PROKASDADIK, Volume 1, Desember (2023) E-ISSN: 3063-5845

Implementasi Algoritma *Dynamic Programming* untuk Menentukan *Manpower Project*

Dhaman
Teknik Informatika, Program Pascasarjana, Universitas Pamulang
e-mail: mas.dhaman@gmail.com

***Abstrak*— Project adalah suatu pekerjaan yang bersifat unik dan dilakukan dalam kurun waktu tertentu dan dengan tujuan tertentu. Project bisa dikatakan sudah selesai jika sudah mencapai tujuan yang sudah direncanakan dan membuahkan hasil dan manfaat yang diinginkan. Selain itu project juga membutuhkan manpower (orang yang bekerja dalam suatu project tertentu) untuk mencapai tujuan tersebut. Dalam bisnis dan ilmu pengetahuan biasanya didefinisikan sebagai sebuah usaha kolaboratif dan juga seringkali melibatkan penelitian atau desain, yang direncanakan untuk mencapai tujuan tertentu Proyek dapat juga didefinisikan sebagai usaha sementara, temporer, dan bukan permanen, yang memiliki sasaran kegiatan khusus dengan waktu pelaksanaan yang tegas. Contoh proyek yang terkenal antara lain adalah Proyek Infrastruktur, Proyek Pengadaan, Proyek pengembangan software aplikasi dan lain sebagainya.**

Kata Kunci: Algoritma; Dynamic; Manpower; Programming; Project.

I. PENDAHULUAN

Dynamic Programming adalah sebuah teknik dalam ilmu komputer dan matematika yang digunakan untuk menyelesaikan masalah yang dapat dibagi menjadi submasalah yang lebih kecil dan kemudian menggabungkan solusi dari submasalah-submasalah tersebut. Teknik ini terutama digunakan dalam konteks pemecahan masalah optimasi, di mana tujuannya adalah untuk mencari solusi terbaik dari sejumlah alternatif. Konsep utama dalam *dynamic programming* adalah memoisasi (penyimpanan ulang) atau menyimpan hasil perhitungan submasalah sehingga tidak perlu menghitung ulang jika masalah yang sama muncul lagi. Ini dapat menghemat waktu komputasi yang signifikan. Dynamic programming digunakan ketika ada tiga karakteristik utama dalam masalah yang ingin diselesaikan:

1) *Optimal Substructure*: Ini berarti bahwa masalah yang lebih besar dapat dipecahkan menjadi submasalah yang lebih kecil dengan solusi optimal. Solusi dari submasalah-submasalah ini kemudian digunakan untuk menyelesaikan masalah yang lebih besar.
2) *Overlapping Subproblems*: Ini berarti bahwa ada tumpang tindih antara submasalah-submasalah yang muncul dalam proses pemecahan masalah. Dalam hal ini, dynamic programming dapat digunakan untuk menghindari perhitungan berulang.
3) *Memoization*: Memiliki kemampuan untuk menyimpan hasil dari submasalah yang sudah dihitung untuk digunakan kembali jika submasalah yang sama muncul lagi.

Dynamic programming sering digunakan dalam berbagai aplikasi, seperti algoritma pencarian jalur terpendek dalam graf, pemecahan masalah pengiriman barang, penjadwalan tugas, dan sebagainya. Salah satu contoh terkenal dari dynamic programming adalah algoritma pencarian jalur terpendek, seperti Algoritma Bellman-Ford dan Algoritma Dijkstra. Teknik dynamic programming sangat berguna untuk mengoptimalkan solusi dalam masalah yang dapat dibagi menjadi submasalah yang lebih kecil dan memiliki struktur matematika yang mendukung pendekatan ini.

Contoh lain permasalah yang menggunakan algoritma dynamic programming in adalah Knapsack Problem (Masalah Knapsack). Masalah ini melibatkan pemilihan elemen dari himpunan elemen dengan bobot dan nilai tertentu, sedemikian rupa sehingga bobot total yang dapat dibawa oleh knapsack (tas ransel) tidak melebihi kapasitas maksimalnya, dan nilai dari elemen-elemen yang dipilih maksimal. Ada dua versi umum dari masalah knapsack:

1) *0/1 Knapsack Problem*: Dalam versi ini, Anda hanya boleh memilih atau tidak memilih setiap elemen (barang) yang tersedia. Dengan kata lain, Anda tidak dapat membagi sebuah elemen. Setiap elemen memiliki dua atribut: nilai (value) dan bobot (weight). Tujuannya adalah untuk memilih kombinasi elemen yang memberikan nilai maksimum tanpa melebihi kapasitas knapsack.
2) *Knapsack Problem dengan Pembagian:* Dalam versi ini, Anda diizinkan untuk membagi elemen-elemen yang tersedia menjadi bagian-bagian yang lebih kecil. Setiap elemen memiliki atribut yang sama, yaitu nilai dan bobot. Tujuannya tetap sama, yaitu memaksimalkan nilai dalam batasan bobot knapsack, tetapi Anda memiliki fleksibilitas untuk membagi elemen-elemen tersebut.

Solusi untuk masalah knapsack menggunakan dynamic programming melibatkan pembentukan tabel (array) yang disebut "tabel knapsack" atau "tabel DP" yang mengandung informasi tentang nilai maksimum yang dapat dicapai pada setiap langkah. Langkah-langkah umum dalam pemecahan masalah knapsack dengan dynamic programming adalah sebagai berikut:

1) Membuat tabel dengan dua dimensi, dengan sumbu x mewakili kapasitas knapsack (dari 0 hingga kapasitas maksimal) dan sumbu y mewakili elemen-elemen yang tersedia (dari 0 hingga jumlah elemen).
2) Menginisialisasi baris pertama dan kolom pertama tabel dengan nilai nol karena jika kapasitas knapsack adalah nol atau tidak ada elemen yang tersedia, maka nilai maksimum yang dapat dicapai adalah nol.
3) Isi tabel dengan nilai maksimum yang dapat dicapai untuk setiap kombinasi kapasitas knapsack dan elemen yang tersedia dengan mempertimbangkan apakah elemen tersebut dipilih atau tidak. Ini melibatkan penggunaan perbandingan dan memoisasi (penyimpanan ulang) nilai yang telah dihitung sebelumnya.
4) Setelah tabel selesai diisi, nilai maksimum yang dapat dicapai akan terdapat di sel terakhir dari tabel (di baris kapasitas maksimal dan kolom elemen terakhir).
5) Untuk menentukan elemen-elemen yang dipilih, Anda dapat melakukan backtrack dari sel terakhir tabel untuk melihat elemen mana yang dipilih.

Masalah knapsack adalah contoh yang sering digunakan untuk mengilustrasikan konsep dynamic programming dalam pemrograman dinamis. Dynamic programming memungkinkan Anda untuk menemukan solusi optimal dengan efisien dan menghindari menghitung ulang submasalah yang sama.

II. HASIL DAN PEMBAHASAN KASUS

Kita akan mencoba melakukan perhitungan dengan pendekatan knapsack problem pada permasalahan penentuan manpower project dengan algoritma dynamic programming. Ada 3 project yang akan dikerjakan dan masing-masing project membutuhkan manpower sebagai berikut:

- Project 1 membutuhkan manpower 4 dengan nilai project 200 juta rupiah,
- Project 2 membutuhkan manpower 7 dengan nilai project 300 juta rupiah dan
- Project 3 membutuhkan manpower 10 dengan nilai project 550 juta rupiah.

A. *Jumlah manpower yang tersedia adalah 15 orang*

Sebelum melakukan perhitungan, akan dijabarkan rumus-rumus dan informasi yang berkaitan dengan knapsack problem ini.

$$Maksimasi\ F = \sum_{i=1}^{n} p_i x_i$$

$$Dengan\ kendala\ constrain = \sum_{i=1}^{n} w_i x_i \leq M$$

Dimana:
M = kapasitas maksimal
x_i = 0 atau 1
i = 1, 2, 3, …., n

Mari kita lihat pada data yang ada dalam kasus pencarian manpower pada suatu project.

n = 3
M = 15 orang

Project ke -i	w_i	p_i
1	4	200
2	7	300
3	6	550

w = jumlah manpower
p = nilai project

Note: perkiraan waktu pengerjaan sama

Pada persoalan ini :

1) Tahap (k) adalah tahap memasukkan jumlah manpower ke dalam perkiraan tempat (knapsack)
2) Status (y) menyatakan kapasitas muat tempat yang tersisa setelah memasukkan object pada tahap sebelumnya

Dari tahap pertama, kita masukkan object pertama kedalam knapsack untuk setiap satuan kapasitas knapsack sampai batas kapasitas maksimal. Ketika memasukkan object pada tahap k, kapasitas muat knapsack sekarang adalah y - w_k. Untuk mengisi kapasitas sisanya, kita menerapkan prinsip optimalitas dengan mengacu pada nilai optimum dari tahap sebelumnya untuk kapasitas y - w_k. Nilai optimum dari tahap sebelumnya adalah $f_{k-1}(y - w_k)$. Selanjutnya akan dibanding nilai keuntungan yang didapat yaitu:

$$p_k + \text{nilai } f_{k-1}(y - w_k) \text{ dengan } f_{k-1}(y).$$

- Jika p_k + nilai $f_{k-1}(y - w_k) < f_{k-1}(y)$ maka object yang ke-k tidak dimasukkan.
- Jika p_k + nilai $f_{k-1}(y - w_k) \geq f_{k-1}(y)$ maka object yang ke-k dimasukkan.

Relasi rekuren untuk ini adalah:
$f_0(y) = 0$ untuk y = 0, 1, 2, … , M
$f_k(y) = \infty$ untuk y = 0
$f_k(y)$ = max$\{f_{k-1}(y), p_k + f_{k-1}(y - w_k)\}$ untuk k = 1, 2, …, n

Keterangan:
$f_0(y) = 0$ adalah nilai persoalan knapsack kosong
$f_k(y) = \infty$ adalah nilai persoalan knapsack untuk kapasitas negatif
$f_k(y)$ adalah keuntungan maksimum pada tahap k untuk kapasitas knapsack sebesar y

Dengan menggunakan rumus dan persamaan diatas mari kita hitung kasus menentukan manpower yang akan dicari untuk project tertentu supaya mendapatkan keuntungan yang maksimal.

M = 15 orang x = {x1, x2, x3}
n = 3

Barang ke-i	wi	pi
1	4	200
2	7	300
3	6	550

Pada tahap pertama kita akan mencoba memasukkan project pertama dengan kapasitas 4 orang dan keuntungan 200 juta rupiah. Bisa dilihat dalam tabel berikut ini:

tahap 1	Manpower 4 , nilai project 200			
y		p1 + f0 (y-w1)	Solusi Optimum	
	f0(y)	200 + f0(y-4)	f1(y) | p terbesar	(x1, x2, x3)
0	0	0	0	{0, 0, 0}
1	0	0	0	{0, 0, 0}
2	0	0	0	{0, 0, 0}
3	0	0	0	{0, 0, 0}
4	0	200 + 0	200	{1, 0, 0}
5	0	200 + 0	200	{1, 0, 0}
6	0	200 + 0	200	{1, 0, 0}
7	0	200 + 0	200	{1, 0, 0}
8	0	200 + 0	200	{1, 0, 0}
9	0	200 + 0	200	{1, 0, 0}
10	0	200 + 0	200	{1, 0, 0}
11	0	200 + 0	200	{1, 0, 0}
12	0	200 + 0	200	{1, 0, 0}
13	0	200 + 0	200	{1, 0, 0}
14	0	200 + 0	200	{1, 0, 0}
15	0	200 + 0	200	{1, 0, 0}

Keterangan tabel: Jika kita masukkan hanya 4 orang maka nilai maksimal yg didapat hanya senilai 1 project tersebut yaitu 200 juta rupiah. Pada tahap berikutnya kita akan masukkan lagi jumlah manpower pada project kedua.

Tahap kedua, kita coba memasukkan lagi kapasitas project 2 yaitu 7 orang kedalam knapsack yang sebelumnya sudah terisi 4 orang, detailnya bisa dilihat pada tabel dibawah ini:

tahap 2	Manpower 7, nilai project 300			
y		p2 + f1 (y-w2)	Solusi Optimum	
	f1(y)	300 + f1(y-7)	f2(y) | p terbesar	(x1, x2, x3)
0	0	0	0	{0, 0, 0}
1	0	0	0	{0, 0, 0}
2	0	0	0	{0, 0, 0}
3	0	0	0	{0, 0, 0}
4	200	0	0	{0, 0, 0}
5	200	0	0	{0, 0, 0}
6	200	0	0	{0, 0, 0}
7	200	300 + 0	300	{0, 1, 0}
8	200	300 + 0	300	{0, 1, 0}
9	200	300 + 0	300	{0, 1, 0}
10	200	300 + 0	300	{0, 1, 0}
11	200	300 + 200	500	{1, 1, 0}
12	200	300 + 200	500	{1, 1, 0}
13	200	300 + 200	500	{1, 1, 0}
14	200	300 + 200	500	{1, 1, 0}
15	200	300 + 200	500	{1, 1, 0}

Keterangan tabel: Pada tahap ini kita lihat pada y ke 11 baru terisi 2 project dengan nilai 500 juta rupiah dengan manpower maksimal 11.

Pada tahap ketiga ini, kita akan masukkan lagi project ke 3 dengan kapasitas manpower 6 orang, detail bisa dilihat pada tabel.

tahap 3	Manpower 10, nilai keuntungan 550			
y		p3 + f2 (y-w3)	Solusi Optimum	
	f2(y)	**500** + f2(y-6)	f3(y) | p terbesar	(x1, x2, x3)
0	0	0	0	{0, 0, 0}
1	0	0	0	{0, 0, 0}
2	0	0	0	{0, 0, 0}
3	0	0	0	{0, 0, 0}
4	0	0	0	{0, 0, 0}
5	0	0	0	{0, 0, 0}
6	0	550 + 0	550	{0, 0, 1}
7	300	550 + 0	550	{0, 0, 1}
8	300	550 + 0	550	{0, 0, 1}
9	300	550 + 0	550	{0, 0, 1}
10	300	550 + 0	550	{0, 0, 1}
11	500	550 + 0	550	{0, 0, 1}
12	500	550 + 0	550	{0, 0, 1}
13	500	550 + 300	850	{0, 1, 1}
14	500	550 + 300	850	{0, 1, 1}
15	500	550 + 300	850	{0, 1, 1}

Keterangan tabel: Pada y ke 13 sampai dengan maksimal y ke 15 sudah ada perubahan keuntungan yaitu 850 juta rupiah dan ini merupakan nilai keuntungan maksimum yg didapat.

III. KESIMPULAN

Dalam algoritma dynamic programming, variabel -variabel maksimum yang sudah didapatkan pada tahap sebelumnya akan disimpan untuk kemudian digunakan kembali untuk perbandingan nilai maksimum yang didapatkan pada tahap berikutnya. Jadi dalam kasus penentuan manpower project ini didapatkan bahwa, dalam kapasitas maksimum manpower yang tersedia yaitu 15 orang hanya akan digunakan 13 orang saja pada project 2 dan project 3 dengan keuntungan nilai project yang didapatkan adalah 850 juta rupiah.

DAFTAR PUSTAKA

[1] Martello, S & P. Toth. 1990. Knapsack Problem. John Wiley & Sons, Singapore
[2] Passa, F. 2009. Permasalahan Optimasi 0-1 Knapsack dan Perbandingan Beberapa Algoritma Pemecahannya. Makalah IF2091 Struktur Diskrit. Bandung: Institut Teknologi Bandung
[3] Horowitz. E, Sahni. S & Rajasekaran. S. 1988. Computer Algorithms. New York: Computer Science Press
[4] Bertsimas, D., & Demir, R, An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problems, 2002.