Penerapan Algoritma Program Dinamis pada Penyelesaian Persoalan Partisi

Diah Eka Yulianti[1], Mira Yunarti[2], Robbi Kurniawan[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail: if14003@students.if.itb.ac.id[1], if14009@students.if.itb.ac.id[2], if14015@students.if.itb.ac.id[3]

Abstrak

Persoalan partisi merupakan persoalan yang sering diterapkan dalam kehidupan sehari-hari seperti misalnya pada persoalan pembagian pekerjaan. Persoalan partisi sendiri memiliki banyak model dan karateristik. Persoalan yang dibahas dalam makalah ini adalah persoalan membagi pekerjaan untuk dikerjakan oleh *N* pekerja secara efisien sedemikian sehingga setiap pekerja mendapat pekerjaan yang relatif sama(perbedaan bobot yang diterima pekerja dibuat seminimum mungkin). Penyelesaian persoalan ini salah satunya adalah dengan menggunakan algoritma program dinamis (*dynamic programming*). Algoritma program dinamis adalah penyelesaian persoalan dengan cara menguraikan solusi menjadi sekumpulan langkah dan memandang solusi dari persoalan tersebut sebagai serangkaian keputusan yang saling berkaitan.

***Kata kunci** : program dinamis , partisi*

1. Pendahuluan

Suatu pekerjaan berskala besar seringkali harus dikerjakan oleh banyak pekerja. Pekerjaan berskala besar itu biasanya dapat dibagi-bagi menjadi beberapa bagian pekerjaan yang masing-masing bagian memiliki bobot tertentu. Pembagian bobot pekerjaan selayaknya relatif sama (perbedaaan bobot yang diterima setiap pekerja dibuat seminimum mungkin) untuk setiap pekerja. Pembagian pekerjaan dengan memperhitungkan bobot tersebut dapat mengefisiensikan waktu pengerjaan dan mendapatkan hasil yang optimal.

Misalnya diberikan pekerjaan untuk melakukan pencarian informasi yang ada pada beberapa buku yang memiliki jumlah halaman berbeda kepada tiga pekerja. Buku-buku ini dibagi ke setiap pekerja dengan jumlah halaman buku relatif sama (perbedaan halaman yang diterima setiap pekerja dibuat seminimum mungkin). Untuk membagi jumlah halaman tersebut dengan perbedaan halaman seminimal mungkin diperlukan sebuah metode pembagian partisi. Proses pembagian partisi ini dapat menerapkan berbagai algoritma salah satunya adalah program dinamis.

2. Persoalan Partisi (*Partitional Problem*)

2.1. Pengertian Persoalan Partisi

Persoalan partisi adalah suatu persoalan *NP-complete* di dalam ilmu komputer [1]. *NP-complete* sendiri adalah *non-deterministic polynomial time*, sebuah masalah yang tidak bisa diselesaikan dalam waktu yang polinomial (kompleksitasnya tidak polinomial) Gambaran dari persoalan ini adalah diberikan sekumpulan integer kemudian temukan cara untuk membagi sekumpulan (*set*) integer tersebut kedalam beberapa *subset* integer sedemikian sehingga setiap *subset* integer memiliki jumlah integer yang sama atau relatif sama.

2.2. Contoh Persoalan Partisi

Diberikan tiga orang pekerja yang mendapatkan tugas melakukan pencarian sebuah informasi dalam sekumpulan buku. Agar pekerjaan tersebut dapat dilakukan dengan efisien dan seimbang. Buku-buku tersebut dibagikan kepada ketiga pekerja dengan pembagian buku yang memiliki perbedaan bobot seminimum mungkin antar pekerja. Jika setiap buku mempunyai jumlah halaman yang sama tidak menjadi persoalan untuk membagi banyaknya buku tersebut sama banyak kepada para pekerja. Misalnya setiap buku memiliki 100 halaman maka partisi 9 buku untuk 3 pekerja adalah:

100 100 100 | 100 100 100 | 100 100 100

jadi setiap pekerja mendapat bagian yang sama yaitu 300. Ketika dihadapkan pada persoalan pembagian buku dengan halaman yang berbeda-beda seperti :

100 200 300 | 400 500 600 | 700 800 900

Maka pembagian untuk tiap pekerja tidak akan berlangsung efektif. Pekerja ke-1 dengan jumlah 600 halaman, pekerja ke-2 dengan jumlah 15 dan pekerja terakhir mendapat bagian paling banyak yaitu 2400. Partisi paling efektif yang seharusnya dilakukan adalah :

100 200 300 400 500 | 600 700 | 800 900

Pekerja ke-1 mendapat bagian 1500 halaman, pekerja ke-2 mendapat bagian 1300 halaman, dan pekerja ke-3 mendapat bagian 1700 halaman. Terlihat bahwa perbedaaan bobot yang diterima oleh masing-masing pekerja bersifat minimum.

3. Algoritma Program Dinamis (*Dynamic Programming*)

3.1 Pengertian Program Dinamis

Program dinamis (*dynamic programming*) adalah suatu algoritma pemecahan persoalan dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan [2].

Inti dari program dinamis adalah membuang satu bagian kecil dari sebuah persoalan dalam setiap langkahnya, kemudian menyelesaikan persoalan yang lebih kecil tersebut dan menggunakan solusi hasil penyelesaian ini untuk ditambahkan kembali ke bagian persoalan dalam langkah berikutnya [3].

Algoritma program dinamis memiliki kemiripan dengan algoritma *greedy* dan algoritma *divide and conquer*, namun diantara ketiga algoritma tersebut terdapat perbedaan yang mendasar.

Kemiripan algoritma program dinamis dengan algoritma *greedy* terletak pada pembentukan solusi secara bertahap. Perbedaan mendasar dari kedua algoritma ini adalah bahwa pada algoritma *greedy* hanya satu rangkaian keputusan yang dihasilkan, sedangkan pada metode program dinamis lebih dari satu rangkaian keputusan.

Kemiripan algoritma program dinamis dengan *divide and conquer* adalah terletak pada pemecahan persoalan menjadi bagian persoalan yang lebih kecil. Perbedaan mendasar dari kedua algoritma ini adalah bahwa pada algoritma *divide and conquer*, persoalan dibagi menjadi beberapa bagian, menyelesaikan setiap bagian, dan dirangkaikan untuk membentuk solusi keseluruhan persoalan, sedangkan pada algoritma program dinamis, menghilangkan satu elemen / bagian kecil persoalan, menyelesaikan persoalan kecil tersebut, dan menggunakan hasil penyelesaian persoalan kecil tersebut untuk menyelesaikan persoalan yang lebih besar.

3.2 Karateristik Persoalan Program Dinamis

Persoalan yang dapat diselesaikan dengan program dinamis adalah persoalan dengan karateristik sebagai berikut [2] :

1. Persoalan dapat dibagi menjadi beberapa tahap, yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (bisa berhingga atau tidak berhingga) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada tahap tersebut.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos pada suatu tahap meningkat secara teratur dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap yang sudah berjalan dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *k* memberikan keputusan terbaik untuk setiap untuk setiap status pada tahap *k*+1.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

4. Penerapan Algoritma Program Dinamis dalam Persoalan Partisi

Langkah-langkah pencarian solusi :

1. Misalkan suatu pekerjaan yang akan dipartisi adalah n, dan banyaknya partisi adalah k. Ide solusinya adalah membagi n buah pekerjaan ke dalam k bagian dan bobot untuk bagian ke-i (1 <= i <= n) dinyatakan dengan S_i.
2. Kita hanya dapat membagi n buah pekerjaan tersebut menjadi k buah bagian setelah sebelumnya kita membagi ke dalam k-1 bagian.
3. Pembatas antara bagian k-1 dan bagian k dapat diletakkan di antara elemen ke – i dan i + 1untuk suatu nilai i, dimana 1 <= i <= n.
4. Nilai cost yang didapat dari pembagian ini merupakan nilai yang paling besar di antara 2 kandidat. Kandidat yang pertama adalah cost

dari bagian terakhir, $\sum_{j=i+1}^{n} s_j$. Dan kandidat yang kedua adalah cost dari bagian di sebelah kiri pembatas.

5. Untuk meminimalisasi cost, kita akan membagi pekerjaan di sebelah kiri pembatas menjadi k-1 bagian serata mungkin.
6. Langkah-langkah ini akan dilakukan secara rekursif.
7. Selanjutnya kita akan mendefinisikan M[n, k] adalah minimum cost yang mungkin dri seluruh pekerjaan ke 1... n yang dibagi menjadi k bagian. Nilai M[n, k] dapat diperoleh melalui fungsi:

$$M[n,k] = \min_{i=1}^{n} \max(M[i,k-1], \sum_{j=i+1}^{n} Sj)$$

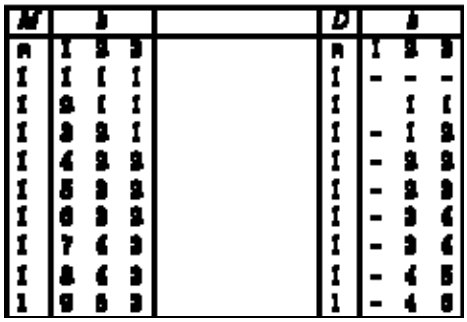
dengan basis $M[i,k] = S_1$, untuk semua k > 0 dan

$$M[n,1] = \sum_{i=1}^{n} S_i \ .$$

Pseudo code algoritma ini :

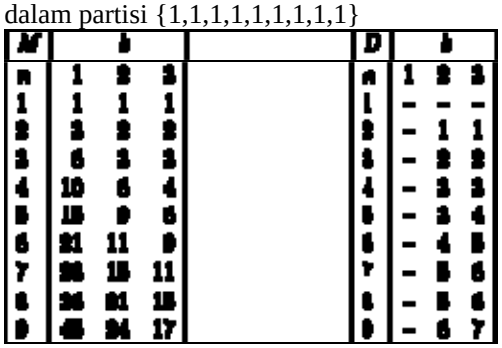
```
Partition[S,k]

{ menghitung jumlah :                    }

p[0] = 0

for i=1 to n do

for i=1 to n do M[i,1] = p[i]

for i=1 to k do

{rekurens utama}

for i=2 to n do

  for j = 2 to k do

          M[i, j]=∞

          for x = 1 to i - 1 do

             S = max(M[x, j -1],p[i]-p[x])

             if (M[i, j] > s) then

                    M[i, j] = s

                    D[i, j] = x
```

Tabel 4.1 Matriks program dinamis M dan D dalam partisi {1,2,3,4,5,6,7}



Tabel matriks M menjelaskan nilai optimum pada setiap tahapan.

Tabel 4.2 Matriks program dinamis M dan D dalam partisi {1,1,1,1,1,1,1,1,1}



Dengan mempelajari *pseudo-code* algoritma dan matriks program dinamis, dapat dipastikan bahwa hasil akhir dari M[n, k] merupakan nilai yang paling optimal dalam pemartisian. Matriks kedua, D, digunakan untuk membangun kembali partisi. Kapanpun kita meng*update* nilai dari M[i, j], kita menyimpan posisi pembagi yang dibutuhkan untuk mencapai nilai itu. Untuk merekonstruksi kembali jalan yang digunakan untuk mendapatkan nilai optimal, kita bekerja kembali dari D[n, k] dan menambahkan sebuah pembagi pada setiap posisi yang sesuai.

Pseudo code untuk rekonstruksi partisi *:*

```
ReconstructPartition(S,D,n,k)

 If (k = 1) then print the first partition
{                }

 else ReconstructPartition(S,D,D[n,k],k-1)

      Print the kth partition {
                              }
```

5. Analisis Hasil Kompleksitas

Penerapan algoritma program dinamis untuk persoalan partisi ini menghasilkan kompleksitas $O(kn^2)$, dihitung dari :

1. Proses loop ke-3 (rekurens utama) :

```
for x=1 to i-1
```

1+2+3+4+5+6+7+....+(n-1)

2. Proses loop ke-2 (rekurens utama) :

```
for j = 2 to k
```

k. (1+2+3+4+5+6+..+(n-1)) =
k. ½ (n-1) (n-2) =
½ k (n^2 – 2n – n + 2)=
½ k (n^3 – 3n +1)

Jadi kompleksitas dalam notasi O besar :
O(k). O(n^2) = O(k.n^2)

Sedangkan jika dibandingkan dengan penyelesaian menggunakan metode *exhaustive search*, *metode ini* akan melakukan komputasi M[n,k] yang diulang beberapa kali untuk nilai n dan k yang sama. Oleh karena itu penyelesaian masalah ini dengan menggunakan metode *exhaustive search* akan memakan waktu secara eksponensial (kompleksitasnya eksponensial).

6. Kesimpulan

Solusi dari masalah pemartisian pekerjaan dapat dilakukan dengan program dinamis dimana setiap kemungkinan partisi dicoba sampai ditemukan nilai yang minimum dengan cost maksimum. Kompleksitas dari algoritma ini adalah O(kn^2), yang ternyata cukup efektif dalam masalah partisi pekerjaan.

Daftar Pustaka

1. Persoalan Partisi. **http://en.wikipedia.org/computerscience/partitionproblem**
2. S. Skiena, *The Algorithm Design Manual*, 53-59, The Electronic Library of Science, New York, 1997.
3. R. Munir, *Diktat Kuliah Strategi Algoritmik*, Bandung, 2005.