

Menu Diet dengan Pemrograman Dinamis

I Putu Eka Surya Aditya - 13516061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516061@std.steiitb.ac.id

Abstrak—Terdapat tiga makro nutrisi yang berguna untuk menyediakan energi bagi tubuh manusia, yaitu karbohidrat, lemak, dan protein. Jika seseorang dapat menyeimbangkan tiga makro nutrisi tersebut orang tersebut biasanya tidak akan mengalami masalah berat badan. Namun seringkali terdapat salah satu makro nutrisi yang berlebih terutama karbohidrat dan lemak yang mengakibatkan seseorang menjadi gendut bahkan obesitas. Banyak faktor yang mengakibatkan hal tersebut terjadi seperti selera terhadap makanan yang banyak karbohidrat dan lemak, kesibukan mereka ataupun karena masalah ekonomi. Oleh karena itu, dibutuhkan suatu cara untuk menentukan menu makan yang tepat bagi orang tersebut untuk mengatur asupan kalori maupun gizinya. Salah satu penyelesaian masalah ini adalah dengan memodelkan permasalahan ini sebagai knapsack problem dan menyelesaikannya dengan pemrograman dinamis. Hasil yang diperoleh dijamin optimal, namun mungkin belum bisa diaplikasikan langsung karena belum mempertimbangkan parameter-parameter tertentu dan juga keterbatasan dari program yang dibuat.

Keywords—knapsack problem; menu makan; optimasi; pemrograman dinamis

I. PENDAHULUAN

Banyak orang mengalami masalah berat badan baik kelebihan berat badan (kegemukan) maupun berat badan yang kurang. Hal ini disebabkan oleh banyak hal, namun yang paling sering dijumpai adalah kesibukan terhadap pekerjaan sehingga seseorang sering mengabaikan kebutuhan energi (kalori) maupun asupan gizi yang mereka dapatkan dari makanan yang mereka konsumsi. Padahal untuk menjaga maupun meningkatkan produktivitas dan tenaga mereka harus menjaga kebutuhan kalori dan asupan gizi dari makanan yang mereka konsumsi.

Masalah pada pola makan tidak hanya pada kurangnya asupan kalori dan gizi. Pola makan yang tidak benar juga dapat menyebabkan asupan kalori maupun gizi yang berlebih. Asupan kalori maupun gizi yang berlebih ini dapat mengakibatkan obesitas (kegemukan) dan juga dapat menimbulkan berbagai penyakit, misalnya diabetes, penyakit jantung, dan hipertensi.

Kalori dan gizi yang dibutuhkan oleh setiap orang berbeda-beda dan terdapat banyak faktor yang memengaruhi hal tersebut. Faktor-faktor tersebut diantaranya adalah usia, jenis kelamin, metabolisme, faktor genetik, dan tingkat keaktifan seseorang. Selain itu, mungkin masih terdapat faktor lain yang

memengaruhi kebutuhan kalori dan gizi seseorang. Akibat dari perbedaan kebutuhan kalori dan gizi tersebut, pola makan setiap orang pun menjadi berbeda-beda.

Berdasarkan permasalahan-permasalahan tersebut, dibutuhkan pengaturan pola makan yang benar untuk memenuhi kebutuhan kalori dan gizi. Salah satu cara untuk mengatur pola makan adalah dengan menentukan menu makan yang cocok berdasarkan tujuan yang diinginkan, misalnya dengan membatasi kalori atau gizi tertentu dan memaksimalkan kalori atau gizi tertentu yang berbeda dengan yang dibatasi. Contohnya pada orang yang menderita diabetes tetapi membutuhkan kalori yang banyak, makanan yang dikonsumsi harus dibatasi kandungan karbohidrat dan lemaknya tetapi tetap harus memaksimalkan kalori yang didapatkan dengan menambah protein. Pengaturan menu makan ini juga terkadang perlu mempertimbangkan biaya diperlukan, terutama untuk pada orang berpendapatan rendah.

Makalah ini akan mengaplikasikan pemrograman dinamis (dynamic programming) yang merupakan salah satu topik mata kuliah “Strategi Algoritma” untuk menyelesaikan permasalahan penentuan menu diet yang memiliki batasan tertentu (karbohidrat terbatas, lemak terbatas, protein terbatas atau menu makan dengan anggaran tertentu) dan memaksimalkan kalori atau gizi tertentu yang didapatkan dari menu tersebut. Masalah ini dapat dimodelkan menjadi permasalahan Integer (1/0) Knapsack yang memiliki beberapa batasan (constraint). Makalah ini tidak akan dibahas masalah penentuan menu diet yang tidak memaksimalkan faktor tertentu.

II. DASAR TEORI

A. Integer (1/0) Knapsack Problem

(1/0) Knapsack Problem adalah permasalahan memilih n buah objek untuk dimasukkan ke dalam sebuah knapsack (karung, tas, buntelan, dsb) yang memiliki kapasitas bobot K sedemikian rupa sehingga objek-objek yang dipilih tidak melebihi kapasitas knapsack namun keuntungan yang didapatkan maksimal. Setiap objek yang akan dipilih memiliki properti bobot (*weight*) w_i dan keuntungan atau nilai (*value*) v_i . Pada Integer (1/0) Knapsack Problem, setiap objek hanya dapat dipilih sekali. Secara matematis, Integer (1/0) Knapsack Problem dapat dituliskan sebagai berikut.

Maksimasi $\sum_{i=1}^n x_i v_i$
 dengan batasan (*constraint*) $\sum_{i=1}^n x_i w_i \leq K$
 dengan $x_i \in \{0, 1\}$
 Solusi: $\{x_1, x_2, \dots, x_n\}$



Gambar 1 – Ilustrasi Integer (1/0) Knapsack Problem [3]

Integer (1/0) Knapsack Problem dapat digunakan untuk memodelkan berbagai persoalan optimasi yang solusinya berupa *subset* yang memiliki batasan (*constraint*) tertentu. Contoh permasalahan yang dapat dimodelkan sebagai Integer (1/0) Knapsack Problem adalah penyimpanan barang di gudang dan pengaturan pengiriman barang seefisien mungkin untuk mendapatkan keuntungan terbesar.

Dalam teori komputasi, Integer (1/0) Knapsack Problem adalah satu persoalan yang termasuk ke dalam persoalan NP-Complete. Ini berarti belum ditemukan sebuah algoritma yang tepat dan mangkus (dapat menyelesaikan persoalan dalam waktu polinomial) untuk menyelesaikan Integer (1/0) Knapsack Problem. Walaupun belum ditemukan algoritma yang mangkus untuk menyelesaikan persoalan ini, terdapat algoritma yang berjalan dalam waktu *pseudo-polynomial* yang dapat menyelesaikan permasalahan ini, yaitu dengan menggunakan pemrograman dinamis (*dynamic programming*). Selain dengan menggunakan pemrograman dinamis (*dynamic programming*), persoalan ini juga dapat diselesaikan dengan metode-metode lain, yaitu *exhaustive search* (mengenumerasi semua kemungkinan solusi), *branch and bound*, dan *greedy* (pendekatan dengan *greedy* tidak selalu menghasilkan solusi yang optimal, tetapi dapat digunakan mengaproksimasi bobot (*cost*) yang dapat digunakan untuk pendekatan *branch and bound*).

Terdapat beberapa variasi dari Integer (1/0) Knapsack Problem yang muncul dari berbagai aplikasi dari Integer (1/0) Knapsack Problem. Variasi-variasi ini muncul dengan mengganti parameter-parameter dari permasalahan semula, misalnya jumlah tujuan, jumlah *knapsack* yang digunakan, atau jumlah batasan (*constraint*). Variasi-variasi dari Integer (1/0) Knapsack Problem diantaranya adalah *Bounded Knapsack*

Problem (setiap objek dapat dipilih sampai maksimal *c* kali), *Unbounded Knapsack Problem* (tidak ada batas pemilihan untuk setiap objek), *Multi-objective Knapsack Problem* (tujuan / hal yang ingin dioptimasi lebih dari 1), *Multi-dimensional Knapsack Problem* (batasan / *constraint* lebih dari 1), dan *Multiple Knapsack Problem* (jumlah *knapsack* lebih dari 1).

B. Pemrograman Dinamis (Dynamic Programming)

Pemrograman Dinamis (*Dynamic Programming*) adalah salah satu metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Istilah pemrograman dinamis pertama kali diperkenalkan oleh seorang professor dari Universitas Princeton yang juga bekerja di RAND corporation, bernama Richard Bellman, pada era tahun 1950-an. Istilah ini muncul karena perhitungan solusi dilakukan dengan menggunakan tabel-tabel. Pemrograman dinamis umumnya digunakan untuk menyelesaikan persoalan optimasi untuk tahapan – tahapan yang sering mengalami tumpang tindih.

Pemecahan masalah dengan menggunakan metode pemrograman dinamis memiliki kemiripan dengan metode *greedy* yang juga membentuk solusi secara bertahap. Perbedaan kedua metode ini terletak pada rangkaian keputusan yang dipertimbangkan dalam penyelesaian masalah. Pada metode *greedy*, hanya satu rangkaian keputusan saja yang dipertimbangkan, yaitu rangkaian keputusan yang terdiri dari keputusan-keputusan yang memberikan hasil paling baik untuk setiap tahapnya (*optimal lokal*), yang harapannya dapat mengarah ke *optimal global*. Pada metode *greedy* keputusan yang dipilih pada tahap tertentu tidak mempertimbangkan rangkaian keputusan untuk tahapan berikutnya maupun rangkaian keputusan yang telah dipilih pada tahap sebelumnya. Salah satu contoh permasalahan yang tidak dapat diselesaikan dengan metode *greedy* (tidak selalu benar) adalah *Integer (1/0) Knapsack Problem*.

Berbeda dengan metode *greedy*, metode pemrograman dinamis mempertimbangkan lebih dari satu rangkaian keputusan. Rangkaian-rangkaian keputusan tersebut dibuat dengan menggunakan prinsip optimalitas yang menyatakan bahwa jika solusi total optimal, maka bagian solusi sampai tahap ke-*k* juga optimal. Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap *k* ke tahap *k + 1*, kita dapat menggunakan hasil optimal dari tahap *k* tanpa harus kembali ke tahap awal. Berdasarkan prinsip optimalitas ini dapat dirumuskan bahwa ongkos pada tahap *k + 1* adalah (ongkos yang dihasilkan pada tahap *k*) + (ongkos dari tahap *k* ke tahap *k + 1*). Pemilihan keputusan pada suatu tahap pada metode pemrograman dinamis dilakukan dengan menggunakan persyaratan optimasi dan kendala untuk membatasi pilihan yang harus dipertimbangkan.

Pemrograman dinamis dapat diterapkan pada persoalan yang memiliki karakteristik sebagai berikut :

1. Terdapat sejumlah berhingga pilihan yang mungkin.

2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Status merupakan kemungkinan masukan yang ada pada tahap tersebut.
3. Solusi setiap tahap dibangun atas hasil solusi tahap sebelumnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-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 mengidentifikasi 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 yang dapat digunakan untuk menyelesaikan suatu persoalan dengan pemrograman dinamis yaitu sebagai berikut [2].

1. Pemrograman dinamis maju (*forward* atau *up-down*). Pemrograman dinamis bergerak mulai dari tahap 1, lalu ke tahap 2, 3, dan seterusnya sampai tahap ke- n . Rangkaian keputusannya adalah x_1, x_2, \dots, x_n .
2. Pemrograman dinamis mundur (*backward* atau *bottom-up*). Pemrograman dinamis bergerak mulai tahap n , lalu mundur ke tahap $n - 1, n - 2$, dan seterusnya sampai tahap ke 1. Rangkaian keputusannya adalah x_n, x_{n-1}, \dots, x_1 .

Kedua pendekatan ini ekuivalen dan sama-sama menghasilkan solusi yang optimum.

Pada umumnya untuk mengembangkan algoritma pemrograman dinamis terdapat empat langkah yaitu, sebagai berikut:

1. Karakteristikkan struktur solusi optimum.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur.
4. Konstruksi solusi optimal.

Pada implementasinya, pemrograman dinamis dapat diimplementasikan secara rekursif maupun secara iteratif dengan memanfaatkan larik (*array*) multi dimensi (tergantung banyaknya parameter yang merepresentasikan status untuk suatu tahap) untuk melakukan memorisasi. Implementasi secara iteratif akan membangun solusi dari bawah ke atas (*bottom-up*) dan menelusuri semua status yang mungkin untuk setiap tahap, sedangkan implemetasi secara rekursi akan membangun solusi dari atas ke bawah (*top-down*) membentuk sebuah pohon dan hanya akan mengunjungi status yang dibutuhkan pada setiap tahapnya untuk mencapai solusi optimal (tidak menelusuri semua status yang mungkin untuk setiap tahapnya).

C. Diet

Banyak orang yang keliru menafsirkan kata '*diet*' sebagai program penurunan berat badan dengan mengurangi porsi makan. Diet sendiri lebih mengarah kepada pengaturan jumlah makanan atau energi yang masuk ke tubuh untuk mencapai atau menjaga berat badan yang terkontrol. Diet dilakukan oleh sebagian orang dengan tujuan yang berbeda-beda. Ada sebagian orang melakukan diet untuk menjaga kesehatannya atas anjuran dokter atau karena menderita penyakit tertentu, dan ada juga yang melakukan diet karena bermasalah dengan berat badannya.

Diet yang dilakukan untuk menjaga kesehatan biasanya dilakukan oleh penderita diabetes, darah tinggi, ginjal, jantung, asam urat, dan penderita penyakit lainnya yang diharuskan melakukan diet. Diet juga biasanya dilakukan oleh ibu hamil dan ibu menyusui. Diet yang dilakukan bertujuan untuk mendapatkan berat badan ideal, mempunyai perbedaan cara dalam menurunkan berat badan bagi yang kelebihan berat badan (obesitas) dan menaikkan berat badan bagi yang kurang atau belum mencapai berat ideal. Diet juga biasa dilakukan oleh para atlet, biasanya mereka mengikuti diet untuk menjaga berat badan dengan mengurangi persentase lemak dalam tubuh mereka serta meningkatkan kepadatan otot mereka.

Diet digunakan untuk menjaga berat badan yang stabil. Bagi mereka yang mempunyai berat badan yang ideal tetapi mudah sekali mengalami kenaikan berat badan biasanya akan melakukan diet untuk menjaga berat badan mereka supaya tetap stabil. Di dalam melakukan diet, diperlukan program diet yang tepat sesuai dengan kebutuhan dan usia. Hal ini perlu karena dengan program diet yang tepat buat kita maka tujuan dari diet yang dilakukan akan tercapai. Program diet banyak macamnya dan setiap orang belum tentu akan sama hasilnya dalam menerapkan program diet yang sama. Untuk itu alangkah baiknya sebelum anda melakukan diet, konsultasikan dulu dengan dokter anda, ahli gizi, atau orang yang pakar dalam program-program diet.

Diet tidak hanya dilakukan oleh orang dewasa saja. Banyak para remaja dan anak-anak melakukan diet. Biasanya program diet mereka akan berbeda dengan program diet yang dilakukan oleh orang dewasa. Di dalam program diet biasanya akan disertakan berbagai resep diet. Selain mengatur pola makan biasanya program diet untuk menjaga berat badan terutama untuk menurunkan berat badan biasanya disertai dengan latihan fisik.

Jadi sebelum memulai program diet, ada beberapa hal yang perlu diperhatikan diantaranya:

1. Tentukan dulu tujuan anda dalam melakukan diet.
2. Tentukan rencana diet, sebaiknya konsultasi dengan dokter, ahli gizi atau para pakar dalam program diet.
3. Konsistensi dalam melaksanakan program diet.

Prinsip utama dari suatu diet adalah mengontrol jumlah energi (dalam satuan kalori), yang masuk ke dalam tubuh dan

menyesuikannya dengan *TDEE*(Total Daily Energy Expenditure) atau jumlah kalori yang digunakan oleh tubuh setiap hari. Jika seseorang ingin menambah berat badannya, maka kalori yang masuk ke dalam tubuh harus lebih besar dari *TDEE*, begitu pula jika seseorang ingin menurunkan berat badan maka jumlah kalori yang masuk ke dalam tubuh harus lebih kecil dari *TDEE*, sehingga terjadi pembakaran energi yang tersimpan di dalam tubuh berupa lemak.

III. PENYELESAIAN MASALAH MULTI-DIMENSIONAL KNAPSACK DENGAN PEMROGRAMAN DINAMIS

Penyelesaian *Multi-dimensional Knapsack Problem* dengan menggunakan pemrograman dinamis akan dilakukan dengan mengikuti empat langkah yang telah disebutkan sebelumnya. Pada persoalan ini, tahap (*k*) adalah proses memasukkan objek ke-*k* ke dalam *knapsack* dan status (*y*) menyatakan kapasitas masing-masing batasan pada *knapsack* yang tersisa setelah memasukkan objek pada tahap sebelumnya. Status (*y*) dan kapasitas *knapsack* (*M*) untuk permasalahan ini dapat dinyatakan dalam bentuk vektor multidimensi tergantung banyaknya batasan (*constraint*) yang ada. Misalkan terdapat *m* batasan (*constraint*) maka $y = (y_1, y_2, \dots, y_m)$.

Struktur solusi optimum untuk persoalan ini adalah sebagai berikut. Misalkan ketika memasukkan objek pada tahap *k*, kapasitas *knapsack* menjadi ($y - w_k$, dengan w_k adalah bobot untuk setiap batasan yang ada). Untuk mengisi kapasitas yang tersisa akan digunakan prinsip optimalitas yang mengacu pada nilai optimum dari tahap sebelumnya untuk kapasitas yang tersisa, yaitu $f_{k-1}(y - w_k)$. Setelah itu, nilai keuntungan dari pemilihan objek pada tahap ke-*k* (p_k) ditambah dengan nilai $f_{k-1}(y - w_k)$ dibandingkan dengan keuntungan tanpa memilih objek pada tahap ke-*k* ($f_{k-1}(y)$) dan dipilih keputusan yang menghasilkan keuntungan yang lebih besar (mengambil objek ke-*k* atau tidak).

Relasi rekurens untuk persoalan ini adalah sebagai berikut.

$$\begin{aligned}
 & f_0(y) = 0, y_i = 0, 1, 2, \dots, M_i \\
 & \quad \text{untuk } 1 \leq i \leq m \text{ (basis)} \\
 & f_k(y) = -\infty, \text{ jika terdapat } y_i < 0 \\
 & \quad \text{untuk } 1 \leq i \leq m \text{ (basis)} \\
 & f_k(y) = \max \{ f_{k-1}(y), p_k + f_{k-1}(y - w_k) \}, \text{ (rekurens)} \\
 & \quad k = 1, 2, \dots, n \text{ dan } y_i \geq 0 \text{ untuk } 1 \leq i \leq m
 \end{aligned}$$

$f_k(y)$ adalah keuntungan optimum dari *Multi-dimensional Knapsack Problem* pada tahap *k* untuk kapasitas sebesar *y*, $f_0(y) = 0$ adalah nilai dari nilai persoalan *knapsack* kosong (tidak ada objek yang ingin dimasukkan ke dalam *knapsack*) dengan kapasitas *y*, dan $f_k(y) = -\infty$ adalah nilai dari persoalan *knapsack* untuk kapasitas negatif (melanggar batasan / *constraint*). Solusi optimum dari dari *Multi-dimensional Knapsack Problem* adalah $f_n(M)$ dengan waktu kompleksitas algoritmanya adalah $O(nM)$.

Berikut adalah contoh penerapan pemrograman dinamis untuk menyelesaikan *Multi-dimensional Knapsack Problem*. Misalkan batasan yang digunakan adalah kapasitas bobot dan volume maksimal yang dapat ditampung oleh *knapsack* ($m = 2$) dengan bobot dan volume maksimal masing-masing adalah 3

dan $2(M_1 = 3, M_2 = 2)$ dan jumlah objek yang ingin dimasukkan ada 3 dengan rincian sebagai berikut.

TABEL I RINCIAN PERSOALAN *KNAPSACK* DENGAN 3 OBJEK

| Objek ke-i | bobot (wi) | volume (vi) | nilai (pi) |
|------------|------------|-------------|------------|
| 1 | 2 | 1 | 55 |
| 2 | 3 | 2 | 70 |
| 3 | 1 | 1 | 35 |

Berikut adalah perhitungan untuk setiap tahapnya.

1. Tahap 1:

$$\begin{aligned}
 f_1(y_1, y_2) &= \max \{ f_0(y_1, y_2), p_1 + f_0(y_1 - w_1, y_2 - v_1) \} \\
 &= \max \{ f_0(y_1, y_2), 55 + f_0(y_1 - 2, y_2 - 1) \}
 \end{aligned}$$

Tabel II Tahap 1 penyelesaian persoalan *knapsack*

| y_1 | y_2 | $f_0(y_1, y_2)$ | $55 + f_0(y_1 - 2, y_2 - 1)$ | $f_1(y_1, y_2)$ | (x_1^*, x_2^*, x_3^*) |
|-------|-------|-----------------|------------------------------|-----------------|-------------------------|
| 0 | 0 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 0 | 1 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 0 | 2 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 1 | 0 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 1 | 1 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 1 | 2 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 2 | 0 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 2 | 1 | 0 | 55 | 55 | (1, 0, 0) |
| 2 | 2 | 0 | 55 | 55 | (1, 0, 0) |
| 3 | 0 | 0 | $-\infty$ | 0 | (0, 0, 0) |
| 3 | 1 | 0 | 55 | 55 | (1, 0, 0) |
| 3 | 2 | 0 | 55 | 55 | (1, 0, 0) |

2. Tahap 2:

$$\begin{aligned}
 f_2(y_1, y_2) &= \max \{ f_1(y_1, y_2), p_2 + f_1(y_1 - w_2, y_2 - v_2) \} \\
 &= \max \{ f_1(y_1, y_2), 70 + f_1(y_1 - 3, y_2 - 2) \}
 \end{aligned}$$

Tabel III Tahap 2 penyelesaian persoalan *knapsack*

| y_1 | y_2 | $f_1(y_1, y_2)$ | $70 + f_1(y_1 - 3, y_2 - 2)$ | $f_2(y_1, y_2)$ | (x_1^*, x_2^*, x_3^*) |
|-------|-------|-----------------|------------------------------|-----------------|-------------------------|
| 0 | 0 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 0 | 1 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 0 | 2 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 1 | 0 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 1 | 1 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 1 | 2 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 2 | 0 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 2 | 1 | 55 | $70 - \infty = -\infty$ | 55 | (1, 0, 0) |
| 2 | 2 | 55 | $70 - \infty = -\infty$ | 55 | (1, 0, 0) |
| 3 | 0 | 0 | $70 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 3 | 1 | 55 | $70 - \infty = -\infty$ | 55 | (1, 0, 0) |
| 3 | 2 | 55 | $70 + 0 = 70$ | 70 | (0, 1, 0) |

3. Tahap 3:

$$f_3(y_1, y_2) = \max \{f_2(y_1, y_2), p_3 + f_2(y_1 - w_3, y_2 - v_3)\}$$

$$= \max \{f_2(y_1, y_2), 35 + f_2(y_1 - 1, y_2 - 1)\}$$

Tabel IV Tahap 3 penyelesaian persoalan knapsack

| y_1 | y_2 | $f_2(y_1, y_2)$ | $35 + f_2(y_1 - 1, y_2 - 1)$ | $f_2(y_1, y_2)$ | (x_1^*, x_2^*, x_3^*) |
|-------|-------|-----------------|------------------------------|-----------------|-------------------------|
| 0 | 0 | 0 | $35 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 0 | 1 | 0 | $35 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 0 | 2 | 0 | $35 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 1 | 0 | 0 | $35 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 1 | 1 | 0 | $35 + 0 = 35$ | 35 | (0, 0, 1) |
| 1 | 2 | 0 | $35 + 0 = 35$ | 35 | (0, 0, 1) |
| 2 | 0 | 0 | $35 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 2 | 1 | 55 | $35 + 0 = 35$ | 55 | (1, 0, 0) |
| 2 | 2 | 55 | $30 + 0 = 30$ | 55 | (1, 0, 0) |
| 3 | 0 | 0 | $30 - \infty = -\infty$ | 0 | (0, 0, 0) |
| 3 | 1 | 55 | $30 + 0 = 30$ | 55 | (1, 0, 0) |
| 3 | 2 | 70 | $35 + 55 = 90$ | 90 | (1, 0, 1) |

Solusi optimum dari persoalan ini adalah $X = (1, 0, 1)$ dengan $\Sigma p = f_3(3, 2) = 95$.

IV. PENERAPAN PEMROGRAMAN DINAMIS UNTUK MENENTUKAN MENU DIET

Misalkan suatu rumah makan X menjual makanan dan minuman sebagai berikut beserta rincian kalori dan zat gizi lainnya, serta harganya.

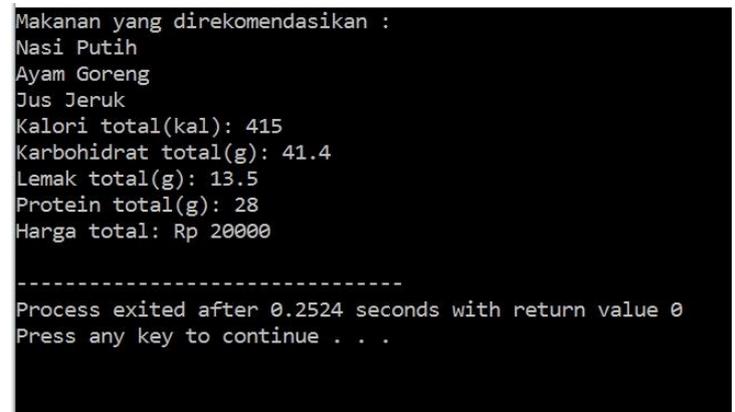
Tabel V Rincian menu di rumah makan X

| Menu | Kalori | Karbo (g) | Lemak (g) | Protein (g) | Harga (Rp) |
|-----------------|--------|-----------|-----------|-------------|------------|
| Nasi Putih | 129 | 27.9 | 0.3 | 2.7 | 4000 |
| Nasi Merah | 110 | 20.8 | 0.5 | 2.6 | 5000 |
| Ikan Goreng | 205 | 0 | 12 | 22 | 8000 |
| Ayam Goreng | 221 | 0 | 13.1 | 24.3 | 10000 |
| Telur Dadar | 98 | 1.2 | 7.1 | 7 | 5000 |
| Sayur Hijau | 76 | 12 | 2 | 2.6 | 6000 |
| Sapi Lada Hitam | 240 | 0 | 15 | 26 | 12000 |
| Mie Goreng | 221 | 40.3 | 3.3 | 7.3 | 13000 |
| Jus Tomat | 35 | 3.9 | 0.2 | 0.9 | 6000 |
| Jus Jeruk | 65 | 11.5 | 0.1 | 1 | 6000 |
| Jus Apukat | 160 | 9 | 15 | 2 | 6000 |

Dari menu tersebut ingin dipilih serangkaian makanan dan atau minuman dengan tujuan mengoptimalkan kalori atau zat gizi tertentu yang didapat dengan batasan tertentu, misalnya batasan anggaran yang dimiliki, maksimal kalori yang didapat, atau batasan terhadap zat gizi tertentu. Persoalan ini dapat dimodelkan sebagai *Multi-dimensional Knapsack Problem* dengan asumsi setiap jenis makanan atau minuman hanya dapat dipilih satu kali saja. Yang menjadi status (y) dalam permasalahan ini adalah jumlah kalori atau zat gizi yang ingin dibatasi (bisa lebih dari 1) dengan tahapan sebanyak 11 (ada 11 buah objek yang dapat dipilih) dengan tujuan untuk mengoptimalkan suatu kandungan tertentu yang diperoleh dari makanan atau minuman yang dipilih (asumsi seseorang harus membeli makanan dan minuman).

Persoalan ini akan diselesaikan dengan menggunakan program dengan metode pemrograman dinamis yang telah dijelaskan pada bagian II. Program dibuat dengan menggunakan algoritma yang telah dibahas pada bagian III dengan pendekatan rekursif (*top-down*). Pengguna akan memasukkan batasan yang diinginkan (maksimal 2 batasan pada program) serta hal yang ingin dioptimasi kemudian program akan mengeluarkan rekomendasi menu makan yang sesuai dengan batasan-batasan yang diberikan. Untuk bilangan desimal akan dikonversikan menjadi bilangan bulat dengan cara dikalikan dengan 10 saat perhitungannya. Berikut adalah hasil eksperimen yang telah dilakukan oleh penulis dengan tiga buah kasus uji (program hanya menampilkan 1 solusi saja).

1. Kasus Uji 1
 Batasan pertama: harga \leq Rp 20.000
 Batasan kedua: lemak \leq 15g
 Optimasi: kalori (untuk menambah berat badan)



Gambar 2 – Hasil Pengujian 1

2. Kasus Uji 2
 Batasan pertama: kalori \leq 390 kal
 Batasan kedua: lemak \leq 24 g
 Optimasi: protein (untuk menambah massa otot)

```

Makanan yang direkomendasikan :
Telur Dadar
Sapi Lada Hitam
Jus Tomat
Kalori total(kal): 373
Karbohidrat total(g): 5.1
Lemak total(g): 22.3
Protein total(g): 33.9
Harga total: Rp 23000
-----
Process exited after 0.1121 seconds with return value 0
Press any key to continue . . .

```

Gambar 3 – Hasil Pengujian 2

3. Kasus Uji 3
 - Batasan pertama: karbohidrat \leq 25 g
 - Batasan kedua: lemak \leq 15 g
 - Optimasi: protein (untuk mengurangi berat badan sekaligus menambah massa otot)

```

Makanan yang direkomendasikan :
Nasi Merah
Ayam Goreng
Jus Tomat
Kalori total(kal): 366
Karbohidrat total(g): 24.7
Lemak total(g): 13.8
Protein total(g): 27.8
Harga total: Rp 20000
-----
Process exited after 0.05055 seconds with return value 0
Press any key to continue . . .

```

Berdasarkan pengujian yang telah dilakukan, dapat dilihat bahwa hasil menu makan yang didapatkan dengan menggunakan pemrograman dinamis merupakan hasil yang paling optimal dari semua kemungkinan solusi yang ada dengan waktu pemrosesan yang cukup cepat yaitu di bawah 0,1s. Namun, hasil yang didapatkan ini belum mempertimbangkan kategori menu yang dipilih, misalnya harus terdiri dari makanan 4 sehat 5 sempurna, dan hasil yang didapatkan mungkin akan lebih baik jika jumlah objek (menu) yang disediakan lebih banyak lagi. Selain itu, terdapat juga permasalahan pada batasan memori yang dapat digunakan oleh program sehingga program yang diimplementasikan dibatasi jumlah batasan (*constraint*) sebanyak semaksimal 2 dan juga dengan semakin banyaknya objek yang tersedia waktu pemrosesannya akan semakin lama. Untuk perkembangan lebih lanjut masalah pemilihan menu makan ini dapat dimodelkan menjadi persoalan *knapsack* yang *multi-objective* (lebih dari satu tujuan yang dioptimasi), *multi-dimensional* (lebih dari satu batasan), dan juga *unbounded* (tidak ada batasan pada jumlah setiap objek) serta membagi objek-objek menjadi kelompok-kelompok tertentu (*multiple choice knapsack*). Persoalan itu tentunya akan membutuhkan algoritma yang berbeda untuk menyelesaikannya dan hal tersebut tidak akan dibahas di makalah ini.

V. KESIMPULAN

Algoritma pemrograman dinamis dapat digunakan untuk menyelesaikan persoalan optimasi dikarenakan adanya salah satu prinsip yang digunakan dalam pemrograman dinamis yaitu prinsip optimalitas. Salah satu aplikasi dari pemrograman dinamis adalah untuk menyelesaikan persoalan *knapsack* dalam waktu *pseudo polynomial time* yang lebih baik dibandingkan dengan metode-metode lain, misalnya *bruteforce* dan *branch and bound*. Namun, kelemahannya terletak pada keterbatasan memori yang dimiliki untuk memorisasi setiap tahap yang ada.

Persoalan *knapsack* yang diselesaikan dengan pemrograman dinamis dapat digunakan untuk memodelkan berbagai persoalan yang berhubungan dengan pemilihan objek-objek tertentu dengan tujuan optimasi yang memiliki batasan tertentu. Salah satunya adalah untuk memilih menu diet yang memenuhi tujuan tertentu, misalnya mendapatkan kalori semaksimal mungkin, dengan batasan-batasan tertentu. Dari hasil eksperimen dapat dilihat bahwa hasil yang didapatkan merupakan hasil yang optimal. Namun, hasil tersebut belum bisa digunakan pada dunia nyata karena implementasi dari program belum mempertimbangkan berbagai faktor lainnya, seperti jenis makanan dan jumlah yang dapat dipilih untuk setiap makanannya. Terdapat juga faktor-faktor lain yang mungkin juga dapat memengaruhi pemilihan menu makan, seperti alergi terhadap makanan tertentu.

VI. UCAPAN TERIMA KASIH

Saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena dengan berkat dan rahmat-Nya saya dapat menyelesaikan tugas makalah ini dengan baik dan tepat pada waktunya. Saya mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. dan Ibu Masayu Leylia Khodra, S.T., M.T., selaku dosen mata kuliah IF2211 Strategi Algoritma yang telah membimbing pengerjaan tugas ini. Saya juga mengucapkan terima kasih kepada kedua orang tua saya yang telah mendukung saya.

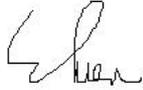
REFERENCES

- [1] <https://www.geeksforgeeks.org/knapsack-problem>, diakses pada tanggal 12 Mei 2018 pukul 19.48 WIB.
- [2] R. Munir, Program Dinamis, 2018. [Online] Tersedia dalam [http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/ProgramDinamis%20\(2018\).pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/ProgramDinamis%20(2018).pdf). [diakses 11 Mei 2017 pukul 22.13 WIB].
- [3] <http://doktersehat.com/apakah-diet-itu/>, diakses pada tanggal 12 Mei 2018 pukul 12.56 WIB.
- [4] <https://www.fatsecret.co.id/kalori-gizi/>, diakses 12 Mei 2017 pukul 16.00 WIB.
- [5] <http://www.brodibalofitness.com/2016/11/cara-menghitung-kalori-panduan-lengkap/>, diakses 12 Mei 2017 pukul 17.30 WIB.
- [6]

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, 12 Mei 2018



I Putu Eka Surya A – 13516061