Implementasi Algoritma Branch and Bound untuk Memaksimalkan Nilai Guna Uang Dalam Pembelian Makanan Bagi Mahasiswa

Stefanus Jeremy Aslan 13519175
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519175@std.stei.itb.ac.id:

Abstract*—Pengelolaan penggunaan uang memainkan faktor yang penting dalam kehidupan, terutama apabila uang yang dimiliki terbatas. Untuk mahasiswa, pengelolaan penggunaan uang merupakan hal yang perlu diperhatikan terutama untuk memenuhi kebutuhan pokok seperti makanan. Mahasiswa didorong untuk memaksimalkan nilai guna uang dengan cara mendapatkan nutrisi semaksimal mungkin dari makanan-makanan yang dibeli dengan uang tersebut. Pembelian makanan dilakukan dengan tetap memperhatikan keterbatasan uang dan variasi makanan. Dalam memecahkan persoalan optimasi tersebut, dapat dimplementasikan suatu strategi algoritma. Salah satu algoritma yang cocok untuk persoalan optimasi tersebut yaitu algoritma *branch and bound

Keywords—algoritma; branch and bound; optimasi; maksimal; uang; nilai nutrisi;

I. PENDAHULUAN

Dalam perjuangan selama berkuliah, setiap mahasiswa didorong untuk menghemat uang yang diberikan oleh orang tua untuk kebutuhan hidup. Dengan uang yang terbatas, memaksimalkan nilai guna uang yang dimiliki menjadi hal yang penting. Salah satu penggunaan uang yang paling umum yaitu untuk membeli kebutuhan hidup seperti makanan. Setiap pembelian makanan harus dipertimbangkan dengan sebaik mungkin sehingga total nutrisi yang didapatkan dari makanan bernilai maksimal. Tidak hanya itu, konsumsi makanan yang sama sebaiknya dihindari karena menyebabkan kejenuhan yang buruk untuk performa mahasiswa dalam berkuliah. Oleh karena itu, variasi makanan sangat dianjurkan untuk ditekankan.

Dalam memecahkan persoalan ini, dapat digunakan strategi algoritma. Salah satu strategi algoritma yang dapat digunakan yaitu algoritma *branch and bound*. Dalam implementasi algoritma *branch and bound*, dapat digunakan bahasa pemrograman seperti bahasa pemrograman Python.

II. LANDASAN TEORI

A. Algoritma

Untuk persoalan dengan instansiasi yang besar, solusinya menjadi lebih sulit untuk ditentukan. Maka dari itu, diperlukan prosedur umum untuk menyelesaikan persoalan tersebut yang disebut dengan algoritma. Algoritma merupakan urutan langkah-langkah untuk memecahkan suatu persoalan dengan memproses masukan menjadi luaran. Beberapa strategi algoritma yang umum antara lain adalah sebagai berikut.

1) Algoritma Brute-Force

Algoritma *brute-force* merupakan algoritma yang paling sederhana karena algoritma ini menyelesaikan persoalan tanpa mempertimbangkan efisiensi proses. Karena algoritma ini umumnya tidak membutuhkan pemikiran yang terlalu mendalam, algoritma ini disebut sebagai algoritma *brute-force* (paksaan).

2) Algoritma Greedy

Algoritma *greedy* merupakan algoritma dengan dasar yaitu pengambilan keputusan terbaik yang ada pada setiap tahapan proses penyelesaian persoalan tanpa mempertimbangkan tahapan setelahnya.

3) Algoritma Divide and Conquer

Algoritma *divide and conquer* merupakan algoritma dengan strategi memecah persoalan menjadi persoalan-persoalan yang lebih kecil. Persoalan-persoalan yang lebih kecil tersebut kemudian diselesaikan satu persatu kemudian digabungkan untuk menyelesaikan persoalan semula.

4) Algoritma Decrease and Conquer

Algoritma *decrease and conquer* merupakan algoritma dengan strategi membagi persoalan menjadi persoalan yang lebih kecil, kemudian menyelesaikan sebagian persoalan yang lebih kecil tersebut. Solusi dari penyelesaian persoalan yang lebih kecil tersebut kemudian digunakan untuk mewakilkan solusi dari penyelesaian persoalan semula.

5) Algoritma Backtracking

Algoritma *backtracking* memiliki kemiripan dengan algoritma greedy, dengan perbedaan utama yaitu algoritma ini dapat kembali tahapan proses yang sudah dilalui untuk menelusuri pengambilan keputusan selain yang telah diambil. Dengan demikian, mungkin untuk algoritma *backtracking* dapat menghasilkan lebih dari satu buah solusi. Algoritma ini memetakan tahapan penyelesaian

permasalahan ke dalam pohon ruang status. Untuk simpul status yang tidak 'mengarah' ke solusi, simpul status akan dimatikan. Umumnya digunakan untuk persolan non-optimasi.

6) Algoritma Branch and Bound

Algoritma yang dapat digunakan untuk menyelesaikan persoalan optimasi. Algoritma ini memetakan tahapan penyelesaian permasalahan ke dalam pohon ruang status yang setiap simpulnya menggambarkan keadaan pada tahapan penyelesaian tertentu.

B. Algoritma Branch and Bound

Algoritma *branch and bound* merupakan algoritma yang ditujukan untuk menyelesaikan persoalan optimasi, persoalan yang meminimalkan atau memaksimalkan suatu fungsi objektif tertentu tanpa melanggar batasan yang ada dalam penyelesaian persoalan tersebut [2]. Algoritma ini memetakan tahapan penyelesaian permasalahan ke dalam pohon ruang status yang setiap simpulnya menggambarkan keadaan pada tahapan penyelesaian tertentu.

Setiap simpul pohon memiliki harga (*cost*) yang menjadi pertimbangan untuk ekspansi simpul berikutnya atau tahapan yang diambil selanjutnya. Batasan yang berlaku dalam penyelesaian persoalan dicerminkan dengan aturan yang disebut sebagai fungsi pembatas. Apabila suatu nilai pada suatu simpul status melanggar batasan yang ada, simpul tersebut akan 'dimatikan' sehingga tidak menjadi pertimbangan untuk pengambilan keputusan pada tahapan-tahapan selanjutnya [1]. Ketika ditemukan simpul status yang memberikan solusi persoalan, setiap simpul hidup yang costnya lebih kecil dari *cost* simpul status solusi terbaik akan 'dimatikan' [1]. Jika ditemukan simpul status solusi persoalan lain, akan dilakukan perbandingan *cost* dengan simpul status persoalan yang baru ditemukan untuk mencari simpul status solusi dengan cost terbesar [1].

Suatu simpul status dilakukan terminasi apabila salah satu dari kondisi berikut terpenuhi

Salah satu persoalan yang dapat diselesaikan menggunakan algoritma *branch and bound* adalah persoalan knapsack [3]. Dalam persoalan ini, *problem solver* dihadapi dengan berbagai objek berbeda yang memiliki harga dan bobot masing-masing. *Problem solver* diminta untuk mencari kombinasi dari objek-objek yang ada sedemikian sehingga total harga objek pada kombinasi bernilai maksimal dengan syarat total bobot objek pada kombinasi tidak melebihi bobot tertentu. Persoalan ini merupakan salah satu dari persoalan klasik yang dapat dipecahkan menggunakan berbagai strategi algoritma umum.

III. IMPLEMENTASI DAN PEMBAHASAN

Persoalan yang dihadapi yaitu persoalan pemilihan makanan-makanan sedemikian rupa sehingga mahasiswa dapat memanfaatkan uang yang dimiliki dengan maksimal dan memakan makanan yang bervariasi.

Bahasa pemrograman Python digunakan untuk implementasi algoritma *branch and bound*. Python memiliki kemampuan untuk membaca data dari file eksternal dan memasukannya ke dalam bentuk *list* yang dengan tipe elemen bervariasi membuat Python efisien dalam analisis data. Selain itu, sintax bahasa pemrograman Python yang sederhana membuat kode program memiliki keterbacaan yang lebih baik sehingga pemrogram dapat melakukan perbaikan atau perubahan dengan mudah [4]. Tidak hanya itu, Python dilengkapi dengan berbagai macam *library* yang bermanfaat untuk pembuatan program. Karena fitur-fitur tersebut, Python dipilih sebagai bahasa pemrograman yang digunakan untuk mengimplementasikan algoritma *branch and bound*.

Data makanan yang menjadi pertimbangan disimpan dalam file eksternal dengan format CSV. Berikut merupakan data makanan yang disajikan dalam bentuk tabel.

Nasi Goreng	8000	25000
Nasi Uduk	7500	15000
Nasi Padang	7000	22000
Sate Padang	8500	28000
Nasi Geprek	8000	18000
Tolak Udara	2000	2500
Nasi Kecap	7000	7500
Nasi Garam	3500	7000
Indahmie	1500	5000
Supermurah	2500	4000
Nasi Polos	2000	6000

Tabel 3.1. Data file eksternal CSV

Pada tabel, kolom pertama berisi nama makanan, kolom kedua berisi nilai nutrisi makanan, dan kolom ketiga berisi harga makanan. Setelah membaca data dari CSV tersebut dan memasukannya ke dalam sebuah *array of list*, diterapkanlah algoritma *branch and bound* untuk menyelesaikan dan menampilkan solusi persoalan.

A. Implementasi Algoritma Branch and Bound

Pada program yang dibuat, algoritma *branch and bound* diimplementasikan dalam bentuk prosedur yang memanggil prosedur lain yang bersifat rekursif. Semua simpul keadaan/status disimpan dalam sebuah list *state*. Setiap simpul menyimpan informasi antara lain nama simpul status, index makanan pada *array of list* makanan, *cost* simpul, total nilai nutrisi makanan yang telah diambil sejauh ini, uang yang telah digunakan sejauh ini, list nama simpul anak, list informasi pengambilan makanan sejauh ini dalam bilangan biner, dan status simpul.

Menggunakan algoritma *branch and bound*, proses penyelesaian persoalan dapat dipetakan menjadi sebuah pohon biner dengan simpul menggambarkan state dan kedalaman simpul menggambarkan jumlah makanan yang dipertimbangkan untuk diambil atau tidak diambil. Pemeriksaan makanan yang dipertimbangkan untuk diambil atau tidak dilakukan secara urut berdasarkan urutan data makanan pada file eksternal CSV. Sebagai contoh, untuk

simpul dengan kedalaman 0, berarti belum ada makanan yang dipertimbangkan untuk diambil pada state yang diwakili simpul tersebut; untuk simpul dengan kedalaman 2, berarti state yang diwakili oleh simpul tersebut sudah mempertimbangkan makanan pertama dan makanan kedua pada file eksternal CSV.

Persoalan yang dihadapi memiliki pola yang sama seperti halnya persoalan knapsack integer, sehingga bisa digunakan fungsi cost knapsack integer. Referensi [3] menyampaikan bahwa untuk persoalan knapsack, cost atau batas atas (upper bound) simpul i dihitung sebagai penjumlahan total keuntungan yang sudah dicapai (F) ditambah dengan perkalian sisa kapasitas knapsack (K – W) dengan rasio keuntungan per bobot objek yang tersisa berikutnya (p_{i+1}/wi_{+1}), atau dengan rumus:

$$c(i) = F + (K - W)p_{i+1}/w_{i+1} \quad [3]$$

Diadaptasikan ke persoalan yang dihadapi, keuntungan yang sudah dicapai (F) merupakan total nutrisi yang didapat dari penerimaan sebuah makanan untuk pembelian, kapasitas knapsack (K) merupakan uang yang digunakan pengguna untuk keperluan pembelian makanan, total bobot objek yang diambil (W) merupakan total harga seluruh makanan yang diterima untuk pembelian, harga objek (p) merupakan nilai nutrisi makanan, dan bobot objek (w) merupakan harga makanan.

Proses algoritma *branch and bound* untuk memecahkan persoalan ini adalah sebagai berikut.

Pertama, list *state* diinisiasi dengan elemen akar, kemudian prosedur rekursif akan dijalankan. Elemen akar tersebut bersifat hardcoded, berisi antara lain informasi uang yang dimiliki mahasiswa dan data makanan pertama pada file eksternal dengan format CSV.

Kedua, dari semua simpul yang ada pada list *state*, dicari simpul daun hidup dengan *cost* terbesar untuk diekspansi, menghasilkan 2 state baru yang diwakilkan oleh simpul pada pohon. Pembangkitan state menggambarkan aksi selanjutnya dan 2 *state* baru yang dibangkitkan menggambarkan 2 keadaan berbeda: keadaan yang menggambarkan penerimaan pembelian makanan dan keadaan yang menolak pembelian makanan; makanan yang dimaksud merupakan makanan dengan indeks makanan berupa kedalaman *state* yang baru saja dibangkitkan, dengan indeks makanan pertama memiliki indeks 1. Setiap simpul memiliki salah satu dari 3 macam status: status 0 berarti simpul sudah diekspansi, status 1 berarti simpul belum diekspansi, dan status 2 berarti simpul dimatikan karena melanggar fungsi pembatas atau karena dieliminasi sebab cost-nya lebih kecil dari cost simpul jawaban terbaik sejauh ini.

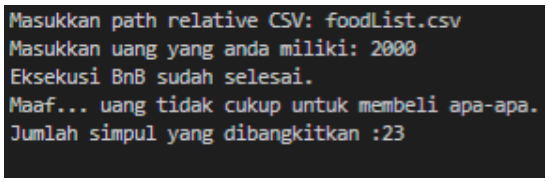
Ketiga, setelah tidak ada simpul yang memiliki jenis status 1, cost terbesar dari simpul yang telah mengeksplorasi semua makanan akan dicari, kemudian list informasi pengambilan makanan yang disimpan pada simpul tersebut akan diambil. Menggunakan list dengan elemen biner tersebut, dicetak kombinasi makanan yang paling menguntungkan yang dapat dibeli oleh mahasiswa, total nilai nutrisi yang diperoleh, uang yang digunakan untuk membeli semua makanan yang ada pada kombinasi makanan tersebut, dan uang tersisa yang dimiliki mahasiswa.

B. Hasil Pengujian Program

Terdapat enam kasus yang akan diujikan: kasus ketika jumlah uang yang dimasukkan tidak cukup untuk membeli makanan apapun, kasus ketika uang yang dimasukkan cukup untuk membeli semua makanan yang tercantum pada file eksternal CSV, kasus ketika uang hampir cukup untuk membeli semua makanan yang tercantum pada file eksternal dengan format CSV, kasus ketika ketika uang hanya cukup untuk membeli makanan termurah, dan kasus ketika uang hanya cukup untuk membeli makanan termurah pertama atau makanan termurah kedua, dengan makanan termurah kedua memiliki nilai nutrisi lebih baik, dan kasus dengan masukan uang tanpa pertimbangan kondisi apapun.

1) Hasil Pengujian Kasus I

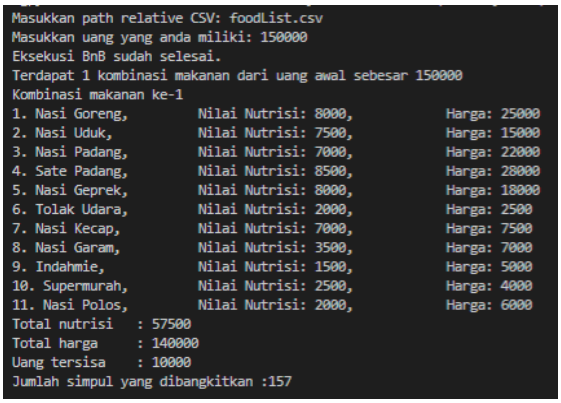
Kasus uji pertama menguji hasil program ketika input uang pengguna lebih kecil dari makanan termurah yang tercantum pada file eksternal dengan format CSV. Pada file eksternal yang digunakan, makanan termurah yaitu makanan bernama Tolak Udara seharga 2500. Untuk pengujian ini, program diberi masukan uang pengguna sebesar 2000. Berikut merupakan hasil yang ditampilkan oleh program.



Gambar 3.1. Hasil pengujian kasus I

2) Hasil Pengujian Kasus II

Kasus uji kedua menguji hasil pemrograman ketika input uang pengguna cukup untuk membeli semua barang yang tercantum pada file eksternal berformat CSV. Dengan menjumlahkan semua harga makanan, didapatkan total harga sebesar 140000. Untuk pengujian ini, program diberi masukan uang pengguna sebesar 150000. Berikut merupakan hasil yang ditampilkan oleh program.

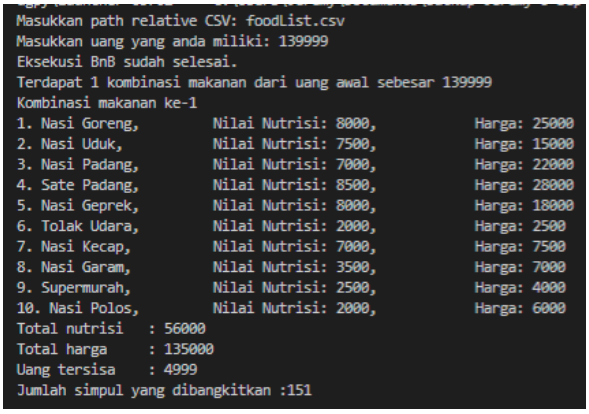


Gambar 3.2. Hasil pengujian kasus II

3) Hasil Pengujian Kasus III

Kasus uji ketiga menguji hasil program ketika masukan uang pengguna hampir cukup untuk membeli semua makanan yang tercantum pada file eksternal berformat

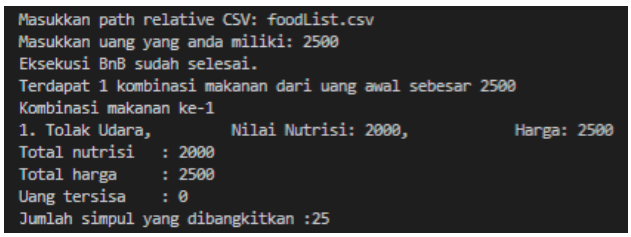
CSV. Dari pengujian kasus I, didapatkan bahwa total harga semua makanan yang tercantum pada file eksternal yang digunakan bernilai sebesar 14000. Oleh karena itu, program akan diberikan masukan uang pengguna sebesar 139999 sebagai nilai untuk pengujian. Berikut merupakan hasil yang ditampilkan oleh program.



Gambar 3.3. Hasil pengujian kasus III.

4) Hasil Pengujian Kasus IV

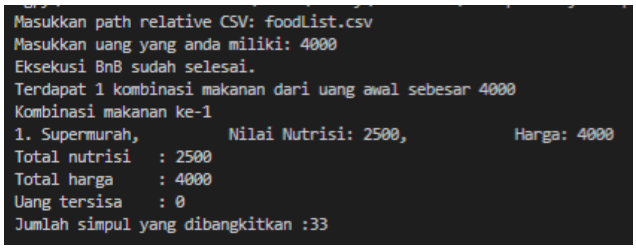
Kasus uji keempat menguji hasil program ketika masukan uang pengguna hanya cukup untuk membeli makanan termurah yang tercantum pada file eksternal. Pada file eksternal yang digunakan, makanan termurah yaitu Tolak Udara dengan harga 2500. Oleh karena itu, digunakan masukan uang pengguna sebesar 2500 sebagai nilai penguji. Berikut merupakan hasil yang ditampilkan oleh program.



Gambar 3.4. Hasil pengujian kasus IV

5) Hasil Pengujian Kasus V

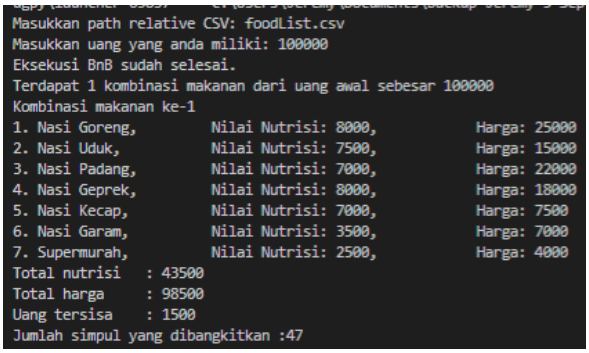
Kasus uji kelima menguji hasil program ketika masukan uang pengguna cukup untuk membeli makanan kedua termurah yang tercantum pada file eksternal. Pada file eksternal yang digunakan, makanan termurah yaitu Tolak Udara dengan harga 2500 dan nilai nutrisi 2000, sedangkan makanan kedua termurah yaitu Supermurah dengan harga 4000 dan nilai nutrisi 2500. Pengujian dilakukan dengan memberi masukan uang pengguna sebesar 4000. Berikut merupakan hasil yang ditampilkan oleh program.



Gambar 3.5. Hasil pengujian kasus V

6) Hasil Pengujian Kasus VI

Kasus uji keenam menguji hasil program ketika masukan uang pengguna bernilai bebas tanpa mempertimbangkan data yang ada pada file eksternal. Sebagai contoh, terdapat mahasiswa dengan uang 100000 yang ingin merencanakan pembelian makanan agar nutrisi yang didapat bernilai maksimal. Berikut merupakan hasil yang ditampilkan dari program.



Gambar 3.6. Hasil pengujian kasus VI

C. Pembahasan Hasil Pengujian Program

1) Pembahasan Pengujian Kasus I

Pada kasus I, program diuji apakah mampu memberi hasil yang tepat atau tidak apabila mahasiswa tidak memiliki uang yang cukup untuk membeli makanan apapun yang tercatat pada file eksternal. Untuk kasus ini, secara logis dapat ditentukan bahwa solusi persoalan yaitu tidak ada makanan yang dibeli, sehingga list biner informasi pengambilan makanan yang disimpan oleh simpul jawaban pastilah berisi elemen 0 sejumlah data macam makanan pada file eksternal CSV atau sama dengan [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Program dianggap lulus pengujian apabila jawaban yang diberikan tidak menampilkan makanan apapun kepada pengguna, melainkan pesan informatif.

2) Pembahasan Pengujian Kasus II

Pada kasus II, program diuji apakah mampu memberi hasil yang tepat atau tidak apabila mahasiswa memiliki uang yang cukup untuk membeli seluruh pilihan makanan yang ada. Untuk kasus ini, secara logis dapat ditentukan bahwa jawaban persoalan yaitu semua opsi makanan diterima untuk dibeli, sehingga list biner informasi pengambilan makanan yang disimpan oleh simpul jawaban pastilah berisi elemen 1 sejumlah data macam makanan pada file eksternal CSV atau sama dengan [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1].

Dari hasil yang diberikan oleh program, dapat dilihat bahwa kombinasi makanan meliputi semua makanan yang tercantum pada file eksternal yang digunakan. Dengan demikian, dapat dikatakan bahwa program implementasi branch and bound lulus pengujian kasus II.

3) Pembahasan Pengujian Kasus III

Pada kasus III, program diuji apakah mampu memberi hasil yang tepat ketika masukan uang pengguna hampir cukup untuk membeli semua makanan yang tercantum pada file eksternal. Apabila program berhasil mengembalikan kombinasi makanan yang meliputi semua macam makanan kecuali makanan dengan nilai nutrisi terendah, maka program lulus pengujian kasus ini.

Dari hasil yang diberikan program, dapat dilihat bahwa kombinasi makanan yang diberikan meliputi semua makanan kecuali makanan bernama Indahmie, yang memiliki nilai nutrisi terendah sebesar 1500. Maka dari itu, dapat dikatakan bahwa program implementasi branch and bound lulus pengujian kasus III.

4) Pembahasan Pengujian Kasus IV

Pada kasus IV, program diuji apakah mampu memberikan hasil yang tepat ketika masukan uang pengguna hanya cukup untuk membeli makanan dengan harga yang termurah. Apabila program menampilkan hanya 1 makanan pada layar dan makanan tersebut merupakan makanan termurah yang dicantumkan pada file eksternal, maka program lulus pengujian kasus ini.

Dari hasil yang diberikan program, dapat dilihat bahwa makanan yang ditampilkan hanya 1, yaitu Tolak Udara dengan harga 2500, harga terendah dari semua harga makanan pada file eksternal yang digunakan. Oleh karena itu, program dapat dikatakan lulus pengujian kasus IV.

5) Pembahasan Pengujian Kasus V

Pada kasus V, program diuji apakah mampu memberikan hasil yang tepat ketika masukan uang pengguna hanya cukup untuk membeli salah satu dari 2 makanan termurah. Pada file eksternal, makanan kedua termurah memiliki nutrisi yang lebih tinggi dari makanan termurah, tetapi memiliki indeks yang lebih tinggi sehingga dipertimbangkan setelah makanan termurah dipertimbangkan. Apabila program menolak pengambilan makanan termurah dan mengambil makanan termurah kedua yang memberikan nilai nutrisi yang lebih tinggi, program lulus pengujian kasus ini.

Dari hasil yang diberikan, dapat dilihat bahwa makanan termurah, Tolak Udara, yang merupakan makanan dengan urutan keenam, ditolak untuk menjadi bagian dari makanan yang diambil, sedangkan makanan kedua termurah, Supermurah, yang memiliki nilai nutrisi lebih tinggi, diterima. Oleh karena itu, program dapat dikatakan lulus pengujian kasus V.

6) Pembahasan Pengujian Kasus VI

Pada kasus ini, program diuji apakah mampu menerima masukan uang pengguna dan menampilkan jawaban yang tepat atau tidak. Berbeda dengan kasus-kasus sebelumnya, masukan ini tidak diikat oleh kondisi tambahan apapun selain kondisi uang pengguna harus valid (lebih besar dari 0), tidak melanggar fungsi pembatas, dan belum dibahas pada kasus-kasus sebelumnya. Pengujian ini lebih diarahkan untuk lebih meyakinkan pengguna bahwa program hasil implementasi branch and bound untuk penghematan uang makanan dapat bekerja dengan semestinya dan mengembalikan hasil yang benar.

Misalkan, ada seorang mahasiswa yang memiliki uang untuk kebutuhan makanan sebesar 100000 dan ingin menggunakannya sedemikian sehingga makanan-makanan yang kemudian akan dibeli memiliki total nutrisi maksimal. Karena sudah sangat frustasi dengan pelajaran kuliah, ia sangat tidak mau semakin frustasi karena kemonotonan makanan. Ia ingin memakan makanan yang bervariasi dan tidak sama, setidaknya hingga awal bulan datang dan orang tua mahasiswa mengirimkan uang bulanan baru.

Dari hasil pengujian yang diberikan program, didapat sebuah kombinasi makanan yang tersusun dari tujuh makanan. Kombinasi makanan tersebut memberikan total nutrisi 43500 dengan harga total makanan lebih kecil sama dengan 10000, yaitu seharga 98500. Jika dilakukan pemeriksaan semua kombinasi makanan yang mungkin secara brute-force, maka akan didapat kombinasi makanan yang sama.

D. Perbandingan Algoritma Branch and Bound dengan Algoritma Brute-Force dan Algoritma Greedy

Selain menggunakan algoritma branch and bound, terdapat algoritma-algoritma lain yang dapat menyelesaikan permasalahan. Beberapa di antaranya yaitu algoritma *brute-force* dan algoritma *greedy*.

1) Algoritma Brute Force

Dalam menghadapi persoalan optimasi, algoritma brute force menyelesaikan permasalahan dengan menelusuri semua kemungkinan lalu dari semua kemungkinan tersebut, dipilih jawaban yang memenuhi kriteria yang dicari. Untuk persoalan ini, setiap makanan akan diberikan salah satu dari 2 perlakuan: akan dibeli atau tidak akan dibeli. Karena dari file eksternal, dicantumkan hanya 11 data makanan, terdapat 2^{11} keadaan yang bisa terjadi, terlepas dari batasan yang ada. Selanjutnya, dengan mempertimbangkan fungsi pembatas, dicari nilai nutrisi total yang terbesar.

Jika diibaratkan dengan simpul status algoritma *branch and bound*, algoritma brute-force mengharuskan pembangkitan simpul status sebanyak 2^{11} kali. Tidak hanya itu, semua dari 2^{11} simpul status harus diperiksa satu persatu apakah melanggar fungsi pembatas atau tidak, dan jika tidak, harus dibandingkan dengan nilai nutrisi total terbesar yang didapat hingga saat pemeriksaan simpul status tertentu.

Sebagai pembanding, dipilih kasus dari pengujian program yang membangkitkan status terbanyak, yaitu kasus I dengan 157 jumlah status. Apabila seorang mahasiswa memiliki uang sebanyak 150000, algoritma *branch and bound* membangkitkan status jauh lebih sedikit dibandingkan algoritma *brute-force* dengan perbandingan 1:13,0446. Berdasarkan perbandingan tersebut, dapat dikatakan bahwa algoritma branch and bound memiliki efisiensi yang lebih baik daripada algoritma brute-force.

2) Algoritma Greedy.

Algoritma *greedy* menghadapi persoalan optimasi dengan cara bertahap seperti algoritma *branch and bound.* Akan tetapi, algoritma greedy berlangsung 1 arah dan tidak dapat meninjau kembali tahapan yang telah dilalui sebelumnya. Apabila suatu makanan diterima sebagai makanan yang direkomendasikan untuk dibeli, maka tidak dapat ditinjau keadaan ketika makanan tersebut tidak diterima sebagai makanan yang direkomendasikan untuk dibeli. Sayangnya, sifat tidak dapat membatalkan keputusan yang diambil ini membuat algoritma *greedy* menjadi kurang dapat diandalkan ketika menghadapi persoalan optimasi.

Dengan prinsip mengambil keputusan yang paling menguntungkan pada suatu tahapan tanpa mempertimbangkan tahapan mendatang, setiap makanan pasti akan diterima sebagai makanan yang direkomendasikan asalkan total harga setelah menerima makanan tersebut tidak lebih besar uang yang dimiliki pengguna. Ini disebabkan setiap penerimaan makanan pasti akan menambahkan nilai nutrisi total dibandingkan tidak menerima makanan. Ditambah dengan sifat tidak bisa membatalkan keputusan, algoritma *greedy* dapat mengembalikan sebuah jawaban dengan sangat cepat.

Dalam persoalan yang dihadapi, terdapat 11 makanan yang dipertimbangkan untuk dibeli atau tidak. Dengan algoritma *branch and bound*, status dari menerima atau menolak suatu makanan selalu setidaknya membangkitkan 2 simpul status untuk setiap kedalaman simpul, kecuali simpul kedalaman 0. Algoritma greedy hanya menelusuri salah satu dari keadaan tersebut, tergantung keputusan yang diambil. Apabila diibaratkan dengan pohon ruang status, algoritma greedy hanya perlu membangkitkan 11 status. Algoritma *branch and bound* perlu setidaknya membangkitkan 23 simpul: 1 simpul akar dan 2 simpul untuk setiap pertimbangan pengambilan makanan dari total 11 makanan. Ini berarti algoritma *greedy* melalui tahapan proses pertimbangan yang lebih sedikit dari separuh jumlah proses pertimbangan pada kasus terbaik algoritma *branch and bound*.

Sayangnya, walaupun lebih cepat dari algoritma branch and bound, kemungkinan algoritma *greedy* mengembalikan solusi yang tidak tepat sangatlah besar disebabkan limitasi tidak dapat membatalkan keputusan, berlangsung 1 arah, dan tidak mempertimbangkan tahapan selanjutnya. Karena mengembalikan solusi yang benar menjadi prioritas dalam pemecahan masalah, algoritma *branch and bound* merupakan algoritma yang lebih baik daripada algoritma *greedy* untuk menangani persoalan ini.

IV. KESIMPULAN

Berdasarkan program hasil implementasi algoritma *branch and bound* beserta hasil pengujiannya, didapatkan bahwa algoritma *branch and bound* efektif dalam menyelesaikan persoalan. Tidak hanya itu, dalam menghadapi persoalan memaksimalkan nilai guna dalam persoalan ini, algoritma *branch and bound* superior terhadap algoritma *brute-force* dalam efisiensi dan juga superior terhadap algoritma *greedy* dalam efektivitas.

VIDEO LINK AT YOUTUBE

https://www.youtube.com/watch?v=QvZAb9cYwFQ&ab_channel=StefanusJeremyAslan

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, sebab melalui perlindungan dan penyertaan yang dilimpahkan-Nya, penulis dapat menyelesaikan makalah ini tanpa kendala berarti dan tepat pada waktunya.

Penulis juga mengucapkan terima kasih kepada dosen pengajar mata kuliah Strategi Algoritma kelas K-04, Bapak Rinaldi Munir selaku dosen pengajar penulis, dan juga kepada seluruh dosen pengajar mata kuliah Strategi Algoritma periode 2020/2021, sebab tanpa bimbingan, ajaran langsung, dan video pelajaran yang diberikan, tidak mungkin penulis mampu untuk menciptakan makalah ini.

REFERENCES

[1] Anany Levitin, Introduction to the Design & Analysis of Algorithms, 3rd Edition, Addison-Wesley, pp.460, 2012

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branchand-Bound-2021-Bagian4.pdf

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pengantar-Strategi-Algoritma-(2021).pdf

[5] https://www.python.org/doc/essays/blurb/

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.

Jakarta, 11 Mei 2021

Stefanus Jeremy Aslan - 13519175