakan waktu lama. Ini dikarenakan oleh banyaknya operasi perkalian integer yang dilakukan.

Misalnya diberikan 2 buah matriks :

A(3x5)
B(5x4)

Jumlah perkalian (usaha) yang diperlukan untuk mendapatkan hasil perkalian dari matriks tersebut adalah :

3x5x4 = 60 perkalian.

Ternyata pilihan urutan perkalian matriks yang berbeda akan membutuhkan jumlah perkalian (usaha) yang berbeda pula. Sehingga dengan memilih urutan perkalian matriks yang tepat, kita bisa menyelesaikan perkalian matriks berantai tersebut dengan lebih cepat dan efisien. Karena dengan memilih urutan perkalian yang tepat, kita dapat mereduksi jumlah perkalian yang harus dilakukan untuk mendapatkan solusi akhir dari perkalian matriks berantai tersebut

2. Deskripsi Masalah

Seperti sudah disebutkan sebelumnya, dalam permasalahan perkalian matriks berantai ini akan diberikan sederetan matriks sejumlah n buah. Pada matriks-matriks tersebut akan dilakukan operasi perkalian matriks.

Operasi perkalian matriks adalah operasi yang bersifat asosiatif, yaitu urutan operasi yang dilakukan dapat diubah-ubah dengan bebas dan tetap tidak akan berpengaruh pada hasil akhir operasi.

Misalnya diberikan 3 buah matriks, yaitu matriks A(10x5), B(5x20), dan C(20x30). Untuk melakukan operasi tersebut tedapat beberapa cara, antara lain :

(AB)C, A(BC)

Tentu sudah kita ketahui sebelumnya bahwa jika ukuran matriks yang akan dikalikan berbeda, maka usaha (dalam hal ini jumlah perkalian integer) yang harus dilakukan juga berbeda. Tentu saja usaha yang dilakukan untuk setiapcara diatas juga berbeda. Seperti pada contoh berikut

(AB)C = (10x5x20) + (10x20x30) = 1000 + 6000 = 7000
A(BC) = (5x20x30) + 10x5x30) = 3000 + 1500 = 4500

Dari contoh diatas, tampak bahwa cara kedua memberikan hasil akhir dengan usaha yang lebih sedikit. Oleh karena itu, tujuan kita adalah mencari solusi urutan perkalian matriks-matriks tersebut agar melakukan usaha seminimum mungkin.

3. Pencarian Solusi Dengan Berbagai Metode

3.1. Secara brute force

Menggunakan metode *Brute Force* berarti cara yang digunakan adalah cara yang paling lempang, yaitu mengenumerasi seluruh kemungkinan yang ada. Untuk semua kemungkinan tersebut, hitunglah jumlah operasi yang perlu dilakukan untuk mencapai solusi yang diinginkan. Setelah itu pilihlah satu kemungkinan terbaik, yaitu kemungkinan yang mempunyai jumlah operasi paling kecil.

Cara ini, sebagaimana metode *brute force* jika diterapkan pada kebanyakan persoalan lain, sangat tidak efisien. Jumlah kemungkinan yang harus di enumerasi eksponensial terhadap jumlah matriks. Kompleksitas algoritma ini adalah $O(2^n)$, dan disebut dengan istilah *Catalan numbers*. Harga yang diperlukan untuk memecahkan permasalahan ini terlalu besar, tidak efektif untuk diimplementasikan.

3.2. Menggunakan metode *greedy*

Dalam menerapkan metoda greedy pada persoalan ini, ada 2 kemungkinan yang mungkin dilakukan, yaitu:

1. Terus-menerus memilih 2 matriks yang hasil perkaliannya membutuhkan operasi paling banyak.
   contoh :
   diberikan 4 buah matriks : A(2x3), B(3x8), C(8x5), D(5x4)
   dengan ide ini didapatkan solusi :
   A((BC)D) = (3x8x5) + (3x5x4) + (2x3x4) = 120 + 60 + 24 = 204
   sedangkan ada solusi lain yang lenih efisien yaitu:
   ((AB)C)D = (2x3x8) + (2x8x5) + (2x5x4) = 48 + 80 + 40 = 168
2. Terus menerus memilih 2 matriks yang membutuhkan hasil perkalian terkecil.
   contoh:
   diberikan 4 buah matriks : A(8x3), B(3x3), C(3x5), D(5x4)
   dengan ide ini didapatkan solusi :
   A((BC)D) = (3x3x5) + (3x5x4) + (8x3x4) = 45+60+96= 201
   sedangkan ada solusi lain yang lebih efisien yaitu:
   A(B(CD)) = (3x5x4) + (3x3x4) + (8x3x4) = 60 + 36 + 96 = 192

Jadi dapat disimpulkan bahwa metode *greedy* tidak memberikan solusi yang optimum.

4. Pencarian Solusi Optimum dengan Program Dinamis

Prinsip program dinamis yang diterapkan biasanya memakai pendekatan secara rekursif, karena konsep dari program dinamis sendiri adalah memulai penyelesaian permasalahan dengan membagi permasalahan menjadi langkah-langkah tertentu dan menyelesaikan langkah tersebut satu-persatu mulai dai yang paling sederhana. Kemudian solusi dari langkah paling sederhana tersebut akan digunakan untuk mencari solusi langkah selanjutnya yang setingkat lebih besar, sampai akhirnya mencapai langkah terakhir yang akan memberikan solusi permasalahan yang sebenarnya.

Untuk menyelesaikan permasalahan ini secara rekursif pertama-tama adalah membagi rangkaian perkalian matriks tersebut menjadi 2 rangkaian yang lebih pendek. Kemudian cari usaha minimum untuk mengalikan setiap sub-rangkaian. Jumlahkan seluruh usaha tersebut dan terakhir tambahkan usaha untuk mengalikan 2 matriks terakhir. Ulangi langkah-langkah tersebut untuk setiap kemungkinan pembagian rangkaian matriks, dan pilih hasil paling minimum dari seluruh kemungkinan tersebut.

Ternyata, langkah-langkah tersebut setelah diterapkan memberikan hasil yang sama tidak efektifnya dengan menggunakan metoda *brute force* yang mengenumerasikan segala kemungkinan.

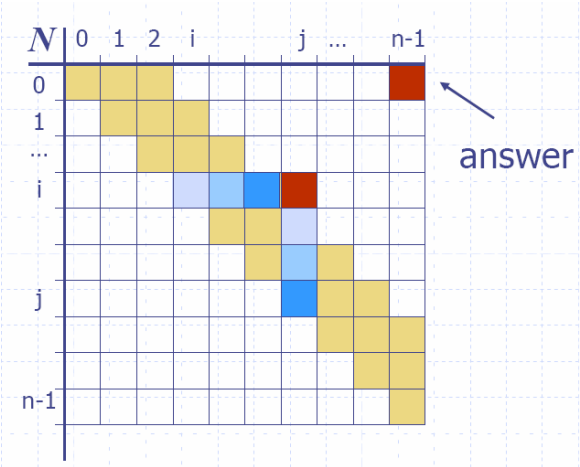
Hal ini ternyata terjadi karena dalam langkah-langkah tersebut banyak operasi perkalian yang secara tidak langsung dilakukan berulang-ulang (*overlapping*). Misalnya jika kita menerapkan proses rekursif diatas untuk mencari cara paling efektif untuk mencari mengalikan 3 matriks ABC, kita akan mencari hasil optimum untuk perkalian AB dan ABC. Padahal untuk mendapatkan solusi optimum

untuk perkalian ABC juga dibutuhkan solusi dari AB, dan dalam proses mencari solusi optimum ABC, perhitungan untuk menemukan solusi AB dilakukan kembali. Semakin panjang rantai pekalian matriks, smakin banyak pula pengulangan-pengulangan yang tidak perlu seperti ini terjadi.

Untuk mengatasi permasalahan ini, muncul sebuah ide yaitu setiap kali dilakukan penghitungan usaha minimum untuk mengalikan sebuah bagian rangkaian matriks, hasilnya akan disimpan dalam suatu tempat yang telah disediakan sebelumnya. Sehingga jika kelak ada perhitungan lain yang membutuhkan hasil perhitungan tersebut, program tidak perlu melakukan perhitungan kembali. Pogram cukup mengambil dari data yang telah disimpan sebelumnya.

Dalam penyelesaian permasalahan untuk panjang rantai perkalian matriks n, terdapat $n^2/2$ kemungkinan pembagian rantai perkalian yang berbeda. Kompleksitas algoritma yang dijalankan oleh program juga turun cukup banyak dari $O(2^n)$ menjadi $O(n^3)$ sehingga menjadi cukup efisien untuk diimplementasikan.

Pemecahan masalah perkalian matriks berantai menggunakan program dinamis membutuhkan sebuah tabel M berukuran n x n, dimana n merepresentasikan panjangnya rantai perkalian matriks. Setiap sel M_{ij} akan diisi dengan usaha yang diperlukan untuk melakukan operasi perkalian matriks ke-i sampai dengan matriks ke-j. Jelaslah bahwa untuk setiap i = j, M_{ij} = 0. Dan solusi optimum yang kita cari terletak di salah satu sel lainnya. Berikut ini adalah contoh penggambaran pengisian tabel oleh pogram.



Gambar 1. Visualisasi tabel penyimpanan tabel solusi.

Proses pengisian tabel membutuhkan waktu selama O(n) dari total kompleksitas $O(n^3)$. Jadi kompleksitas memori dari algoritma ini adalah $O(n^2)$

Setiap cara mengalikan serangkaian matriks bisa direpresentasikan dalam sebuah pohon biner (*binary tree*) dimana matriks-matriks yang ada adalah daun dan cabang-cabang di bagian dalam adalah hasil antara.

Misalkan T adalah pohon yang mewakili cara optimum untuk melakukan perkalian pada serangkaian matriks X_i, …, X_j. Cabang kiri pohon (L) mewakili perkalian matriks-matriks X_i, …, X_k dan cabang kanan (R) mewakili perkalian matriks-matriks X_{k+1}, …, X_j, dengan k adalah sembarang bilangan diantara i dan j.

Usaha total untuk pohon T (usaha(T))adalah:

$$\text{Usaha}(T) = \text{Usaha}(L) + \text{Usaha}(R) + \text{Usaha}(L*R)$$

Untuk memastikan bahwa T adalah pohon yang optimum sebagai solusi perkalian matriks berantai tersebut, perlu dilakukan pembuktian bahwa upapohon kiri adalah pohon yang optimum untuk X_i, …, X_k dan upapohon kanan adalah pohon yang optimum untuk X_{k+1}, …, X_j.

Cara pembuktiannya adalah secara kontradiksi. Jika pohon tersebut bukanlah pohon solusi yang optimum, maka pasti ada pohon yang mempunyai usaha lebih baik daripadanya untuk rangkaian matriks yang sama

$$\text{Usaha}(L') < \text{Usaha}(L)$$

atau

$$\text{Usaha}(R') < \text{Usaha}(R)$$

Kemudian buatlah sebuah pohon T' dimana pohon ini mempunyai cabang kiri L' dan cabang kanan R kemudian hitung usahanya dan bandingkan dengan T.

$$\text{Usaha}(T') = \text{Usaha}(L') + \text{Usaha}(R) + \text{Usaha}(L'*R)$$

Jika Usaha(T') lebih kecil dari Usaha (T) maka ini merupakan sebuah kontradiksi dimana sebelumnya dinyatakan bahwa T adalah pohon solusi tebaik untuk deretan matriks yang diberikan. Maka upapohon L haruslah iganti dengan upapohon L'.

Berikut ini adalah contoh algoritma pencarian solusi perkalian matriks berantai dengan menggunakan program dinamis:

```
Matrix-Chain-Order(int p[])
{
  n=p.length-1;
  for(i=1;i<=n;i++)
    m(i,i) = 0;

  for(l=2;l<=n;l++)
```

```
  //l is chain length
  {
    for(i=1;i<=n-l+1;i++)
    {
      j=i+l-1;
      m[i,j]=MAXINT;
      for(k=i;k<=j-1;k++)
      {
        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;
        }
      }
    }
  }
}
```

Dengan penerapan program dinamis ini pada masalah perkalian matriks berantai, solusi optimum bisa didapatkan dengan kompleksitas waktu $O(n^3)$ dan memori(n^2).

5. Kesimpulan

Program dinamis (*dynamic programming*) adalah metoda pemecahan masalah yang cukup mangkus dan mampu memberikan solusi optimum untuk berbagai permasalahan. Cara kerja program dinamis adalah membagi sebuah permasalahan menjadi beberapa langkah yang lebih sederhana dan saling berhubungan satu sama lain. Kemudian satu persatu langkah tesebut diselesaikan sampai akhirnya pada langkah terakhir didapatkan solusi optimum dari permasalahan tersebut.

Permasalahan perkalian matriks berantai (*chain matrix multiplication*) adalah persoalan mencari urutan langkah terbaik untuk menyelesaikan perkalian serangkaian matriks sehingga proses perkalian matriks-matriks tersebut memerlukan usaha seminimum mungkin.

Solusi untuk permasalahan perkalian matriks berantai ini dapat dicari dengan menggunakan algoritma *brute force* yaitu dengan mengenumerasi seluruh kemungkinan yang ada. Akan tetapi cara ini sangat tidak praktis untuk diimplementasikan karena membutuhkan usaha yang sangat besar. Hal ini terlihat dari kompleksitasnya yang sangat tinggi, yaitu $O(2^n)$

Selain dengan algoritma *brute force*, permasalahan perkalian matriks berantai ini juga dapat diselesaikan dengan algoritma *greedy*, akan tetapi algoritma ini tidak selalu memberikan solusi yang optimum.

Dengan menggunakan program dinamis untuk mendapatkan solusi permasalahan perkalian matriks berantai ini, kita bisa mendapatkan solusi yang optimum dengan kompleksitas algoritma yang jauh lebih kecil yaitu sebesar $O(n^3)$

Daftar Pustaka

[1] Munir, Rinaldi. 2005. Strategi Algoitmik. Teknik Informatika ITB : Bandung

[2] http://www.seas.gwu.edu. Diakses tanggal 15 Mei 2006 pukul 16.30 BBWI

[3] http://www.wikipedia.org Diakses tanggal 15 Mei 2006 pukul 16.30 BBWI

[4] Benli, Omer S. 1999. Dynamic Programming. http://www.ie.bilkent.edu.tr Diakses tanggal 15 Mei 2006 pukul 16.30 BBWI