Penerapan Algorima Program Dinamis untuk Memaksimalkan Keuntungan Rumah Makan Padang

Muhammad Alif Arifin/13516078
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13516078@std.stei.itb.ac.id

***Abstract*—Pemilihan masakan yang ingin dibuat untuk memenuhi permintaan pelanggan di Rumah Makan Padang diperlukan banyak pertimbangan, seperti lama waktu memasak dan harga bahan mentah yang dibutuhkan. Di dalam dunia bisnis, yang terpenting adalah mendapatkan keuntungan semaksimum mungkin dengan modal seminimum mungkin. Namun, cukup susah untuk mempertimbangkan seluruh aspek yang ada dengan cepat apalagi modal yang didapat kebanyakan berasal dari penghasilan hari sebelumnya. Oleh karena itu, dibutuhkan suatu cara untuk mendapatkan keuntungan maksimal dengan batasan waktu dan modal. Salah satu penyelesaian masalah ini adalah dengan melakukan pemodelan masalah sebagai *Integer Knapsack Problem* yang dapat diselesaikan dengan program dinamis. Hasil yang didapatkan dapat dipastikan optimal.**

Keywords—Program dinamis; knapsack problem; optimasi; keuntungan maksimum; Rumah Makan Padang

I. PENDAHULUAN

Rumah Makan (RM) Padang adalah suatu bisnis di bidang kuliner yang menjual atau menghidankan berbagai macam kuliner atau masakan Minangkabau yang berasal dari Sumatera Barat [3].

Masakan Padang terkenal enak dan disukai oleh berbagai kalangan serta bermacam etnis dan bangsa karena masakan yang lezat serta daya adaptasinya yang bisa menyesuaikan diri dengan lidah atau selera masyarakat. Sehingga tidak heran apabila rumah makan ini sangat terkenal di Indonesia bahkan dunia [3]. Rumah Makan Padang di luar Sumatera Barat menghidangkan masakan yang tidak terlalu pedas, berbeda dengan rumah makan yang ada di tanah kelahirannya sendiri, yaitu Sumatera Barat.

Usaha rumah makan ini hadir dalam berbagai tingkatan sosial, mulai dari warung Padang kaki lima yang harganya terhangkau oleh kalangan bawah, rumah makan yang menargetkan kalangan menengah sebagai sasaran pasarnya, hingga restoran mewah yang menargetkan kalangan atas dengan harga yang cukup tinggi [3].

Masakan atau kuliner Minangkabau sangat beragam. Hal tersebut dikarenakan setiap wilayah di Sumatera Barat memiliki varian masakannya sendiri-sendiri walaupun memiliki banyak kesamaan. Di sekitar danau Maninjau, Agam, terdapat Palai Rinyuak yang tidak ditemukan di wilayah lain karena berbahan dasar ikan Rinyuak yang hanya terdapat di Danau Maninjau. Begitu pula dengan Salai Ikan Bilih dari wilayah Solok dan Tanah Datar. Ikan tersebut hanya terdapat di danau Singkarak, tidak terdapat di perairan lain yang ada di dunia [5]. Dari wilayah Agam lainnya terdapat suatu jenis makanan yang biasa dikenal dengan nasi Kapau, dan sebagainya.

Fig. 1. Salah Satu Contoh Masakan di Rumah Makan Padang
Sumber: https://carainvestasibisnis.com/7-cara-memulai-bisnis-rumah-makan-padang-agar-ramai/

Banyaknya masakan Minangkabau tentunya tidak memungkinkan apabila menyediakan seluruh masakan pada Rumah Makan Padang terkhusus pada Rumah Makan Padang yang tidak terlalu besar. Selain itu agar bisnis dapat tetap berjalan, tentunya para pemilik usaha kuliner Rumah Padang harus menghitung pengeluaran dan pemasukan mereka dengan hati-hati. Mereka juga harus mencari strategi yang tepat untuk menghasilkan keuntungan yang terbanyak.

Terdapat banyak hal yang perlu dipertimbangkan ketika memilih masakan yang akan dibuat seperti waktu membuat, biaya bahan dasar dan biaya yang diperoleh dari masakan tersebut. Waktu membuat menjadi bahan pertimbangan karena kebanyakan Rumah Makan Padang memasak makanannya terlebih dahulu dan ketika buka hanya melayani pelanggannya saja. Biaya bahan dasar menjadi pertimbangan karena tentunya pemilik Rumah Makan Padang memiliki *budget* 'modal' yang

terbatas sehingga mungkin tidak dapat memasak seluruh masakan Padang.

Hal tersebut tentunya tidak mudah untuk menimbangkan segala aspek yang ada. Apalagi modal sebagian besar didapatkan pada penghasilan sebelumnya sehingga masakan yang dibuat setiap harinya mungkin berbeda. Tentunya seluruh pemilik usaha Rumah Makan Padang ingin memiliki keuntungan yang maksimal dengan menimbang hal-hal yang telah disebutkan sebelumnya.

Oleh karena itu, penulis ingin membantu mempermudah pemilik Rumah Makan Padang untuk memilih makanan yang akan dimasak esok harinya dengan menimbang modal beserta waktu yang dimiliki untuk memasak. Hal tersebut diharapkan dapat mendapatkan senarai masakan yang akan menghasilkan keuntungan terbesar. Masalah ini memiliki beberapa kemiripan dengan *Integer Knapsack.* Sehingga, dilakukan dengan pendekatan penyelesaian masalah *Integer Knapsack* untuk menemukan solusi.

II. DASAR TEORI

A. Program Dinamis

Program dinamis (*dynamic programming*) merupakan salah satu metode pemecahan masalah dengan menguraikan masalah yang rumit menjadi upa masalah yang lebih sederhana. Pemecahan upa masalah dilakukan sekali saja dan menyimpan solusinya. Sehingga, apabila terdapat upa masalah yang sama, alih-alih melakukan komputasi ulang untuk mencari solusinya, seseorang hanya perlu mencari solusi yang sebelumnya pernah dipecahkan. Teknik menyimpan solusi dari upa masalah disebut *memoization* [1].

Pemecahan masalah akan memiliki sekumpulan langkah (*step*) dan tahapan (*stage*). Karakteristik penyelesaian persoalan dengan program dinamis [2]:

- Terdapat sejumlah berhingga pilihan yang mungkin;
- Solusi pada setiap tahap akan dibangun dari hasil solusi pada tahap-tahap sebelumnya;
- Penggunaan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada program dinamis, rangkaian keputusan yang optimal dibuat menggunakan prinsip optimalitas. Prinsip optimalitas menyatakan bahwa jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Dengan prinsip optimalitas ini akan menjamin bahwa pengambilan keputusan pada suatu tahap adalah keputusan yang benar untuk tahap-tahap selanjutnya. Hal tersebut akan berdampak pada pengurangan pengenumerasian rangkaian keputusan yang tidak mengarah ke solusi optimum secara drastis [2].

Program dinamis diterapkan pada persoalan yang memiliki karakteristik sebagai berikut [2]:

1) Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.
2) Setiap tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Pada umumnya, status merupakan berbagai jenis kemungkinan masukan yang ada pada tahap tersebut. Jumlah status bisa berhingga (*finite*) atau tidak berhingga (*infinite*).

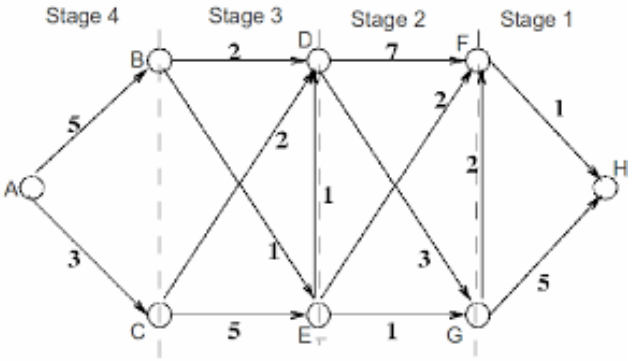

Fig. 2. *Stage* dan *State* pada Algoritma Program Dinamis dengan Pendekatan *Bottom-Up*
Sumber: https://astarmathsandphysics.com/a-level-maths-notes/d2/3384-the-dynamic-programming-labelling-procedure.html

3) Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4) Biaya (*cost*) pada stuatu tahap meningkat secara teratur dengan bertambahnya jumlah tahapan yang telah dilalui.
5) Biaya pada suatu tahap bergantung pada biaya pada tahap-tahap yang sudah berjalan dan biaya pada tahap tersebut.
6) Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7) Terdapat hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *k* memberikan keputusan terbaik untuk setiap status pada tahap $k + 1$.
8) Prinsip optimalitas berlaku pada persoalan tersebut.

Terdapat dua pendekatan dalam menyelesaikan permsalahan dengan menggunakan algoritma program dinamis, yaitu [2].

1) Program dinamis maju (*forward* atau *up-down*). Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya hungga tahap *n.* Runtutatn peubah keputusan adalah $x_1, x_2, \ldots, x_n$.
2) Program dinamis mundur (*backward* atau *bottom-up*). Program dinamis bergerak mulai dari tahap *n*, terus munder ke tahap $n - 1$, $n - 2$, dan seterusnya hingga tahap 1. Tuntutuan peubah keputusan adalah $x_n, x_{n-1}, \ldots, x_1$.

Secara umum, terdapat empat langkah yang perlu dilakukan dalam mengembangkan algoritma program dinamis [1]:

1) Karakteristikkan struktur dari solusi optimal.
2) Mendefinisikan nilai solusi optimal secara rekursif.
3) Hitung nilai solusi optimal, biasanya dengan pendekatan *bottom-up*.
4) Membangun solusi optimal dari informasi yang dihitung.

B. *Integer Knapsack Problem*

Integer Knapsack Problem adalah permasalahan dalam optimasi kombinatorial. *Knapsack* dapat diartikan sebagai karung atau tas yang akan digunakan untuk menampung objek-objek. Objek yang akan ditampung akan memiliki berat dan profit atau nilai dari objek tersebut. Objek yang ditampung harus memiliki total berat yang kurang dari sama dengan kapasitas dari *knapsack* yang dimiliki. *Integer Knapsack* memiliki sifat yaitu mengambil suatu objek menyeluruh atau tidak sama sekali. Tujuan dari penyelesaian masalah ini adalah ingin mendapatkan profit yang paling maksimum dengan batas berupa kapasitas dari *knapsack*.

Permasalahan serupa sering muncul pada permasalahan-permasalahan yang memiliki kendala keuangan dan dipelajari pada bidang-bidang seperti kombinatorika, ilmu komputer, teori kompleksitas, kriptografi, matematika terapan, dan olahraga fantasi harian.

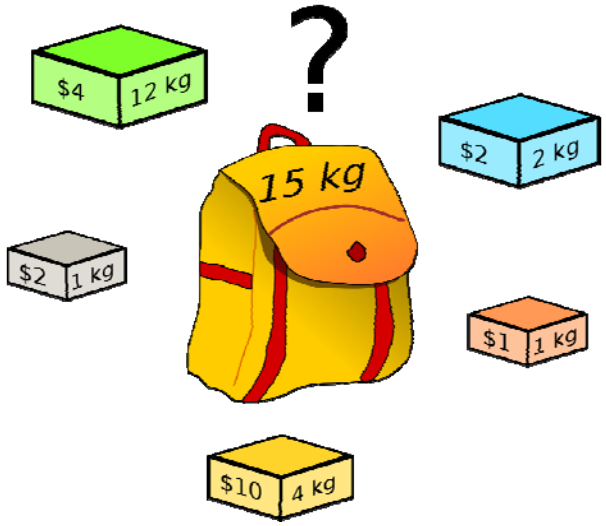

Fig. 3. Salah Satu Studi Kasus *Integer Knapsack*
Sumber:
https://en.wikipedia.org/wiki/Knapsack_problem#/media/File:Knapsack.svg

Integer Knapsack Problem dapat digunakan untuk memodelkan berbagai macam persoalan optimasi yang memiliki batasan tertentu. Salah satu permasalahan yang dapat dimodelkan dengan *Integer Knapsack Problem* adalah persoalan pemilihan menu masakan untuk mendapatkan keuntungan semaksimal mungkin berdasarkan harga bahan dan harga jual masakan.

Banyak variasi dari *Integer Knapsack Problem* untuk menyelesaikan berbagai permasalahan kompleks manusia. Hal yang dapat divariasikan adalah parameter-parameter permasalahan, yaitu jumlah batasan (*constraint*) dan jumlah *knapsack* yang digunakan. Variasi-variasi dari *Integer Knapsack Problem* antara lain sebagai berikut [4]:

- *Bounded Knapsack Problem*. Setiap objek dapat dipilih sampai maksimal *c* kali.
- *Unbounded Knapsack Problem.* Setiap objek tidak memiliki batasan pemilihan.
- *Multi-objective Knapsack Problem.* Hal atau tujuan yang ingin dioptimasi lebih dari satu.
- *Multi-dimensional Knapsack Problem.* Batasan lebih dari satu.
- *Multiple Knapsack Problem.* Jumlah *knapsack* lebih dari satu.

III. Penyelesaian *Integer Knapsack Problem* Dengan Algoritma Program Dinamis

Permasalahan yang ingin diselesaikan memiliki persamaan dengan *Multi-dimensional Knapsack Problem,* yaitu *Integer Knapsack Problem* dengan batasan lebih dari satu. Batasan yang dimiliki berupa waktu pembuatan masakan dan harga mentah bahan yang akan dimasak. Oleh karena itu, dilakukan pendekatan ke permasalahan dengan lingkup yang lebih kecil terlebih dahulu.

A. *Penyelesaian Integer Knapsack Problem*

Penyelesaian *Integer Knapsack Problem* dengan menggunakan program dinamis akan melakukan empat langkah yang telah dijabarkan sebelumnya. Pada persoalan ini akan terdapat [2]:

- Tahap (k) adalah proses memasukkan barang ke dalam *knapsack*
- Status (y) menyatakan kapasitas muat *knapsack* yang tersisa setelah memasukkan barang pada tahap sebelumnya. Kapasitas muat di sini adalah batasan dari *knapsack* tersebut.

Awalnya diperlukan untuk mengaraktersitikkan struktur dari solusi optimal. Struktur dari solusi optimalnya adalah sebagai berikut. Misalnya dipilih suatu objek pada tahap k, kapasitas *knapsack* akan menjadi $y - w_k$, dengan w_k adalah bobot dari objek yang diambil pada tahap tersebut. Karena program dinamis menganut prinsip optimalitas, maka optimalitas di sini mengacu pada nilai optimum dari tahap sebelumnya yaitu, $f_{k-1}(y - w_k)$ ditambah dengan nilai keuntungan pada tahap ke-k, p_k, yang dibandingkan dengan nilai dari tahap sebelumnya apabila tidak mengambil objek tersebut. Perbandingan dilakukan untuk mencari total profit tertinggi dan akan dijadikan sebagai keputusan pada tahap tersebut. Keputusan berupa mengambil objek pada tahap k atau tidak.

Selanjutnya mendefinisikan nilai solusi optimal secara rekursif. Relasi rekurens pada persoalan ini adalah

$f_0(y) = 0;$ $y = 0, 1, .., m$ (basis)

$f_k(y) = -\infty;$ $y < 0$ (basis)

$f_k(y) = max\ \{f_{k-1}(y), p_k + f_{k-1}(y - w_k)\};\ k = 1,\ldots, n$ (rekurens)

Untuk perhitungan nilai solusi optimal dan pembangunan solusi optimal dari informasi yang dihitung dilakukan sesuai dengan studi kasus yang ada.

B. Studi Kasus dengan Multi-dimensional Knapsack Problem

Tinjau persoalan memuatkan 3 macam barang ke dalam sebuah *knapsack*. Tiap macam barang memiliki berat w_i, volume v_i, dan akan memberikan keuntungan (apabila dijual) p_i dengan $i = 1, 2, 3$. Kapasitas muat barang *knapsack* adalah M ($M = 2$) untuk berat dan N ($N = 3$) untuk volume. Dengan tujuan adalah mendapatkan keuntungan maksimum sesuai dengan batasan yang diberikan. Dengan rincian benda yang akan dimasukkan sebagai berikut.

TABLE I. RINCIAN BENDA

No.	Barang Ke-i	W_i	V_i	P_i
1.	1	1	2	60
2.	2	1	1	45
3.	3	2	3	80

Jumlah tahapan sesuai dengan jumlah benda yang ada. Karena pada kasus ini memiliki benda sebanyak 3, jumlah tahapan pada studi kasus ini adalah 3 juga. Berikut merupakan perhitungan pada setiap tahapannya:

1) Tahap 1:

$f_1(y_1, y_2) = max\ \{f_0(y_1, y_2), p_1 + f_0(y_1 - w_1, y_2 - v_1)\}$

$= max\ \{f_0(y_1, y_2), 60 + f_0(y_1 - 1, y_2 - 2)\}$

TABLE II. TAHAP 1 PENYELESAIAN *KNAPSACK PROBLEM*

y_1	y_2	$f_0(y_1, y_2)$	$60 + f_0(y_1 - 1, y_2 - 2)$	Solusi Optimum	
				$f_1(y_1, y_2)$	(x_1^*, x_2^*, x_3^*)
0	0	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
0	1	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
0	2	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
0	3	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
1	0	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
1	1	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
1	2	0	60 + 0 = 60	60	(1, 0, 0)
1	3	0	60 + 0 = 60	60	(1, 0, 0)
2	0	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
2	1	0	$60 + (-\infty) = -\infty$	0	(0, 0, 0)
2	2	0	60 + 0 = 60	60	(1, 0, 0)
2	3	0	60 + 0 = 60	60	(1, 0, 0)

2) Tahap 2:

$f_2(y_1, y_2) = max\ \{f_1(y_1, y_2), p_2 + f_1(y_1 - w_2, y_2 - v_2)\}$

$= max\ \{f_1(y_1, y_2), 45 + f_1(y_1 - 1, y_2 - 1)\}$

TABLE III. TAHAP 2 PENYELESAIAN *KNAPSACK PROBLEM*

y_1	y_2	$f_1(y_1, y_2)$	$45 + f_1(y_1 - 1, y_2 - 1)$	Solusi Optimum	
				$f_2(y_1, y_2)$	(x_1^*, x_2^*, x_3^*)
0	0	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
0	1	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
0	2	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
0	3	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
1	0	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
1	1	0	45 + 0 = 45	45	(0, 1, 0)
1	2	60	45 + 0 = 45	60	(1, 0, 0)
1	3	60	45 + 0 = 45	60	(1, 0, 0)
2	0	0	$45 + (-\infty) = -\infty$	0	(0, 0, 0)
2	1	0	45 + 0 = 45	45	(0, 1, 0)
2	2	60	45 + 0 = 45	60	(1, 0, 0)
2	3	60	45 + 60 = 105	105	(1, 1, 0)

3) Tahap 3:

$f_3(y_1, y_2) = max\ \{f_2(y_1, y_2), p_2 + f_2(y_1 - w_3, y_2 - v_3)\}$

$= max\ \{f_2(y_1, y_2), 80 + f_2(y_1 - 2, y_2 - 3)\}$

TABLE IV. TAHAP 3 PENYELESAIAN *KNAPSACK PROBLEM*

y_1	y_2	$f_2(y_1, y_2)$	$80 + f_2(y_1 - 1, y_2 - 1)$	Solusi Optimum	
				$f_3(y_1, y_2)$	(x_1^*, x_2^*, x_3^*)
0	0	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
0	1	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
0	2	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
0	3	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
1	0	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
1	1	45	$80 + (-\infty) = -\infty$	45	(0, 1, 0)
1	2	60	$80 + (-\infty) = -\infty$	60	(1, 0, 0)
1	3	60	$80 + (-\infty) = -\infty$	60	(1, 0, 0)
2	0	0	$80 + (-\infty) = -\infty$	0	(0, 0, 0)
2	1	45	$80 + (-\infty) = -\infty$	45	(0, 1, 0)
2	2	60	$80 + (-\infty) = -\infty$	60	(1, 0, 0)
2	3	105	80 + 0 = 80	105	(1, 1, 0)

Solusi optimum pada persoalan yang telah diselesaikan adalah $X = (1, 1, 0)$ dengan nilai keuntungan maksimum adalah $f = 105$.

IV. IMPLEMENTASI PEMAKSIMALAN KEUNTUNGAN RUMAH MAKAN PADANG

A. Definisi Persoalan

Diberikan daftar masakan yang dapat dibuat oleh Rumah Makan Padang Z beserta rincian waktu memasak w_i (dalam menit), harga barang mentah v_i (dalam rupiah) dan keuntungan apabila masakan tersebut dijual p_i. Dengan modal pemilik

usaha pada hari tersebut adalah sebesar *A* dengan waktu yang dimiliki sebanyak *B*. kedua peubah tersebut akan diberitahu pada setiap *test case* yang dilakukan.

TABLE V. RINCIAN BENDA

No.	Nama Masakan	W_i	V_i	P_i
1.	Sate Padang	30	20.000	50.000
2.	Rendang	100	25.000	80.000
3.	Dendeng Batokok	20	10.000	35.000
4.	Ikan Balado	50	5.000	15.000
5.	Dendeng Balado	25	15.000	50.000
6.	Ikan Asam Padeh	30	20.000	25.000
7.	Ayam Pop	10	10.000	20.000
8.	Gulai Kepala Ikan	40	5.000	15.000
9.	Ikan Cuka	60	20.000	70.000
10.	Kalio Cumi	30	35.000	65.000

B. *Perancangan Solusi dengan Program Dinamis*

Untuk menyelesaikan permasalahan ini dengan menggunakan algoritma program dinamis diperlukan mengikuti 4 langkah yang telah disebutkan sebelumnya.

Awalnya diperlukan untuk mengaraktersitikkan struktur dari solusi optimal. Struktur dari solusi optimalnya adalah sebagai berikut. Misalnya dipilih masakan pada tahap *k*, batasan waktu akan menjadi $y_1 - w_k$, dengan w_k adalah waktu yang diperlukan untuk memasak masakan pada tahap *k* dan batasan modal akan menjadi $y_2 - v_k$, dengan v_k adalah biaya bahan dasar untuk memasak masakan tersebut. Karena program dinamis menganut prinsip optimalitas, maka optimalitas di sini mengacu pada nilai optimum dari tahap sebelumnya, yaitu $f_{k-1}(y_1 - w_k, y_2 - v_k)$ ditambah dengan nilai keuntungan pada tahap ke-*k*, p_k, yang dibandingkan dengan nilai dari tahap sebelumnya apabila tidak mengambil objek tersebut, $f_{k-1}(y_1, y_2)$. Perbandingan dilakukan untuk mencari total profit tertinggi dan akan dijadikan sebagai keputusan pada tahap tersebut. Keputusan berupa apakah perlu memasak masakan pada tahap *k* atau tidak.

Selanjutnya mendefinisikan nilai solusi optimal secara rekursif. Relasi rekurens pada persoalan ini adalah

$$f_0(y_1, y_2) = 0; \quad y_1 = 0, 1, .., a \wedge y_2 = 0, 1, .., b \quad \text{(basis)}$$

$$f_k(y_1, y_2) = -\infty; \quad y_1 < 0 \vee y_2 < 0 \quad \text{(basis)}$$

$$f_k(y_1, y_2) = max\{ f_{k-1}(y_1, y_2), p_k + f_{k-1}(y_1 - w_k, y_2 - v_k)\}; \quad k = 1, 2, \ldots, n \quad \text{(rekurens)}$$

Untuk perhitungan nilai solusi optimal dan pembangunan solusi optimal dari informasi yang dihitung dilakukan dengan program menimbang akan banyak sekali tabel yang dibangkitkan untuk mendapatkan solusi.

C. *Penjelasan Kode*

Implementasi diselesaikan dengan menggunakan bahasa Java JDK 8. Dengan antarmuka dengan menggunakan bahasa Indonesia. Awalnya program akan meminta masukan berupa file eksternal yang menyimpan data mengenai masakan secara rinci. Selanjutnya akan meminta berapa waktu yang dimiliki dalam menit dan modal yang dimiliki dalam rupiah.

```
Sate Padang        | 30  | 20000 | 50000
Rendang            | 100 | 25000 | 80000
Dendeng Batokok    | 20  | 10000 | 35000
Ikan Balado        | 50  | 5000  | 15000
Dendeng Balado     | 25  | 15000 | 50000
Ikan Asam Padeh    | 30  | 20000 | 25000
Ayam Pop           | 10  | 10000 | 20000
Gulai Kepala Ikan  | 40  | 5000  | 15000
Ikan Cuka          | 60  | 20000 | 70000
Kalio Cumi         | 30  | 35000 | 65000
```

Fig. 4. Format *food.txt* untuk menyimpan data masakan. Secara berurutan kolom menunjukkan nama, waktu yang dibutuhkan, biaya bahan mentah, keuntungan apabila dijual.
Sumber: Dokumen Penulis

Lalu, program akan membangkitkan dua tabel. Tabel yang pertama untuk menyimpan tahap $k - 1$ dan tabel yang kedua untuk menyimpan tahap *k*. kedua tabel tersebut identik dengan jumlah baris yang menyesuaikan batasan yang ada. kolom y_1 dan y_2 di-*generate* dan digunakan heuristik agar tidak terlalu memakan memori. Heuristik dilakukan pada kelipatan uang karena terlalu besar apabila harus menyimpan seluruh satuan dengan *increment* 1 satuan.

Setelah tabel dibangkitkan, program akan mulai mencari solusi pada setiap upa masalah yang ada (upa masalah berupa permasalahan perbaris) dan mencatat solusi tersebut ke dalam tabel, agar tidak perlu mencari ulang solusi apabila didapat permasalahan yang sama. Hal tersebut dilakukan hingga tahap ke *n*. Solusi dicari dengan pendekatan iteratif karena apabila rekursif akan memakan waktu. Relasi rekurens diubah dengan pendekatan iteratif.

Proses pemilihan solusi sama seperti yang telah dijelaskan sebelumnya, yaitu dengan prinsip optimalitas. Setelah proses pencarian solusi selesai. Program akan menampilkan masakan yang disarakan untuk dibuat, total modal yang dibutuhkan untuk membeli bahan mentah masakan tersebut, total waktu yang dibutuhkan untuk memasak dan total keuntungan kotor dari hasil penjualan masakan. Program juga menampilkan waktu eksekusi dalam *milisecond*.

D. *Penerapan Kasus Uji*

1) Kasus Uji 1: Pemilik Rumah Makan Padang memiliki waktu untuk memasak selama 50 menit dan modal untuk membeli bahan mentah sebesar Rp35.000,00.

```
D:\Kuliah\Tugas\Semester 4\Strategi Algoritma\Makalah>java Knapsack
Masukkan nama file yang menyimpan data masakan = food.txt
Masukkan waktu yang dimiliki = 50
Masukkan modal yang dimiliki = 35000

Dengan modal = Rp35.000,00
Dengan batas waktu = 50 menit
Masakan yang disarankan
- Dendeng Batokok
- Dendeng Balado
Modal yang dibutuhkan = Rp25.000,00
Waktu yang dibutuhkan = 45 menit
Dengan total profit = Rp85.000,00
Memakan waktu: 9,721592 ms
```

Fig. 5. Hasil Kasus Uji 1
Sumber: Dokumen Penulis

2) Kasus Uji 2: Pemilik Rumah Makan Padang memiliki waktu untuk memasak selama 200 menit dan modal untuk membeli bahan mentah sebesar Rp100.000,00.

```
D:\Kuliah\Tugas\Semester 4\Strategi Algoritma\Makalah>java Knapsack
Masukkan nama file yang menyimpan data masakan = food.txt
Masukkan waktu yang dimiliki = 200
Masukkan modal yang dimiliki = 100000

Dengan modal = Rp100.000,00
Dengan batas waktu = 200 menit
Masakan yang disarankan
- Sate Padang
- Dendeng Batokok
- Dendeng Balado
- Ikan Cuka
- Kalio Cumi
Modal yang dibutuhkan = Rp100.000,00
Waktu yang dibutuhkan = 165 menit
Dengan total profit = Rp270.000,00
Memakan waktu: 92,469254 ms
```

Fig. 6. Hasil Kasus Uji 2
Sumber: Dokumen Penulis

3) Kasus Uji 3: Pemilik Rumah Makan Padang memiliki waktu untuk memasak selama 100 menit dan modal untuk membeli bahan mentah sebesar Rp200.000,00.

```
D:\Kuliah\Tugas\Semester 4\Strategi Algoritma\Makalah>java Knapsack
Masukkan nama file yang menyimpan data masakan = food.txt
Masukkan waktu yang dimiliki = 100
Masukkan modal yang dimiliki = 200000

Dengan modal = Rp200.000,00
Dengan batas waktu = 100 menit
Masakan yang disarankan
- Sate Padang
- Dendeng Balado
- Ayam Pop
- Kalio Cumi
Modal yang dibutuhkan = Rp80.000,00
Waktu yang dibutuhkan = 95 menit
Dengan total profit = Rp185.000,00
Memakan waktu: 104,396284 ms
```

Fig. 7. Hasil Kasus Uji 3
Sumber: Dokumen Penulis

E. Analisis

Berdasarkan pengujian yang telah dilakukan, dapat dilihat bahwa hasil yang didapat merupakan hasil yang optimal dari semua kemungkinan solusi yang ada. Dengan menggunakan algoritma program dinamis, didapat waktu eksekusi pada rentang 9 – 100 ms. Hal tersebut tergantung pada banyaknya batasan karena semakin tinggi batasan semakin banyak baris pada tabel yang dibangkitkan sehingga akan memakan waktu komputasi.

Hal tersebut telah dicontohkan pada kasus uji. Ketika batasan modal hanya Rp.35.000,00 dan batasan waktu hanya 50 menit maka waktu komputasi yang dibutuhkan hanya 9 ms. Sedangkan ketika batasan pada kasus uji tersebut besar, yaitu batasan modal sebesar Rp.200.000,00 dan batasan waktu sebesar 100 menit maka waktu komputasi yang dibutuhkan mencapai 100 ms. Waktu komputasi akan dipengaruhi oleh jumlah masakan juga, apabila semakin banyak jumlah masakan maka akan semakin banyak tahap yang perlu dilalui.

Selain itu, meskipun didapatkan keuntungan terbesar namun belum menimbang hal-hal yang penting seperti gizi yang didapatkan. Aplikasi ini hanya terbatas untuk memenuhi kebutuhan Pemilik Rumah Makan Padang tanpa bisa mengoptimasi nutrisi agar nutrisi makanan yang didapatkan sudah 4 sehat 5 sempurna atau sesuai dengan jumlah kalori yang dibutuhkan.

Algoritma lain yang dapat digunakan untuk memecahkan masalah ini adalah *divide and conquer* dan *brute force*. Program dinamis dapat dipastikan lebih cepat dari kedua algoritma tersebut karena kalau *brute force* akan mencari seluruh kemungkinan yang ada dan apabila *divide and conquer* akan membangkitkan solusi walaupun upa masalah tersebut pernah dipecahkan sebelumnya, apalagi jika menggunakan rekursif akan memakan waktu eksekusi yang cukup lama. Namun, kekurangan pada program dinamis adalah memori yang dibutuhkan, karena setiap tahap diperlukan tabel yang tidaklah kecil sehingga cukup memakan memori.

V. KESIMPULAN

Berdasarkan hasil eksperimen dan analisis, dapat disimpulkan bahwa algoritma program dinamis dapat menyelesaikan permasalahan optimasi karena menganut prinsip optimalitas. Salah satu persoalan optimasi yang dapat diselesaikan adalah *Multi-dimensional Knapsack Problem*. Persoalan tersebut salah satunya adalah persoalan optimasi keuntungan Rumah Makan Padang dengan batasan berupa waktu membuat masakan dan modal yang dimiliki untuk membeli bahan mentah.

VI. APENDIKS

Implementasi program yang telah dibuat penulis beserta data uji coba dapat diakses pada GitHub melalui tautan berikut https://github.com/AlifArifin/Multidimensional-Knapsack.
Program ditulis dengan menggunakan bahasa Java JDK 8 sehingga apabila ingin mengeksekusi program diperlukan Java JDK 8.

UCAPAN TERIMA KASIH

Ucapan syukur penulis panjatkan kepada Tuhan yang Maha Esa karena dengan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan makalah yang berjudul "Penererapan Algoritma Program Dinamis untuk Memaksimalkan Keuntungan Rumah Makan Padang". Penulis juga mengucapkan terima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma, Dr. Ir. Rinaldi Munir, M.T., Masayu Leylia Khodra, S.T., M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M. Sc., yang telah senantiasa mendidik kami dengan penuh pengertian dan tidak lelah untuk membagi ilmu yang telah dimiliki. Terakhir, penulis juga ingin berterima kasih kepada keluarga penulis yang secara terus-menerus memberikan semangat dan dorongan pada penulis untuk menjalani studi di Institut Teknologi Bandung.

REFERENCES

[1] Cormen, Thomas H. dkkc. 2009. *Introduction to Algorithms Third Edition*. Cambridge: The MIT Press.

[2] Munir, Rinaldi. 2007. *Diktat Kuliah IF2211 Strategi Algoritma*. Bandung.

[3] Darmawan, Dadan dan Muhammad Firmansyah. 2009. *Bisnis Anti Bangkrut.* Jakarta: Visimedia.

[4] Chang, T. J., dkk. 1998. *Computers & Operations Research 27: Heuristics for Cardinality Constrained Portfolio Optimization.* England.

[5] Nugroho, Roni Aryanto. 2013. *Berkah Bilik Danau Singkarak.* Diakses pada tanggal 13 Mei 2018 dari https://travel.kompas.com/read/2013/09/02/1327508/Berkah.Bilih.Danau.Singkarak

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, 13 Mei 2018

Muhammad Alif Arifin
13516078