Pengaplikasian Algoritma *Branch and Bound* untuk Melakukan *Smart Buying* di Tengah Pandemi COVID-19

Anna Elvira Hartoyo - 13518045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]13518045@std.stei.itb.ac.id [2]naelvirahart@gmail.com

***Abstrak*—Pandemi COIVD-19 yang tengah melanda negara-negara di dunia, tak terkecuali Indonesia memberikan dampak besar di berbagai aspek kehidupan manusia. Salah satu dampaknya adalah banyak masyarakat yang melakukan *panic buying* untuk menyimpan persediaan bahan makanan untuk jangka waktu tertentu. Hal ini berakibat pada minimnya stok bahan makanan yang tersedia di pusat perbelanjaan. Sementara, ada sebagian masyarakat yang tidak mempunyai budget untuk dapat membeli bahan makanan dalam jumlah besar sekaligus. Padahal, seorang manusia dewasa hanya memerlukan rata-rata 2.200 sampai dengan 2.500 kalori sehari. Perilaku *panic buying* ini harus diubah menjadi *smart buying,* salah satu caranya adalah dengan membatasi budget uang untuk berbelanja. Pada makalah ini, penulis membahas aplikasi algoritma *branch and bound* dalam menentukan bahan makanan yang cukup dibeli dengan memperhitungkan budget uang yang tersedia, harga bahan makanan, dan kandungan kalori pada bahan makanan. Dilakukan optimasi pembelian bahan makanan agar dapat memenuhi kebutuhan kalori manusia selama beberapa hari, tetapi juga tetap memperhatikan ketersediaan stok bahan makanan bagi masyarakat lain.**

Kata kunci—branch and bound; optimasi; smart buying;

I. PENDAHULUAN

Sejak bulan Desember 2019, dunia digemparkan dengan kemunculan virus corona. Virus yang berasal dari Kota Wuhan, Cina ini merupakan penyebab penyakit coronavirus disease 2019 atau yang lebih popular dengan istilah COVID-19. Penyebaran penyakit ini sangatlah cepat dan telah memakan banyak sekali korban jiwa di berbagai negara di dunia. Pada pertengahan bulan Maret 2020 lalu, World Health Organization (WHO) telah mengumumkan penyakit ini berstatus pandemi global. Menurut Kamus Besar Bahasa Indonesia (KBBI) pandemi adalah wabah yang berjangkit serempak di mana-mana, meliputi daerah geografi yang luas. Tak terkecuali, Indonesia juga termasuk dalam daftar negara yang terdampak COVID-19.

Dampak yang ditimbulkan dari pandemi COVID-19 telah meluas ke berbagai aspek kehidupan. Salah satu dampak yang cukup serius adalah masalah fenomena *panic buying*, yaitu perilaku memborong bahan makanan dan sembako di tengah kepanikan. Tak dapat dipungkiri, merebaknya kasus corona di Indonesia juga membuat sebagian masyarakat melakukan *panic buying*. Perilaku *panic buying* ini dapat menular antar masyarakat, akibatnya banyak stok bahan makanan yang kosong karena sudah diborong oleh sebagian masyarakat. Sementara itu, tidak semua orang mempunyai budget yang banyak untuk dapat membeli bahan makanan dalam jumlah besar sekaligus.

Gambar 1. Ilustrasi panic buying
Sumber: [2]

Panic buying tentunya menjadi sebuah perilaku yang tidak baik dan harus dihindari. Ada cara yang lebih bijaksana, yaitu dengan melakukan *smart buying* atau berbelanja secara cerdas. Menyimpan stok bahan makanan memang diperlukan untuk meminimalkan pergi keluar rumah, namun belilah keperluan yang sangat dibutuhkan dalam jumlah yang cukup untuk orang atau keluarga selama waktu tertentu, yang rasional atau sesuai kemampuan. Jika Anda mempunyai budget lebih untuk membeli stok bahan makanan, dengan menerapkan smart buying, Anda harus membatasi budget Anda sehingga tidak membeli bahan makanan secara berlebihan yang nantinya juga dapat terbuang jika terlalu banyak. Menurut Barbara Gordon [3], kebutuhan kalori seorang manusia dalam sehari bervariasi, tergantung pada faktor jenis kelamin, usia, dan pekerjaan. Namun, rata-rata manusia dewasa hanya memerlukan 2.200 sampai dengan 2.500 kalori per hari. Jumlah kalori tersebut sudah dapat terpenuhi dari makanan yang secukupnya, tanpa perlu berlebihan.

Setiap bahan makanan memberikan nilai kalori tertentu bagi tubuh manusia saat dikonsumsi. Akan tetapi, seringkali

seseorang mengalami kesulitan untuk memilih bahan makanan yang harus dibeli dengan budget terbatas. Akibatnya, bahan makanan yang dibeli tidak memberikan nilai kalori yang maksimal.

Pada kesempatan ini, penulis akan mengimplementasikan algoritma *branch and bound* untuk membantu masyarakat melakukan *smart buying*, yaitu membeli bahan bakanan yang tepat dengan budget yang terbatas agar mendapatkan keuntungan berupa nilai kalori yang maksimal. Hal-hal yang akan diperhitungkan antara lain: budget maksimum berupa uang dalam mata uang Rupiah, harga masing-masing bahan makanan per satuan berat (Rp/Kg), kandungan kalori dalam masing-masing bahan makanan per satuan massa (kalori/Kg), dan banyaknya bahan makanan yang hendak dibeli dalam satuan massa (Kg).

Untuk menunjang penelitian, penulis juga membuat sebuah aplikasi sederhana berbasis *text-terminal* (*Command Line Interface*) dengan bahasa Python 3 untuk mensimulasikan perhitungan.

II. Dasar Teori

A. Integer Knapsack Problem

Integer knapsack problem atau dikenal juga dengan istilah *1/0 knapsack problem* merupakan masalah optimasi yang sangat populer di dunia pemrograman. Tujuan dari masalah optimasi adalah untuk meminimalkan atau memaksimalkan suatu fungsi objektif tanpa melanggar batasan (*constraint*) yang ditetapkan.

Pada persoalan ini diberikan n buah objek dan sebuah knapsack dengan kapasitas bobot maksimal K tertentu. Setiap objek memiliki properti bobot (*weigth*) w_i dan keuntungan (*profit*) p_i. Persoalan menekankan cara untuk memilih objek-objek yang dimasukkan ke dalam *knapsack* sedemikian sehingga diperoleh keuntungan (*profit*) yang maksimal. Total bobot objek yang dimasukkan ke dalam knapsack tidak boleh melebihi kapasitas knapsack. Secara formal, persoalan dapat didefinisikan sebagai berikut:

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

dengan constraint $\sum_{i=1}^{n} w_i x_i \leq K$

di mana $x_i = 0\ atau\ 1$ dan $i = 1, 2, 3, \ldots, n$

Sebagai contoh, pada tabel berikut terdapat 3 jenis barang dengan bobot dan keuntungannya masing-masing. Diketahui kapasitas knapsack sebesar 16.

Tabel 1. Contoh persoalan integer knapsack

No	Bobot	Profit
1	6	12
2	5	15
3	10	20

Untuk memperoleh profit yang maksimal, barang yang harus diambil adalah barang 2 dan 3 sehingga diperoleh total profit sebesar 35 dan total bobot sebesar 15 yang tidak melebihi kapasitas *knapsack*.

Ada banyak algoritma dan pendekatan yang dapat dilakukan untuk menyelesaikan persoalan ini, antara lain algortima *brute force*, *greedy*, *dynamic programming*, dan *branch and bound*. Setiap algoritma tersebut memiliki kompleksitas dan optimalitas perhitungan yang berbeda-beda. Pada makalah ini akan digunakan pendekatan dengan algroritma *branch and bound*.

B. Algortima Branch And Bound

Algortima *branch and bound* atau B&B merupakan salah satu algoritma pencarian di dalam ruang solusi secara sistematis. Penggunaan algoritma ini yang paling umum adalah untuk menyelesaikan persoalan optimasi, yaitu mencari nilai Pembentukkan ruang solusi dari suatu persoalan akan diorganisasikan ke dalam pohon ruang status dalam bentuk graf. Ruang solusi ini dibangun dengan pencarian secara melebar atau *Breadth First Search* (BFS). Setiap simpul diberi sebuah nilai atau *cost* $\hat{c}$(i) yang merupakan nilai taksiran lintasan termurah ke simpul tujuan yang melali simpul tersebut. Berbeda dengan BFS yang menggunakan aturan FIFO dalam ekspansi simpul, algoritma B&B mengekspan simpul yang memiliki nilai *cost* $\hat{c}$(i) paling kecil (*least cost search*) pada kasus minimasi.

Untuk mengurangi jumlah simpul yang harus dibangkitkan, algoritma B&B menerapkan pemangkasan pada simpul-simpul yang tidak mengarah ke solusi. Simpul layak untuk dipangkas jika memenuhi salah satu dari tiga kriteria pemangkasan berikut:

- Nilai dari suatu simpul tidak lebih baik dari solusi nilai terbaik sejauh ini
- Simpul merupakan solusi yang tidak *feasible* karena melanggar batasan.
- Solusi yang *feasible* pada simpul tersebut hanya terdiri atas satu titik.

Secara umum, berikut langkah-langkah umum pada algortima B&B:

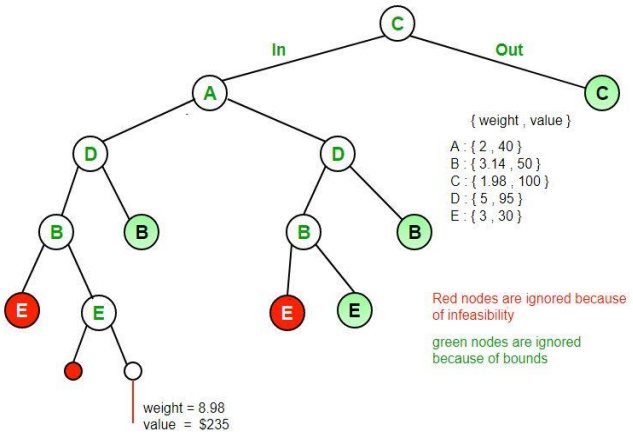
1. Masukkan simpul akar ke dalam antrian (*queue*) Q. Jika simpul tersebut adalah simpul solusi, maka solusi sudah ditemukan. Tidak perlu melanjutkan pencarian.
2. Jika Q kosong, berarti tidak ada solusi untuk persoalan. Hentikan pencarian.
3. Jika Q tidak kosong, pilih dari Q simpul yang mempunyai nilai *cost* $\hat{c}$(i) terkecil. Jika terdapat lebih dari satu simpul i yang memenuhi, pilih salah satu secara random.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, sehingga pencarian dihentikan. Jika simpul i bukan simpul solusi, ekspansi simpul tersebut dengan membangkitkan seluruh anak-anaknya. Jika i tidak dapat diekspansi, kembali ke langkah 2.
5. Untuk setiap anak j dar simul I, hitung nilai *cost* $\hat{c}$(j) dan masukkan ke dalam Q.
6. Kembali ke langkah 2.

Banyak persoalan optimasi yang dapat diselesaikan menggunakan algortima *branch and bound*. Algoritma ini cocok digunakan untuk menyelesaikan persoalan *integer*

knapsack problem dibanding algortima lain karena kompleksitasnya yang relatif rendah, yaitu sebesar $O(n^2)$. Berikut algoritma untuk menyelesaikan persoalan integer knapsack problem dengan B&B:

1. Urutkan semua objek berdasarkan rasio profit/bobot secara menurun.
2. Inisialisasi nilai profit maksimum (maxProfit) dengan 0.
3. Buat sebuah antrian (*queue*) kosong Q
4. Buat sebuah simpul dummy dan masukkan ke antrian Q. Nilai *profit* dan bobot simpul *dummy* tersebut adalah 0.
5. Lakukan langkah-langkah berikut selama antrian Q belum kosong:
   a. Ambil sebuah elemen simpul dari Q sebagai simpul u
   b. Hitung nilai profit dari simpul u. Jika nilainya lebih besar dari maxProfit, perbarui nilai maxProfit.
   c. Hitung nilai bound dari simpul u. Jika nilainya lebih besar dari maxProfit, masukkan simpul u ke Q.
   d. Jika u bukan bagian dari solusi, tambahkan u ke antrian Q tanpa menambahkan nilai profit dan bobot simpul selanjutnya

Nilai bound digunakan untuk mengetahui suatu simpul dapat memberikan solusi yang lebih baik dari solusi terbaik sejauh ini atau tidak.



Gambar 2. Pohon ruang status yang dibentuk saat penyelesaian knapsack problem dengan B&B
Sumber: [1]

III. METODOLOGI

Dalam makalah ini, persoalan pemilihan bahan makanan dimodelkan sebagai integer knapsack problem. Bobot dalam perosalan ini adalah harga bahan makanan, sedangkan profit adalah banyaknya kalori yang diperoleh. Batasan atau contraint yang merupakan kapasitas knapsack dalam persoalan ini adalah budget maksimum yang dialokasikan untuk berbelanja bahan makanan.

Beberapa hal yang perlu dipersiapkan sebelum melakukan perhitungan dengan program adalah:

1. Bahan makanan apa saja yang menurut prakiraan ingin dibeli sebagai stok makanan
2. Bobot yang ingin dibeli untuk masing-masing bahan makanan dalam satuan kilogram (Kg)
3. Kisaran kandungan kalori per kilogram masing-masing bahan makanan dalam satuan (kalori/Kg)
4. Kisaran harga per kilogram untuk masing-masing bahan makanan dalam satuan (Rp/Kg)

Program akan menerima sebuah berkas file eksternal yang terdiri atas empat kolom, yaitu nama bahan makanan, banyaknya bahan makanan yang hendak dibeli, kandungan kalori per kilogram, dan harga per kilogram. Usahakan agar daftar bahan makanan bervariasi dan lengkap, mulai dari sumber karbohidrat, protein, vitamin, serat, hingga lemak.

Data dari berkas eksternal ini akan diambil dan dimasukkan ke dalam sebuah list. Berikut kode program untuk mengambil data tersebut:

```
def readFile(file):
    global items
    f = open(file)
    for line in f:
       temp = (line.split())
       items.append(Objek(temp[0], float(temp[1]
       ), float(temp[1]) * float(temp[3]), float
       (temp[1]) * float(temp[2])))
```

Selanjutnya, dibutuhkan *input* dari pengguna berupa budget uang dalam rupiah yang akan dibelanjakan. Berikut adalah layer penerimaan *input* dari pengguna yang telah dibuat oleh penulis.

Gambar 3. Tampilan penerimaan input dari pengguna
Sumber: dokumentasi penulis (dibuat 2 Mei 2020)

Seluruh data yang diperlukan program telah tersedia, lalu program akan mulai menghitung keuntungan maksimum yang dapat diperoleh tanpa melanggan *constraint*. Sudah dijelaskan sebelumnya bahwa pencarian solusi dilakukan menggunakan algoritma *branch and bound*. Pencarian simpul tujuan dilakukan sesuai algoritma yang telah dijelaskan pada bab sebelumnya. Algoritma utama *branch and bound* dibuat oleh penulis dalam fungsi solve yang menerima tiga parameter, yaitu kapasitas masksimum *knapsack*, yang dalam kasus ini adalah budget maksimum dari input pengguna, sebuah *list* yang berisi informasi bahan makanan dari file eksternal, dan panjang list tersebut.

Fungsi solve membutuhkan fungsi lainnya, yaitu calcBound untuk menghitung nilai *bound* dari suatu simpul. Fungsi

calcBound ini memiliki empat parameter, yaitu simpul yang akan dihitung nilai boundnya, panjang list bahan makan, kapasitas maksimum *knapsack*, dan *list* bahan makanan.

Berikut ini adalah kode program untuk fungsi calcBound dan solve.

```
def calcBound(node, n, capacity, items):
    # node is not feasible
    if(node.weight >= capacity):
        return (0)

    profitBound = node.profit
    j = node.level + 1
    totalWeight = node.weight

    while ((j<n) and (totalWeight + items
     [j].weight <= capacity)):
        totalWeight += items[j].weight
        profitBound += items[j].value
        j += 1

    if(j<n):
        profitBound += (capacity –
         totalWeight) * (items[j].value /
         items[j].weight)

    return profitBound
```

```
def solve(capacity, items, n):
    global result

    items.sort(key = lambda x: (x.value/x.weight
     ), reverse = True)
    result = [0 for i in range (len(items))]
    q = []
    dummy = Node(-1, 0, 0, 0) #dummy element
    q.append(dummy)

    v = Node(0, 0, 0, 0)

    maxProfit = 0

    while (not len(q)==0):
        w = q.pop(0)

        if(w.level == -1):
            v.level = 0


        if(w.level == n-1):
            break

        v.level = w.level + 1


    v.weight = w.weight + items[v.level].weight
     v.profit = w.profit + items[v.level].value


    if (v.weight <= capacity and v.profit > max
    Profit):
            result[v.level] = 1
            maxProfit = v.profit

    v.bound = calcBound(v, n, capacity, items)

    if (v.bound > maxProfit) :
            r = deepcopy(v)
            q.append(r)

    v.weight = w.weight
    v.profit = w.profit
    v.bound = calcBound(v, n, capacity, items)

    if (v.bound > maxProfit) :
            r = deepcopy(v)
            q.append(r)
            q.append(r)

  return maxProfit
```

Setelah perhitungan selesai, program akan menampilkan daftar bahan makanan yang sebaiknya dibeli, total hraga dari bahan makanan yang dibeli, dan total kalori yang dapat diperoleh dari bahan makanan tersebut. Selain itu, program juga akan menampilkan informasi jumlah hari bahan makanan tersebut cukup untuk satu orang dewasa, dengan asumsi kebutuhan kalori sebesar 2.200 kalori per hari.

IV. Pengujain

Pada bab ini algoritma akan dieksekusi dengan beberapa kasus pengujian. Sebelum mengeksekusi program, dibutuhkan data bahan makanan yang menurut prakiraan hendak dibeli.

Data terdiri atas empat kolom, yaitu nama bahan makanan, bobot yang hendak dibeli (Kg), kandungan kalori (kal/Kg), dan harga (Rp/Kg). Berikut contoh tabel data untuk studi kasus.

Tabel 2. Data Bahan Makanan

Nama Bahan Makanan	Bobot (Kg)	Kandungan Kalori (kal/Kg)	Harga (Rp/Kg)
Telur	1	1551	12.000
Beras	5	1290	10.000
Wortel	1	410	5.000
Kentang	0.5	850	6.000
Ayam	1	2390	35.000
Sapi	0.5	2505	80.000
Apel	1	512	13.000

Data pada tabel 2 lalu dimuat ke dalam sebuah file eksternal dalam format .txt sebagai berikut.

```
telur   1   1551    12000
beras   5   1290    10000
wortel  1   410     5000
kentang 0.5 850     6000
ayam    1   2390    35000
sapi    0.5 2505    80000
apel    1   512     13000
```

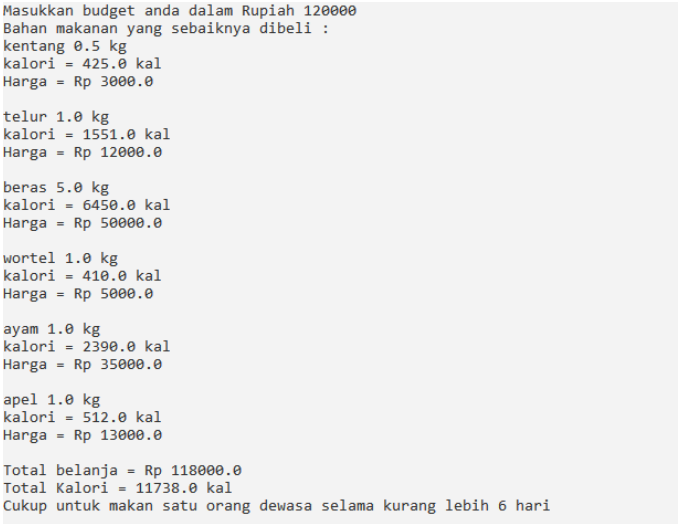
Ketika program dijalankan, program akan membaca data dari file eksternal tersebut. Selanjutnya, program memerlukan *input* budget maksimum yang dialokasikan untuk berbelanja.

Untuk pengujian pertama, misalkan budget yang dimasukkan sebesar Rp120.000.

Gambar 4. Input untuk pengujian 1
Sumber: koleksi pribadi (dibuat 2 Mei 2020)

Kemudian program akan menjalankan algoritma *branch and bound* hingga menemukan hasil berupa keuntungan maksimum, yang dalam hal ini adalah banyaknya kalori maksimum yang dapat diperoleh. Program akan menuliskan daftar bahan makanan yang sebaiknya dibeli beserta detail total kalori dan harganya, total uang yang digunakan, total kalori yang akan diperoleh, dan informasi lama hari stok bahan makanan itu dapat mencukupi kebutuhan kalori satu orang dewasa.

Dalam studi kasus yang pertama, diperoleh output berupa enam bahan makanan yang disarankan untuk dibeli karena akan memberikan keuntungan yang maksimum. Bahan makanan tersebut adalah kentang, kelur, beras, wortel, ayam, dan apel. Total uang yang harus dikeluarkan untuk berbelanja sebesar Rp118.000 yang masih berada dalam batas budget maksimum. Keuntungan maksimum berupa kalori yang diperoleh sebesar 11.738 kalori. Jumlah kalori ini dapat mencukupi kebutuhan kalori satu orang dewasa untuk kurang lebih 6 hari.



Gambar 5. Solusi untuk pengujian 1
Sumber: koleksi pribadi (dibuat 2 Mei 2020)

Untuk pengujian yang kedua masih menggunakan file eksternl yang sama, budget yang dimasukkan adalah sebesar Rp50.000.

Gambar 6. Input untuk pengujian 2
Sumber: koleksi pribadi (dibuat 2 Mei 2020)

Program akan kembali mengeksekusi algoritma yang sama lalu menampilkan hasil sesuai *constraint*. Dalam studi kasus kedua, diperoleh output berupa empat bahan makanan yang disarankan untuk dibeli, yaitu kentang, telur, beras, dan wortel. Total uang yang dibelanjakan adalah Rp70.000. Keuntungan maksimum berupa kalori yang diperoleh sebesar 8.836 kalori yang dapat mencukupi kebutuhan satu orang dewasa selama kurang lebih 4 hari.

```
Masukkan budget anda dalam Rupiah 75000
Bahan makanan yang sebaiknya dibeli :
kentang 0.5 kg
kalori = 425.0 kal
Harga = Rp 3000.0

telur 1.0 kg
kalori = 1551.0 kal
Harga = Rp 12000.0

beras 5.0 kg
kalori = 6450.0 kal
Harga = Rp 50000.0

wortel 1.0 kg
kalori = 410.0 kal
Harga = Rp 5000.0

Total belanja = Rp 70000.0
Total Kalori = 8836.0 kal
Cukup untuk makan satu orang dewasa selama kurang lebih 4 hari
```

Gambar 7. Solusi untuk pengujian 2
Sumber: koleksi pribadi (dibuat 2 Mei 2020)

V. KESIMPULAN

Untuk melakukan *smart buying* dalam bentuk penyelesaian model persoalan optimasi *integer knapsack problem*, dapat diaplikasikan algoritma *branch and bound*. Pengaplikasian algortima *branch and bound* ini terbukti dapat digunakan untuk membantu masyrakat melakukan *smart buying* di tengah pandemi COVID-19. Pemilihan jenis bahan makanan dengan budget terbatas ini dilakukan dengan mempertimbangkan budget maksimal untuk berbelanja, kandungan kalori per kilogram bahan makanan, dan harga per kilogram bahan makanan. Solusi yang diperloleh berupa daftar bahan makanan yang disarankan untuk dibeli beserta detail total uang yang dibelanjakan, total kalori yang diperoleh, dan informasi lama hari kalori tersebut dapat mencukupi kebutuhan satu orang dewasa.

Melakukan *smart buying* tidak hanya terbatas pada masa pandemic COVID-19 saja. Penggunaan program ini juga dapat dilakukan untuk membantu seseorang dengan uang terbatas untuk memilih bahan makanan yang memberikan jumlah kalori paling banyak.

LINK VIDEO PADA YOUTUBE

Video penjelasan makalah ini dapat dilihat pada link berikut https://www.youtube.com/watch?v=TO7Pe43Zv7s

UCAPAN TERIMA KASIH

Pertama-tama penulis ingin manyampaikan ucapan syukur kepada Tuhan Yang Maha Esa atas berkat-Nya penulis diberi kesehatan dan kekuatan sehingga dapat menyelesaikan makalah ini dengan baik. Penulis berterima kasih kepada orang tua penulis yang telah mendukung secara moral, doa, dan pembiayaan. Penulis juga mengucapkan terima kasih kepada dosen mata kuliah strategi algoritma, yaitu Bapak Dr.Ir. Rinaldi Munir, M.T., Ibu Dr. Nur Ulfa Maulidevi ST,M.Sc., dan Ibu Dr. Masayu Leylia Khodra, ST., MT yang telah membimbing penulis dalam belajar strategi algoritma selama satu semester ini. Terakhir, penulis juga mengucapkan terima kasih kepada teman-teman Teknik Informatika 2018 yang saling mendukung dan menyemangati hingga proses pembuatan makalah ini dapat selesai tepat waktu.

REFERENSI

[1] Anonim.2019."Implementation of 0/1 Knapsack using Branch and Bound".https://www.geeksforgeeks.org/implementation-of-0-1-knapsack-using-branch-and-bound/ diakses tanggal 30 April 2020

[2] CNN Indonesia.2020. "Alasan Psikologi di Balik 'Panic Buying'".https://www.cnnindonesia.com/gaya-hidup/20200322161747-284-485813/alasan-psikologi-di-balik-panic-buying diakses tanggal 29 April 2020

[3] Gordon, Barbara.2019."How Many Calories Do Adults Need?".https://www.eatright.org/food/nutrition/dietary-guidelines-and-myplate/how-many-calories-do-adults-need diakses tanggal 29 April 2020

[4] Munir, Rinaldi. 2009.Diktat Kuliah IF2211: Strategi Algoritma.Bandung: Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

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, 3 Mei 2020

Anna Elvira Hartoyo 13518045