Penerapan *Dynamic Programming* dalam Menyusun Paket Kebutuhan Pokok dalam Masa Pandemik COVID-19

Gregorius Jovan Kresnadi – 13518135[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]13518135@std.stei.itb.ac.id

Abstrak*—Dalam masa pandemic COVID-19, pemerintah telah menganjurkan warganya untuk tidak bepergian dan tetap tinggal di rumah, bahkan di beberapa kota besar telah diterapkan PSBB (Pembatasan Sosial Berskala Besar). Namun terdapat sebuah kendala terutama bagi masyarakat kalangan bawah yang tidak bisa menetap di rumah karena keberlangsungan hidupnya bergantung pada pekerjaannya di luar rumah. Maka dari itu, terdapat beberapa organisasi masyarakat dan komunitas yang berbaik hati untuk menggalang dana dan menyusun paket-paket bantuan untuk dibagikan kepada masyarakat yang membutuhkan. Makalah ini ditujukan untuk membantu orang yang ingin menyusun paket kebutuhan pokok dengan biaya tertentu dengan bantuan *Dynamic Programming.

***Kata Kunci—Dynamic Programming;* kebutuhan pokok*; budget;* optimal*;* prioritas**

I. PENDAHULUAN

Pada tahun 2019, ditemukan sebuah *strain* virus corona baru di Wuhan, Cina. Virus tersebut menyebar ke seluruh daerah Wuhan, lalu menyebar ke daerah lain di negeri Cina, dan akhirnya ke seluruh dunia. Pada tanggal 12 Maret 2020, WHO menyatakan bahwa penyebaran virus tersebut telah mencapai status pandemik yang bersifat global. Virus yang menjadi penyebab tersebut dinamakan COVID-19. [1]

Dalam upaya memperlambat penyebaran, peneliti dan pemerintah di seluruh dunia telah bersepakat bahwa metode yang paling tepat adalah untuk melakukan *social distancing*. Tindakan ini, yang sekarang diganti oleh pemerintah Indonesia menjadi *physical distancing*, adalah salah satu langkah pencegahan dan pengendalian infeksi virus COVID-19 dengan menganjurkan orang sehat untuk membatasi kunjungan ke tempat ramai dan kontak langsung dengan orang lain. Dalam penerapannya, seseorang tidak diperbolehkan untuk berjabat tangan serta menjaga jarak setidaknya 1 meter saat berinteraksi dengan orang lain, terutama dengan orang yang sedang sakit atau beresika tinggi menderita COVID-19. Selain itu, penerapan *social distancing* yang lain adalah bekerja dari rumah (*work from home*), belajar di rumah secara *online* bagi siswa sekolah dan mahasiswa, menunda acara yang dihadiri orang banyak, dan tidak mengunjungi orang yang sedang sakit. [2]

Di Indonesia, pemerintah telah gencar menerapkan *physical distancing*, bahkan di beberapa kota besar seperti Jakarta dan Bandung telah diterapkan PSBB, atau Pembatasan Sosial Berskala Besar. Penerapan PSBB ini juga sangat mendukung tindakan *work from home,* di mana setiap orang harus bekerja dari rumah. Namun kenyataannya, tidak semua orang bisa bekerja dari rumah. Masyarakat ekonomi kelas bawah telah dilaporkan mengalami kesulitan bekerja dari rumah. Pekerjaan seperti pedagang keliling, penjual di pasar, dan pemulung tidak mungkin dilakukan dari rumah. Pekerja-pekerja ini terpaksa tetap bekerja untuk memenuhi kebutuhan hidup sehar-harinya, walau pendapatan mereka mulai menipis. Dilansir dari kompasiana.com, sebagian besar warga khususnya yang pendapatan ekonominya tergolong kelas menengah ke bawah mulai merasa panik menghadapi kebutuhan pokok harian, sementara kabar bantuan sembako yang dijanjikan pemerintah hingga kini secara nyata belum tersentuh secara merata dan warga masih menunggu kepastian kapan akan dibagikan kepada mereka. [3]

Dalam berupaya membantu sesama, terdapat beberapa kelompok masyarakat yang tergerak hatinya untuk mencoba membantu mereka. Namun karena ketatnya kebijakan pemerintah dalam PSBB, mereka terpaksa untuk membatasi orang yang dapat dibantu. Bantuan yang paling nyata dan berguna bagi yang membutuhkan adalah pembagian sembako. Contoh yang penulis ambil dalam penulisan makalah ini adalah bantuan yang diberikan dari Gereja Katolik Paroki Salib Suci Bandung. Menurut Randy, pengurus organisasi Gereja Kemuning (Dewan Pastoral Paroki), dalam program sosial ini Gereja berinisiatif untuk meringankan beban warga Gereja yang terkena dampak dari pandemik wabah COVID-19 dengan menyumbangkan paket bahan pokok bagi keluarga-keluarga yang membutuhkan. Data orang yang perlu dibantu didapatkan dari sensus umat Katolik dan data dari ketua lingkungan. Data kemudian direkap untuk menyaring keluarga yang membutuhkan. Dari data tersebut juga dibuat *budgeting* untuk menyusun paket bahan pokok yang akan dibagikan tiap minggunya. Randy juga menambahkan, "Kami tidak langsung membantu semua warga di sekitar karena beberapa alasan. Pertama, dana yang tersedia tidak mencukupi untuk membantu semua orang. Kedua, dengan adanya kebijakan *social*

distancing, kami perlu mengurangi resiko dengan cara mengatur sistem pembagian yang aman." (Randy, wawancara, 1 Mei 2020, pukul 17.00).

II. LANDASAN TEORI

A. Rekursi

Rekursi adalah sebuah fungsi yang memanggil dirinya sendiri secara langsung maupun tidak langsung. Fungsi yang melakukan rekursi disebut fungsi rekursif. Rekursi banyak digunakan dalam penyelesaian masalah, contohnya dalam permasalahan *Tower of Hanoi*, *Depth First Search*, dan *Dynamic Programming*.

Sebuah fungsi rekursi memiliki 2 buah bagian. Bagian basis adalah sebuah kondisi awal di mana solusi untuk permasalahan yang lebih kecil telah disediakan. Bagian rekursi adalah sebuah kondisi di mana masalah terhitung besar, maka fungsi rekursi akan memecah permasalahan menjadi bagian-bagian kecil dan akan memanggil dirinya sendiri dengan masukan berupa pecahan masalahnya. Di akhir perhitungan ketika pecahan masalahnya cukup kecil untuk masuk ke basis, maka fungsi akan menggabungkan solusi-solusi yang didapat dari pecahan menjadi solusi dari masalah awalnya.

Keuntungan menggunakan fungsi rekursi adalah penulisan kode yang lebih rapi, dan juga untuk memecahkan masalah yang tidak dapat diselesaikan dengan pengulangan biasa. Misalnya dalam sebuah struktur pohon, cara yang paling tepat untuk mencari sebuah elemen dalam pohon tersebut adalah menggunakan rekursi di mana bagian rekursi akan memanggil fungsinya sendiri dengan memasukkan upapohon sebagai parameternya hingga ditemukan elemen yang dicari.

Kerugian dari memakai fungsi rekursi terletak pada bagian teknikal di mana memori komputer yang dipakai akan lebih banyak akibat dari pemanggilan fungsinya sendiri yang lebih cepat memenuhi *memory stack*. Proses ini juga berpotensi terjadinya *stack overflow* pada sistem. [6]

B. Dynamic Programming

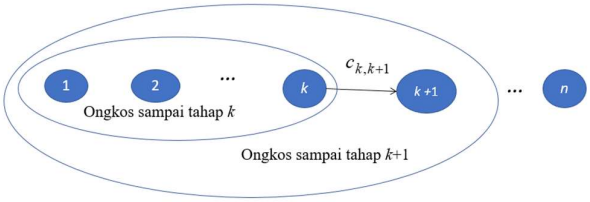
Program Dinamis atau *Dynamic Programming* adalah metode pemecahan masalah dengan cara menguraikan solusi yang disusun menjadi sekumpulan tahapan (stage). Setiap stage ini disusun sedemikian mungkin sehingga dapat dipandang sebagai serangkaian keputusan yang saling berkaitan. Kata 'program' dalam istilah program dinamis di sini tidak berkaitan dengan pemrograman dan penulisan kode, namun merujuk kepada metode tabular yang digunakan saat memecahkan masalah. Sedangkan istilah 'dinamis' muncul karena dalam proses pencarian solusi, metode melakukan perhitungan dengan menggunakan tabel yang dapat berkembang.

Dynamic Programming umumnya digunakan dalam permasalahan optimasi. Persoalan ini dapat menghasilkan solusi lebih dari satu, di mana solusi tersebut memiliki suatu nilai. Solusi di mana nilai tersebut bernilai maksimum atau minimum dinamakan solusi optimal. Dengan *dynamic programming*, dapat ditemukan solusi optimal yang lebih dari satu, jika tersedia.

Algoritma lain yang biasa digunakan untuk permasalahan optimasi adalah algoritma *greedy*. Secara prinsip, kedua algoritma ini ditujukan ke hal yang sama yaitu optimasi. Namun perbedaan antara kedua algoritma terletak pada perangkaian keputusannya. Pada algortima *greedy*, hanya satu buah rangkaian keputusan yang dihasilkan, yaitu solusi yang pertama kali disusun oleh algoritma, dibandingkan dengan algoritma *dynamic programming* yang mampu menyusun lebih dari satu rangkaian keputusan.

Pada *dynamic programming*, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. Prinsip ini berarti jika solusi total optimal, maka bagian solusi sampai tahap ke-*k* juga optimal. Prinsip Optimalitas ini juga berarti jika proses pencarian solusi pada tahap *k* ke tahap *k*+1, hasil optimal pada tahap ke-*k* dapat digunakan tanpa perlu kembali pada tahap awal. Metode ini menggunakan ongkos untuk mencari solusi optimalnya. Maka dari itu, sesuai dengan Prinsip Optimalitas, ongkos pada tahap *k*+1 dihasilkan dari ongkos pada tahap ke-*k* ditambah ongkos dari tahap *k* ke tahap *k*+1, atau $c_{k,k+1}$.

$$\text{Cost}_{k+1} = \text{Cost}_k + \text{Cost}_{k,k+1}$$



Gambar 1. Ilustrasi Prinsip Optimalitas (Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf)

Dynamic programming memiliki beberapa ciri dan karakteristik [4]:

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam kemungkinan masukan yang ada pada suatu tahap.
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
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 onkos dari tahap tersebut ke tahap berikutnya.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya 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.

Dalam merancang algoritma *dynamic programming*, terdapat serangkaian langkah yang wajib ditaati:

1. Karakteristikkan struktur solusi optimal
   ➔ Merancang struktur tahap, variabel keputusan, jumlah status, dsb.
2. Definisikan secara rekursif nilai solusi optimal
   ➔ Merancang fungsi rekursif yang menghubungkan nilai optimal suatu tahap dengan nilai optimal pada tahap selanjutnya
3. Hitung nilai solusi optimal secara maju / mundur
   ➔ Melakukan pencarian nilai optimal, proses direpresentasikan menggunakan tabel
4. Rekonstruksi solusi optimal
   ➔ Menyusun ulang setiap langkah solusi dari arah sebaliknya untuk memverifikasi jawaban

Langkah 1 sampai 3 menunjukkan dasar dari pencarian solusi menggunakan algoritma *Dynamic Programming,* sedangkan langkah ke 4 dapat diabaikan jika yang diperlukan hanyalah nilai optimal dari solusinya saja, bukan langkah yang diperlukan untuk mencapai solusi tersebut.

Terdapat 2 jenis pendekatan dengan Program Dinamis:

1. Program Dinamis Maju (*Forward* atau *up-down*)
   ➔ Perhitungan dilakukan dari tahap 1, 2, … n-1, n.
2. Program Dinamis Mundur (*Backward* atau *bottom-up*)
   ➔ Perhitungan dilakukan dari tahap n, n-1, …, 2, 1.

Jika algoritma dilakukan secara maju, maka pencarian solusi dimulai dari tahap awal dan pada langkah ke-4 algoritma akan menyusun ulang solusi secara mundur. Begitu pula sebaliknya jika algoritma dilakukan secara mundur.

III. PEMBAHASAN

Untuk penerapan algoritma *dynamic programming*, akan diambil studi kasus dari hasil wawancara dengan narasumber mengenai upaya pemberian bantuan Gereja Katolik Paroki Salib Suci Bandung. Alasan pengambilan uji kasus ini adalah penulis akan mengajukan sebuah alternatif solusi selain susunan bahan kebutuhan pokok yang sudah dirancang oleh pihak pengurus. Susunan yang sudah ada akan dijadikan sebuah perbandingan untuk solusi yang dihasilkan dari penerapan *dynamic programming*.

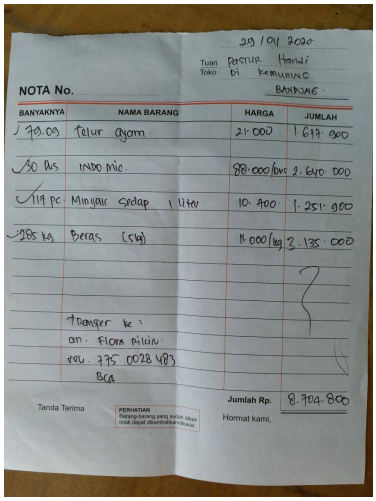
A. Pengumpulan Data

Dari hasil wawancara, didapat sebuah susunan kebutuhan pokok telah dirancang oleh pihak pengurus. Setiap paket kebutuhan pokok akan dibagikan kepada 1 keluarga. Setiap paket memiliki budget sebesar Rp. 100.000,00. Berikut adalah isi dari setiap paket kebutuhan pokok:

Barang	Jumlah
Beras	5 kg
Minyak	1 liter
Mie Instan	10 bungkus
Telur	10 butir

Tabel 1. Paket Kebutuhan Pokok yang sedang dipakai (Sumber: Wawancara)

Penulis juga mendapatkan kuitansi pembelian barang yang jadi bahan kebutuhan pokok.



Gambar 2. Bukti kuitansi pembelian bahan kebutuhan pokok (Sumber: wawancara)

Bila disusun sesuai kuitansi, maka setiap paket kebutuhan pokok bernilai Rp. 99.366,00. Nilai ini akan menjadi perbandingan dengan hasil perhitungan program.

Dari kuitansi yang didapatkan dan hasil wawancara, disusun sebuah daftar barang yang sudah disusun berdasarkan unit tiap barang, harga yang telah mengalami pembulatan, dan urutan prioritasnya sebagai berikut:

	Prioritas	Barang	Satuan	Harga
A	1	Beras	1 kg	11000
B	2	Minyak	200 ml	2140
C	3	Mie Instan	2 bungkus	4400
D	4	Telur	5 butir	5833

Tabel 2. Tabel prioritas awal bahan kebutuhan pokok

B. Penyusunan Algoritma Dynamic Programming

Sebelum merancang algoritma dengan *dynamic programming* untuk menentukan susunan bahan kebutuhan pokok yang optimal, perlu didefinisikan dulu apa yang dimaksud dengan 'optimal'. Optimal pada kasus ini adalah semua barang dapat masuk ke dalam paket atau *budget* yang digunakan tersisa sesedikit mungkin.

Tahap yang akan dilalui algoritma tergantung kepada *budget* yang tersisa. Pilihan bahan akan terus dipertimbangkan selama *budget* yang tersedia cukup untuk memasukkan pilihan tersebut ke dalam paket. Kemudian, jika suatu bahan dipilih, maka prioritasnya akan turun (nilai prioritas naik), metode ini digunakan untuk menghindari susunan yang 'berat' ke arah prioritas terbesar di awal pemasukan data, yaitu beras. Maka dari

itu, program dinamis perlu modifikasi untuk mengatasi perubahan prioritas ini. Kemudian status setiap tahap adalah pilihan yang diambil dari tahap sebelumnya, maka pada iterasi pertama, status adalah kosong.

Rekursi yang disusun berdasarkan algoritma *dynamic programming* adalah sebagai berikut:

Basis:

$F(p, 0) = v_p$

$F(p, w<0) = \{\}$

Rekurens:

$F_k(p, w) = \{v_p + \{(F(\min(p), w\text{-}w_i) \mid p_p+1 \text{ if } p>0) \text{ V}$

$(\max(F(p_i, w\text{-}w_i) \mid w_i < w))\} \}$

Keterangan:

p = prioritas

w = sisa *budget*

w_i = *budget* barang ke-i

v_p = bahan kebutuhan pokok dengan prioritas p

F(p, 0) = saat budget bernilai 0, maka dipilih bahan dengan prioritas p

F(p, w<0) = saat budget kurang maka tidak dipilih barang

$F_k(p, w)$ = jika budget cukup, maka diambil barang dengan prioritas terbesar (nilai p terkecil), jika tidak cukup, diambil barang termahal

Solusi yang dihasilkan akan berupa larik dari susunan barang yang dipilih. Dari susunan ini, akan dihitung jumlah harga yang dihasilkan, jika terdapat beberapa susunan yang menghasilkan jumlah barang yang sama, hanya urutan pembangkitannya saja yang berbeda, maka hanya akan ditampilkan susunan pertama yang dikeluarkan.

C. Cara Kerja Algoritma

Pertama, algoritma akan menerima masukan berupa tabel harga dan tabel prioritas. Tabel harga akan bersifat tetap, namun tabel prioritas akan berubah setiap pemasukan barang ke dalam susunan. Selain itu, algoritma juga perlu menerima nilai total prioritas terakhir, state yang dipilih pada tahap sebelumnya, dan *budget* yang ditentukan. Nilai *budget* juga akan berubah seiring berjalannya algoritma, sebagai penentu kapan algoritma akan mencapai basis. Pada awal perhitungan, prioritas total bernilai 0 dan state bernilai kosong. Algoritma yang didesain menggunakan pendekatan program dinamis maju, berarti masukan awal budget akan bernilai sesuai dengan *budget*, bukan 0.

Berikut akan diilustrasikan cara kerja algoritma dalam beberapa tahap.

Tahap 1

w	v_p	p	A	B	C	D	Solusi Optimal		
			+1	+2	+3	+4	min(p)	i	w-w_i
100.000	[ ]	0	1	2	3	4	1	A	89.000

Tabel 3. Ilustrasi tahap 1

Tahap 2

w	v_p	p	A	B	C	D	Solusi Optimal		
			+2	+2	+3	+4	min(p)	i	w-w_i
89.000	A	1	3	3	4	5	3	A	78.000
								B	86.860

Tabel 4. Ilustrasi tahap 2

Tahap 3

w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+2	+3	+4	min(p)	i	w-w_i
78.000	A	3	6	5	6	7	5	B	75.860
w	v_p	p	A	B	C	D	Solusi Optimal		
			+2	+3	+3	+4	min(p)	I	w-w_i
86.860	B	3	5	6	6	7	5	A	75.860

Tabel 5. Ilustrasi tahap 3

Dapat dilihat pada tahap 3, kedua state menghasilkan berat sisa yang sama, prioritas dan tambahan prioritas yang sama, jumlah barang yang sama (2 x 1kg beras, 1x500ml minyak), namun dalam susunan berbeda. Untuk mempersingkat ilustrasi, maka hanya akan diambil salah satu tahap untuk dikembangkan lagi

Tahap 4

w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+3	+3	+4	min(p)	i	w-w_i
75.860	A	5	8	8	8	9	8	A	64.860
								B	73.720
								C	71.460

Tabel 6. Ilustrasi tahap 4

Tahap 5

w	v_p	p	A	B	C	D	Solusi Optimal		
			+4	+3	+3	+4	min(p)	i	w-w_i
64.860	A	8	12	11	11	12	11	B	62.720
								C	60.460
w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+4	+3	+4	min(p)	i	w-w_i
73.720	B	8	11	12	11	12	11	A	62.720
								C	69.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+3	+4	+4	min(p)	i	w-w_i
71.460	C	8	11	11	12	12	11	A	60.460
								B	69.320

Tabel 7. Ilustrasi tahap 5

Tahap 6

w	v_p	p	A	B	C	D	Solusi Optimal		
			+4	+4	+3	+4	min(p)	i	w-w_i
62.720	B	11	15	15	14	15	14	C	58.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+4	+3	+4	+4	min(p)	i	w-w_i
60.460	C	11	15	14	15	15	14	B	58.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+4	+4	+3	+4	min(p)	i	w-w_i
62.720	A	11	15	15	14	15	14	C	58.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+4	+4	+4	min(p)	i	w-w_i
69.320	C	11	14	15	15	15	14	A	58.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+4	+3	+4	+4	min(p)	i	w-w_i
60.460	A	11	15	14	15	15	14	B	58.320
w	v_p	p	A	B	C	D	Solusi Optimal		
			+3	+4	+4	+4	min(p)	i	w-w_i
69.320	B	11	14	15	15	15	14	A	58.320

Tabel 8. Ilustrasi tahap 6

Karena tidak memungkinkan untuk menulis semua tahap dalam laporan ini akibat jumlah status yang akan semakin membesar, maka akan dibuat sebuah program untuk menyelesaikan tahap-tahapnya dan mendapatkan hasil akhir.

IV. Implementasi

Sebelum algoritma *Dynamic Programming* dijalankan, di dalam program telah disediakan hasil konversi tabel prioritas menjadi dua buah larik item_prio dan item_harga. Kedua larik tersebut akan dibaca algoritma.

```
item_prio = [0, 1, 2, 3, 4]
item_harga = [0, 11000, 2140, 4400, 5833]
```

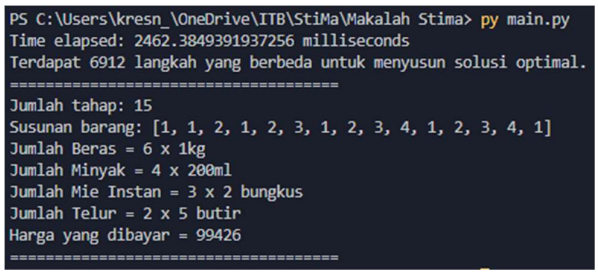
Gambar 3. Dua buah larik yang akan dibaca program

Berikut adalah implementasi rancangan algoritma yang dibuat dalam bahasa Python.

```
def calculate(budget, prio, item_prio, item_harga, last):
  # Basis
  if budget < 0:
    return [[]]
  elif budget == 0:
    return [[last]]
  # Rekurens
  else:
    opt = []
    for o in range(1, len(item_prio)):
      opt.append(prio + item_prio[o])
    choice = min(opt)
    res = []
    for i in range(len(opt)):
      if opt[i] == choice:
        result = []
        bud = budget - item_harga[i+1]
        item_prio_new = updatePrio(item_prio, i+1)
        calculate_choice = calculate(bud, i, item_prio_new
, item_harga, i+1)
        for cal in calculate_choice:
          result = []
          if last > 0:
            result.append(last)
          result = appendSequence(result, cal)
          res.append(result)
  if len(calculate_choice[0]) == 0:
    for i in range(len(opt)):
      if opt[i] != choice:
        result = []
        bud = budget - item_harga[i+1]
        item_prio_new = updatePrio(item_prio, i+1)
        calculate_choice = calculate(bud, i, item_prio_new
, item_harga, i+1)
        for cal in calculate_choice:
          result = []
          if last > 0:
            result.append(last)
          result = appendSequence(result, cal)
          res.append(result)
  ret = []
  latest = [[]]
  if len(res) > 0:
    max = 0
    for r in res:
      temp = countBudget(r, item_harga)
      if max < temp:
        max = temp
    for r in res:
      if max == countBudget(r, item_harga) and compareResu
lt(latest, r):
        latest = [r]
        ret.append(r)
    return ret
  else:
    return ret
```

V. Pengujian

Berikut adalah hasil dari pengujian program yang menerima masukan sesuai dengan tabel prioritas



Gambar 4. Hasil pengujian program (Sumber: penulis)

Dapat dilihat bahwa pengujian menghasilkan biaya paling optimal yang dapat disusun dari keempat bahan pokok adalah Rp. 99.436,00, dengan susunannya adalah sebagai berikut:

Barang	Nilai	Satuan	Total
Beras	6	1 kg	6 kg
Minyak	4	200 ml	800 ml
Mie	3	2 bungkus	6 bungkus
Telur	2	5 butir	10 butir

Tabel 9. Tabel hasil pengujian program

Berdasarkan hasil pengujian, dapat dilihat bahwa harga total dari keluaran program lebih besar dari harga total dari paket yang telah disusun oleh pihak pengurus. Perbandingan di atas membuktikan bahwa algoritma dapat menyusun paket kebutuhan pokok secara optimal, dan susunan yang telah dibuat oleh pihak pengurus dinilai masih kurang optimal dalam mencapai *budget*.

VI. KESIMPULAN

Dynamic Programming adalah sebuah metode pemecahan masalah yang bekerja dengan menguraikan solusinya, di mana salah satu contoh aplikasinya adalah dalam menyusun sebuah paket kebutuhan pokok dari sekumpulan bahan. Metode digunakan untuk menyusun sebuah paket kebutuhan pokok dengan *budget* sebesar Rp.100.000 dan sekumpulan bahan dengan keterangan:

- Beras dengan harga Rp 11.000,00 per 1 kg,
- Minyak dengan harga Rp 2.140,00 per 200 ml,
- Mie Instan dengan harga Rp 4.400,00 per 2 bungkus,
- Telur dengan harga Rp. 5833,00 per 5 butir,

Berdasarkan hasil pengujian, algoritma *Dynamic Programming* terbukti mampu menghasilkan nilai optimal di mana *budget* yang digunakan adalah Rp. 99.436,00, dengan bahan yang disusun adalah 6 kg beras, 800 ml minyak, 6 bungkus mie instan, dan 10 butir telur.

VIDEO LINK YOUTUBE

Video mengenai penjelasan algoritma dan cara kerjanya terdapat pada link berikut: https://youtu.be/Pg3UtIzEERY

UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa atas rahmat dan pertolongan-Nya selama perkuliahan di Teknik Informatika semester 4, dan bimbingan-Nya dalam mengerjakan tugas makalah IF2211 Strategi Algoritma ini sehingga makalah dapat terselesaikan dengan tepat waktu.

Kemudian, penulis mengucapkan terima kasih kepada dosen-dosen IF2211 Strategi Algoritma, Bu Nur Ula Maulidevi, Bu Masayu Leylia Khodra dan Pak Rinaldi Munir (sebagai dosen K3), atas ilmu yang telah diajarkan di kelas selama ini, dan melalui media online selama masa pandemik COVID-19 sejak pertengahan akhir semester yang mengharuskan mahasiswa menjalani kuliah dari rumah atas anjuran Pemerintah untuk dirumah saja #dirumahaja.

Lalu, penulis mengucapkan terima kasih atas bantuan keluarga, terutama orang tua, atas dukungannya selama perkuliahan dan proses penulisan makalah ini.

Penulis juga mengucapkan terima kasih kepada Bapak Randy dan pihak Gereja Katolik Paroki Salib Suci Bandung yang telah bersedia menjadi narasumber dan menyumbangkan data yang diperlukan untuk menyelesaikan makalah ini.

Penulis juga tidak lupa untuk mengucapkan terima kasih kepada teman-teman IF2018 dan terlebih kepada anggota kelas K3 yang telah menemani penulis selama masa perkuliahan, baik di kampus atau saat kuliah *online*.

REFERENSI

[1] Tim WHO, http://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/news/news/2020/3/who-announces-covid-19-outbreak-a-pandemic , diakses tanggal 1 Mei 2020, pukul 14.15 GMT +7

[2] Adrian, Kevin. https://www.alodokter.com/pentingnya-menerapkan-social-distancing-demi-mencegah-covid-19 , diakses tanggal 1 Mei 2020 pukul 14.22 GMT +7

[3] Rukka, Imansyah. https://www.kompasiana.com/imansyah_roekka/5e9d7e8bd541df4c304ad0d2/dampak-ekonomi-dan-pemberlakuan-psbb-covid-19-warga-biringkanaya-kesulitan-cari-makan , diakses tanggal 1 Mei 2020 pukul 14. 44 GMT +7

[4] Rinaldi, Munir. *Program Dinamis (Dynamic Programming)*, http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf , diakses tanggal 1 Mei 2020 pukul 20.21 GMT +7

[5] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Clifford. 2009. *Introduction to Algorithms, Third Edition*. Cambridge:Massachusetts Institute of Technology

[6] Tuteja, Sonal. https://www.geeksforgeeks.org/recursion/ , diakses tanggal 2 Mei 2020 pukul 01.49 GMT +7

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

Gregorius Jovan Kresnadi
13518135