Pemanfaatan Program Dinamis dalam Pemilihan Barang yang akan Dijual pada Toko Ritel

William Aristea Tantiono - 18215011

Program Studi Teknik Informatika,

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10, Bandung, Jawa Barat, Indonesia

18215011@std.stei.itb.ac.id

***Abstrak*— Salah satu masalah yang paling banyak dihadapi dan sulit dipecahkan oleh toko ritel adalah pemilihan barang yang harus dipajang dan dijual. Hal tersebut dikarenakan terdapat banyak faktor yang dapat dipertimbangkan dalam pemilihan barang yang akan dijual tersebut. Faktor utama yang sering dipertimbangkan tentunya adalah jumlah keuntungan yang didapat dari suatu barang yang dipilih. Makalah ini akan membahas tentang penerapan algoritma program dinamis yang dapat digunakan untuk membantu dalam proses pemilihan barang berdasarkan besar keuntungan yang ada.**

Kata Kunci*—*maksimal; keuntungan; program dinamis; toko ecerean

I. LATAR BELAKANG

Toko ritel (*retail store*) merupakan tempat menjual barang-barang dalam jumlah kecil bagi khalayak umum [1]. Toko ritel biasa terdapat pada suatu kompleks pertokoan atau daerah yang memiliki tingkat kepadatan penduduk yang relatif tinggi. Pada suatu toko ritel barang yang akan dijual umumnya diletakkan pada suatu etalase atau katalog agar pengunjung dapat memilih barang yang akan mereka beli, baik secara langsung (pasar swalayan) ataupun tidak langsung (misal: toko bangunan).

Toko ritel memiliki ukuran yang bervariasi. Toko ritel berukuran kecil biasanya terdapat di pasar atau daerah perumahan. Toko ritel yang berukuran besar, yang dapat disebut juga perusahaan ritel, biasa terdapat pada suatu mal. Beberapa perushaan ritel besar yang terdapat di Indonesia antara lain: Alfamart, Indomart, dan Carrefour.

Toko ritel umumnya mendapat barang yang akan mereka jual dari satu atau beberapa *sales* distributor yang masing – masing bertanggungjawab atas suatu merek atau perusahaan dagang tertentu. Umumnya terdapat dua cara untuk mendapatkan barang dari seorang *sales* distributor, yaitu dengan cara membelinya terlebih dahulu, atau secara cuma – cuma. Pada cara yang kedua, bila terdapat transaksi jual beli yang dilakukan konsumen akhir, barulah proses pembayaran dilakukan kepada *sales* distributor.

Seiring dengan perkembangan zaman, kebutuhan masyarakat semakin meningkat, dan toko ritel berusaha semampu mereka untuk memenuhi kebutuhan tersebut. Pemenuhan kebutuhan tersebut dapat dilakukan dengan berbagai macam cara, baik dengan memperbaiki tampilan toko, menambah fasilitas seperti pendingin ruangan, atau mengembangkan sistem pelayanan pelanggan. Kemampuan suatu toko dalam menangkap dan bertindak akan kebutuhan dalam masyarakat sangatlah penting dan dapat menentukan keberlangsungan toko di masa yang akan datang.

Salah satu cara yang paling mudah untuk memenuhi kebutuhan konsumen adalah dengan menambah pilihan barang yang terdapat pada toko guna memenuhi permintaan barang (*demand*) konsumen. Hal ini dilakukan dengan harapan semakin banyak varian barang yang dijual, semakin tinggi jumlah transaksi yang dapat dilakukan dan keuntungan yang didapat. Akan tetapi pendekatan ini tentunya membutuhkan perhitungan yang seksama dikarenakan dibutuhkan sumber daya berupa uang, waktu, dan tempat yang cukup dalam pelaksanaannya.

Di lain sisi, bidang keilmuan IT berkembang semakin pesat setiap harinya. Fenomena ini dapat diamati dari semakin besarnya perannya dalam perusahaan dan kehidupan masyarakat umum. Pada lingkungan perusahaan, perkembangan tersebut dapat kita lihat dari pengingkatan efisiensi komunikasi dengan menggunakan email, maupun penggunaan pembelajaran mesin dan sistem ahli dalam menentukan produk yang akan digunakan.

Melihat perkembangan tersebut, penulis merasa bahwa bukanlah hal yang sulit untuk membuat suatu sistem penyaranan barang sederhana untuk toko ritel. Sistem ini akan dapat digunakan untuk menyelesaikan permasalahan pada toko ritel yang telah dijabarkan. Pertama, sistem akan menerima jumlah uang yang dapat dihabiskan dan daftar harga jual-beli barang. Kemudian, sistem akan mengeluarkan saran kumpulan barang yang memiliki tingkat keuntungan yang optimal. Sistem ini memiliki permasalahn yang senada dengan permasalahan *knapsack* yang telah sering ditemui pada bidang keilmuan IT.

II. LANDASAN TEORI

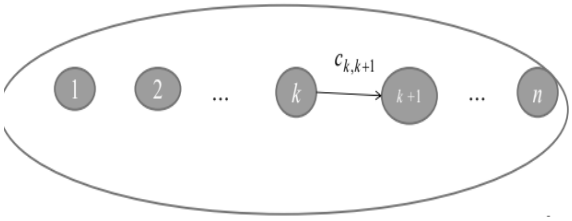
A. *Program Dinamis*

Program dinamis (*dynamic programming*) merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan [2].

Karakteristik penyelesaian persoalan dengan program dinamis antara lain [2]:

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang dharus dipertimbangkan pada suatu tahap

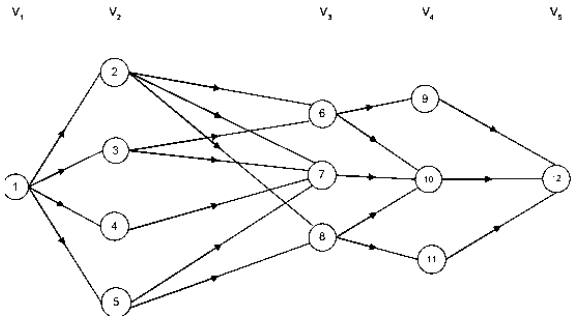
Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan prinsip optimalitas. Prinsip optimalitas berbunyi: "*Jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal*". Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap *k* ke tahap *k+1*, kita dapat menggunakan hasil optimal dari tahap *k* tanpa harus kembali ke tahap awal. Ongkos pada tahap k+1 = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap k+1)



Gambar 1 Ilustrasi propagasi hasil program dinamis

Karakteristik persoalan program dinamis antara lain:

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 tahap tersebut. Status dan tahap dapat digambarkan dengan menggunakan graf multitahap (*multistage graph*), dimana *V* menyatakan tahap dan lingkaran yang ada menyatakan status (*state*)



Gambar 2 Gambar graf multitahap [2]

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 ongkos pada tahap tersebut.
6. Keputusan terabik 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 terabik untuk setiap status pada tahap *k+1*.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Untuk menyelesaikan masalah dengan menggunakan program dinamis, terdapat dua pendekatan berbeda yang biasa digunakan, yaitu:

1. Program dinamis maju (*forward* atau *up-down*)
   Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap n.
2. Program dinamis mundur (*backward* atau *bottom-up*)
   Program dinamis bergerak mulai dari tahap *n*, terus mundur ke tahap *n-1*, *n-2*, dan seterusnya sampai tahap 1.

Langkah – langkah pengembangan algoritma program dinamis:

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

Program dinamis memiliki kekurangan, yaitu solusi mungkin saja terjebak dalam minimum lokal. Masalah ini timbul dikarenakan program dinamis tidak mencoba setiap cabang solusi, sebagaimana pada *brute force*, tetapi program dinamis membangun suatu solusi besar (n) dari solusi yang lebih kecil (n-1). Walaupun demikian, penulis merasa bahwa solusi ini tepat dikarenakan keterbatasan waktu, data dan infrastruktur yang ada.

B. *Permasalahan Knapsack*

Pada permasalahan *knapsack*, diberikan *n* buah objek dan sebuah *knapsack* (kantung) dengan kapasitas bobot *K*. Setiap objek memiliki bobot (*weight*) *w* dan keuntungan (*profit*) *p*. Akan dicari objek - objek yang dimasukkan ke dalam *knapsack* sedemikian sehingga diperoleh keuntungan yang maksimal. Total bobot objek yang dimasukkan ke dalam *knapsack* tidak boleh melebihi kapasitas *knapsack*.

Pada penyelesaian persoalan dengan menggunakan program dinamis:

1. Tahap (*k*) adalah proses memasukukan barang ke dalam *knapsack* (karung)
2. Status (*y*) menyatakan kapasitas muat karung yang tersisa setelah memasukkan barang pada tahap sebelumnya.

Bila digunakan program dinamis, dari tahap ke-1, kita masukkan objek ke-1 ke dalam karung untuk setiap satuan kapasitas karung sampai batas kapasitas maksimumnya. Bila kapasitas karung adalah bilangan bulat, pendekatan ini praktis. Misalkan ketika memasukkan objek pada tahap *k*, kapasitas muat karung sekarang adalah y-w_k. Untuk mengisi kapasitas sisanya, kita menerapkan prinsip optimalitas dengan mengacu pada nilai opitmum dari tahap sebelumnya untuk kapasitas sisa. Selanjutnya, kita bandingkan nilai keuntungan dari objek pada tahap *k* (yaitu *pk*) plus nilai $f_{k-1}(y-w_k)$ dengan keuntungan pengisian hanya *k-1* macam objek, $f_{k-1}(y)$

III. RUMUSAN MASALAH

Sistem akan memecahkan permasalahan berupa pemilihan barang yang akan dijual berdasarkan harga jual-beli barang. Sistem akan menerima jumlah uang yang ingin dibelanjakan oleh sebuah toko ritel serta daftar transaksi jual dan beli barang yang ada dalam bentuk sebuah *file .xls*. Sistem yang ada nantinya akan mengeluarkan saran daftar barang dibeli.

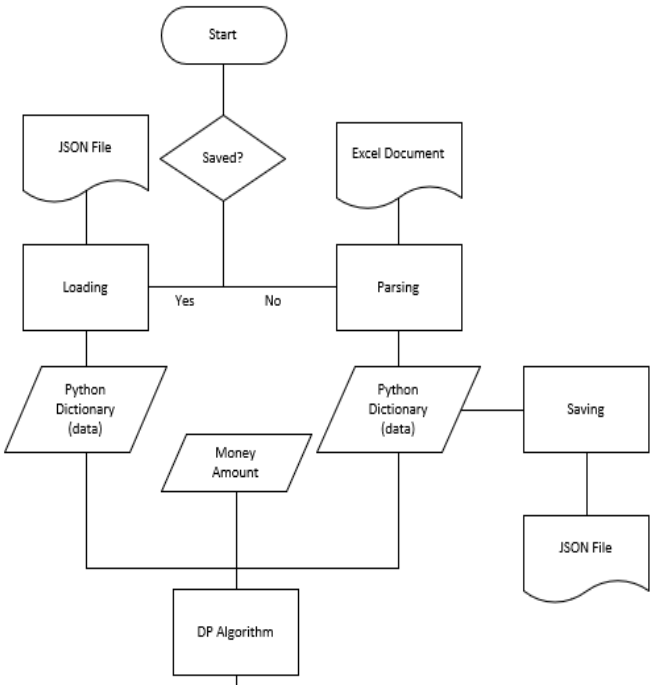
Permasalahan ini akan dipecahkan dengan menggunakan pendekatan *integer 1/0 knapsack*. Pendekatan ini digunakan agar toko ritel dapat memiliki lebih banyak variasi barang, dan tidak hanya menjual satu jenis barang saja. Permasalahan ini akan dipecahkan dengan menggunakan algoritma program dinamis untuk menghemat waktu (dibandingkan *brute force*) dan menghasilkan hasil yang cukup reliabel. Algoritma ini akan dikembangkan dengan menggunakan bahasa pemrograman Python. Penulis juga akan menggunakan *library pandas* untuk mengolah data berbentuk *.xls* menjadi *dictionary* Python yang dapat diproses [3].

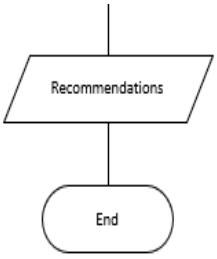
Dikarenakan data yang ada sebenarnya adalah data transaksi keuangansecara keseluruhan, data yang ada bercampur dengan keperluan – keperluan perusahaan lainnya selain pembelian dan penjualan barang (misal adanya retur atau penulisan ongkos kirim).

	NO	TGL	CUSTOMER		NAMA BARANG	HARGA@
3	1	05-Jan	abc	50	rol kertas sony upp 110 hg	170.000
4	2	06-Jan	def	1	unit sterilisator fortune 2 pintu	1.250.000
5				6	tube strip accu cek aktif	225.000
6				1	okir indah kargo	285.000
7				1	okir	28.000
8	3	06-Jan	ghi	1	box pipette led vacuette 200's	750.000
9				1	pak kartu golda	20.000
10				1	okir	35.000
11	4	06-Jan	jkl	1	unit sterilisator fortune 2 pintu	1.250.000
12	5	07-Jan	mno	1	unit termometer citizen ct 810	375.000
13				1	okir	9.000
14	6	07-Jan	pqr	2	pcs kotak p3k maspion 28x28x10,5cm	80.000
15				1	pcs bola pipet filler	90.000
16				1	pcs termometer ruangan kayu	15.000
17				1	box tabung reaksi pyrex 12x75 (100's)	500.000
18				3	pcs rak tabung reaksi 50 hole	47.500
19				1	unit rotator gemmy vrn 200	3.600.000
20				1	okir	50.000
21	7	07-Jan	stu	2	unit linen trolly tec 084	4.200.000
22				2	unit trolly makan 8 tingkat tec 080	5.500.000
23				1	unit rotator gemmy vrn 360	3.850.000
24				1	okir 4 coli trolly makan(2), linen(1)roda(1)	1.669.500

Gambar 3 Contoh data transaksi PT Sanna Medika

Oleh karena itu, penulis harus melakukan ‘pembersihan’ terlebih dahulu untuk mendapatkan data yang diinginkan. Data inilah yang nantinya akan dimasukkan ke dalam algoritma program dinamis. Berikut adalah gambaran pemrosesan data excel yang ada:





Gambar 4 Flow chart pemrosesan data

Berikut adalah algoritma *parsing* dan penyimpanan yang dibuat oleh penulis dalam proses diatas:

```
def excel_parser(folder_name):
    # will output a dictionaries with:
    # key = item-name
    # values = dictionary
    ## key = sell-price; buy-price; sell-quantity; buyers

    os.chdir(folder_name)
    file_names = os.listdir('.')

    files = []
    sheets = []
    for name in file_names:
        # Read excels files
        file = pandas.ExcelFile(name)
        files.append(file)

        # Read sheet files
        sheet_names = file.sheet_names
        sheet = file.parse(sheet_names[0])
        sheets.append(sheet)

    os.chdir('..')
    #"""
    items = {}
    column_names = list(sheets[0])
    for sheet in sheets:
        last_buyer=""
        for i in range(sheet.shape[0]):
            name = sheet.at[i, column_names[4]]
            quantity = sheet.at[i, column_names[3]] # sell quantity
            sell_p = sheet.at[i, column_names[5]]
            buyer = sheet.at[i, column_names[2]]
            valid = True
            # Penyaringan sel kodong dan tidak valid
            ## harga atau jumlah <=0 biasanya merupakan retur barang atau
            if (pandas.isna(name)): valid = False
            elif (pandas.isna(quantity)) or quantity<=0 : valid = False
            elif (pandas.isna(sell_p)) or sell_p<=0: valid = False

            # kalau kolom buyer kosong, lihat buyer terakhir yang membeli
            # (biasa dalam satu pembelian nama buyer tidak ditulis lagi)
            if pandas.isna(buyer):
                buyer=last_buyer
            else:
                last_buyer=buyer

            # Pemasukan ke dictionary
            if valid: # jika cell tidak kosong
                if name not in items: # jika item belum pernah ada
                    item_info = {}
                    item_info['sell-price'] = sell_p
                    item_info['buy-price'] = sell_p*100/110
                    item_info['quantity'] = quantity
                    item_info['buyers'] = [buyer]
                else:
                    item_info = items[name]
                    if buyer not in item_info['buyers']:
                        item_info['buyers'].append(buyer)
                        item_info['quantity'] += quantity
                items[name] = item_info
    #"""

    print(len(items))

    return items
def write_parsed_data(items_dict, file_name):
    with open(file_name, 'w') as file:
        file.write(json.dumps(items_dict))
def read_parsed_data(file_name):
    with open(file_name, 'r') as file:
        items_dict = json.load(file)
    return items_dict
```

Gambar 5 Algoritma parsing dan penyimpanan file

Penulis bekerja sama dengan perusahaan ritel PT Sanna Medika untuk mendapatkan data – data ini. PT Sanna Medika merupakan sebuah toko ritel yang berpusat di Jakarta. PT Sanna Medika menjual barang – barang yang berhubungan dengan alat – alat kesehatan, yang biasa dibeli untuk keperluan sekolah dan laboratorium.

IV. PEMECAHAN MASALAH

Oleh dikarenakan terdapatnya sangat banyak varietas barang pada data yang diterima, untuk menghemat waktu dan tempat, penulis akan mengilustrasikan pemecahan masalah dengan menggunakan 4 jenis barang yang terdapat dalam katalog barang PT Sanna Medika. Berikut adalah barang – barang tersebut:

Barang ke-i	**Nama**	***wi***	***pi***
1	unit sterilisator fortune 2 pintu	1136363	113636
2	unit centrifuge 12 hole nesco	1636363	163636

3	unit autoclave 18 liter nesco	2818181	281818
4	unit meja gynecolog	1454545	145454

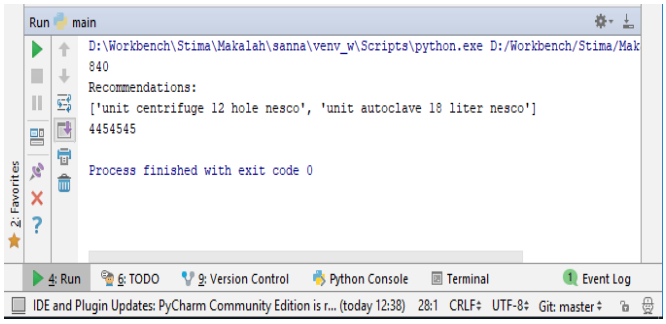
Tabel 1 Tabel contoh barang yang terdapat pada PT Sanna Medika

Seperti yang dapat diamati pada tabel 1, barang yang dijual cendrung memiliki harga di atas Rp100.000,00. Bila pada setiap tahapan peningkatan *cost* hanya sebesar Rp1,00, maka untuk nominasi sebesar Rp5.000.000,00 akan membutuhkan waktu dan tempat yang sangat besar. Oleh karena itu akan lebih baik bila digunakan skala yang merupakan FPB (atau setidaknya mewakilkan) semua jenis barang yang terdapat pada daftar barang. Pada data sebenarnya, nominasi terkecil adalah Rp1.400,00 dan kedua terkecil adalah Rp4.600,00, sehingga skala yang cocok, menurut penulis, adalah Rp4.000,00. Sedangkan untuk data testing ini, menurut pengamatan penulis, skala yang cocok adalah Rp100.000,00. Berikut adalah potongan kode Python yang menggambarkan maksud keseluruhan penulis:

```
def dp_knapsack(items_dict, total_money):
    # Testing
    # for trial purposes use 5 item only
    testing_keys = [
        'unit sterilisator fortune 2 pintu',
        'unit centrifuge 12 hole nesco',
        'unit autoclave 18 liter nesco',
        'unit meja gynecolog tec 102',
    ]

    keys_list = testing_keys
    temp_items_dict = {}
    for key in testing_keys:
        temp_items_dict[key] = items_dict[key]
    items_dict=temp_items_dict
    #"""

    # Real
    # keys_list = items_dict.keys()

    SCALE = 100000 # dari harga barang kedua terkecil, yaitu Rp4500,00
    BRACKET_SIZE = int(math.ceil(total_money/SCALE))+1

    bitmap = [0]*len(items_dict)
    bracket = [0]*BRACKET_SIZE # f0(y) di slide
    bracket_bitmap = [bitmap]*BRACKET_SIZE # f1(y) di slide
    #"""
    for j in range(len(keys_list)):
        item = items_dict[keys_list[j]]
        spendable = 0
        cost = item['buy-price']/SCALE
        revenue = (item['sell-price']-item['buy-price'])/SCALE
        new_bracket = copy.deepcopy(bracket)  # f0(y) di slide
        new_bracket_bitmap = copy.deepcopy(bracket_bitmap)  # f1(y) di slide

        for i in range(BRACKET_SIZE):
            if spendable<int(math.ceil(cost)):
                x = -9999
            else:
                x = bracket[spendable-int(math.ceil(cost))] + revenue
                bitmap = copy.deepcopy(bracket_bitmap[spendable-int(math.ceil(cos
                bitmap[j] = 1

            if bracket[i]<x: # max(f0(y), p1+f0(y-w1))
                new_bracket[i]=x
                new_bracket_bitmap[i]=bitmap
            spendable+=1
        bracket = new_bracket
        bracket_bitmap = new_bracket_bitmap
    #"""

    total_list = []
    for bmp in bracket_bitmap:
        total = 0
        for i in range(len(bmp)):
            if bmp[i]==1 : total+=items_dict[keys_list[i]]['buy-price']
        total_list.append(int(math.floor(total)))


    max_total = max(total_list)
    max_index = total_list.index(max(total_list))
    max_bitmap = bracket_bitmap[max_index]

    recommendations = []
    for i in range(len(max_bitmap)):
        if max_bitmap[i]==1 : recommendations.append(keys_list[i])

    return recommendations, max_total
def main():
    # parse the excels
    FOLDER_NAME = 'penjualan-sanna'
    JSON_NAME = 'penjualan_dictionary.json'

    items_dict = {}
    items_dict = excel_parser(FOLDER_NAME)
    write_parsed_data(items_dict, JSON_NAME)

    items_dict = read_parsed_data(JSON_NAME)

    # Knapsack DP
    MONEY = 5000000
    recommendations = dp_knapsack(items_dict, MONEY)

    # print out recommendations
    print("Recommendations:")
    for rec in recommendations:
        print(rec)

if __name__=="__main__":
    main()
```

Gambar 6 Potongan gambar implementasi algoritma

Berikut adalah keluaran dari algoritma di atas:



Gambar 7 Keluaran kode

V. ANALISIS

Hasil keluaran yang dihasilkan oleh algoritma pemrograman dinamik cukup dapat diandalkan, terutama bila dibandingkan dengan algoritma *brute force*. Akan tetapi, hasil pengamatan penulis, algoritma program dinamis membutuhkan waktu dan tempat (memori) yang besar untuk dijalankan. Hal ini dikarenakan algoritma ini menyimpan semua tahapan yang ada.

VI. KESIMPULAN

Algoritma program dinamis dapat digunakan untuk mencari barang yang dapat menghasilkan keuntungan maksimum dalam suatu perusahaan. Akan tetapi, perlu diketahui bahwa banyak pendekatan lain yang lebih baik yang dapat digunakan oleh suatu perusahaan dalam menentukan barang yang akan dipilih. Salah satu pendekatan alternatif yang sering digunakan adalah pembelajaran mesin.

UCAPAN TERIMA KASIH

Pertama, penulis ingin mengucapkan puji dan syukur kepada Tuhan YME, atas bantuanNya dalam penyusunan makalah ini. Selanjutnya, penulis ingin mengucapkan terima kasih kepada para dosen pengajar matakuliah IF2211, yaitu Dr. Masayu Leylia Khodra S.T., M.T., Dr. Nur Ulfa Maulidevi S.T., M.Sc., Dr. Ir. Rinaldi Munir, M.T., serta para asisten pengampu mata kuliah atas bimbingan yang telah diberikan pada mata kuliah ini. Selain itu, penulis juga ingin berterima kasih kepada PT Sanna Medika atas bantuannya dalam menyediakan data penjualan yang komprehensif, yang digunakan oleh penulis sebagai dasar analisis dalam makalah ini. Terakhir, penulis juga tidak lupa berterima kasih kepada orang tua dan teman – teman penulis atas bantuan, kritik, dan saran yang diberikan dalam penyusunan makalah ini.

REFERENSI

[1] https://kbbi.kemdikbud.go.id/
[2] Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma. Bandung: ITB, 2007.
[3] Willems, Karlijn. “Python Excel Tutorial: The Definitive Guide”. Datacamp. Jan. 31, 2017. [Online]. Available: https://www.datacamp.com/community/tutorials/python-excel-tutorial

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, 11 Mei 2018

William Aristea Tantiono / 18215011