JURNAL ILMIAH FLASH
Vol. 10 No. 1, Halaman: 17 - 25
Juni 2024

SISTEM PENUNJANG KEPUTUSAN: PEMILIHAN LABA OPTIMAL DALAM KETERBATASAN ANGGARAN MELALUI ALGORITMA *KNAPSACK* DENGAN *PYTHON*

Herryansyah
Universitas Bina Sarana Informatika
E-mail: herryansyah.hrr@bsi.ac.id

Abstrak

Penelitian kali ini bertujuan untuk membuat program dengan bahasa *Python*, yang mana fungsi dari program tersebut mampu mengoptimalkan pengeluaran berdasarkan keterbatasan anggaran, demi memperoleh nilai laba optimal dengan menerapkan algoritma *Knapsack*. Algoritma *Knapsack* merupakan metode matematis yang digunakan untuk memecahkan masalah alokasi sumber daya terbatas dengan mencari kombinasi item yang memberikan hasil terbaik tanpa melampaui batas anggaran yang ada. Dalam penelitian ini, dilakukan implementasi algoritma tersebut menggunakan *Python* untuk memodelkan suatu situasi dimana *user* dihadapkan pada masalah pemilihan item manakah yang sekiranya mampu memberikan laba atau nilai optimal dengan mempertimbangkan keterbatasan anggaran yang dimiliki. Hasil dari penelitian ini selain menyajikan langkah-langkah pembuatan program, juga menunjukkan efektivitas algoritma *Knapsack* dalam mengoptimalkan alokasi anggaran yang terbatas serta menawarkan solusi praktis untuk pengambilan keputusan dalam kondisi keuangan yang serupa melalui program yang telah dibuat dengan *Python*.

Kata kunci: Sistem Penunjang Keputusan, SPK, *Decission Support System*, *DSS*, *Knapsack*, *Python,* Laba

PENDAHULUAN

Aktivitas pengalokasian dana merupakan aspek penting dalam aktivitas bisnis yang sehat berkualitas. Dalam situasi keterbatasan anggaran, menemukan cara optimal untuk memilih kombinasi investasi yang mampu memberikan laba maksimum menjadi suatu tantangan tersendiri bagi pelaku bisnis, baik dalam *level* kecil ataupun *level* global sekalipun. Penelitian ini berfokus pada pengembangan sistem penunjang keputusan yang menerapkan algoritma *Knapsack* dengan menggunakan bahasa pemrograman *Python*.

Algoritma *Knapsack* telah terbukti sebagai metode matematis yang sederhana namun efektif dalam menyelesaikan masalah alokasi sumber daya terbatas. Dalam penelitian kali ini, Penulis mencoba menerapkan algoritma ini dalam sebuah kode pemrograman *Python*. Yang mana nanti akan hadir sebuah program *user-friendly* yang mampu membantu untuk menemukan kombinasi pemilihan investasi yang menghasilkan laba optimal dengan keterbatasan anggaran yang ada.

Diharapkan bahwa implementasi algoritma *Knapsack* dalam lingkungan bahasa pemrograman *Python* dalam bentuk program siap pakai akan mampu memberikan solusi efisien dan praktis untuk menangani tantangan dari kasus-kasus serupa. Dalam bab-bab selanjutnya, akan diuraikan langkah-langkah, metodologi, serta hasil dari penelitian ini untuk memberikan wawasan mendalam tentang proses pengembangan sistem penunjang keputusan ini.

METODE PENELITIAN

Penelitian ini melibatkan serangkaian tahapan penting yang harus dilalui. Tahapan tersebut meliputi:

1. Teknik Pengumpulan Data
   a. Pengumpulan dan analisis data
      Penulis mengumpulkan data dari sejumlah sumber terkait dalam menentukan variabel-varibel terkait seperti *budget*, biaya, harga jual, laba dan variabel lainnya yang terkait untuk dianalisa sesuai kebutuhan penelitian.
   b. Wawancara
      Wawancara dengan sejumlah pelaku bisnis guna mendapatkan wawasan langsung terkait strategi bisnis dan faktor-faktor penting lainnya yang mereka terapkan dalam memperoleh keuntungan. Serta memperoleh informasi terkait apa saja kebutuhan mereka jika nantinya ada suatu aplikasi yang bisa membantu dalam menjalani bisnis mereka.

c. Studi Literatur dan Referensi
Tahap ini melibatkan beberapa literatur, jurnal, serta sumber referensi terkait algoritma *Knapsack* dalam pemecahan masalah keterbatasan anggaran. Penulis mempelajari dan menganalisis penelitian terdahulu guna memperoleh wawasan mendalam terkait teknik optimalisasi dengan algoritma *Knapsack*.

2. Model Pengembangan Sistem
Pengembangan sistem pada penelitian kali ini, digunakan metode *waterfall*. Adapun langkah-langkah dalam model ini [1]:
a. Analisa kebutuhan sistem
Proses mendefinisikan kebutuhan yang harus dipenuhi oleh program Sistem Penunjang Keputusan yang akan dibangun.
b. Desain
Tahapan yang terdiri dari merancang tampilan program, bagaimana alur data yang diproses dalam program, prosedur pengetikan kode program agar sesuai dengan yang diinginkan oleh *user*
c. *Code Generation*
Tahap pengetikan kode program dengan bahasa pemrograman *Python* menggunakan *Compiler Spyder*.
d. Testing
Tahap pengujian program, untuk mengetahui apakah terdapat *error* saat dijalankan. Yang nantinya akan dilakukan perbaikan.
e. Support
Tiap perangkat lunak yang baik tentu ada proses *maintenance* dan proses *updating*-nya. Adapun proses-proses tersebut diperlukan demi memenuhi kebutuhan *user* yang sewaktu-waktu bisa berubah dan berkembang.

3. Sistem Penunjang Keputusan
Sistem Pendukung Keputusan (*Decision Support System*) adalah sistem berbasis komputer yang interaktif dalan membantu pengambil keputusan dengan me-manfaatkan data dan model untuk menyelesaikan masalah-masalah yang tak terstruktur. Sistem pendukung keputusan juga merupakan suatu sistem informasi berbasis komputer yang menghasilkan berbagai alternatif keputusan untuk membantu manajemen dalam menangani berbagai permasalahan yang terstruktur ataupun tidak terstruktur dengan menggunakan data atau objek [2].

4. *Python*
Python merupakan salah satu bahasa pemrograman yang banyak digunakan oleh perusahaaan besar maupun para *developer* untuk mengembangkan berbagai macam aplikasi berbasis *desktop*, *web* dan *mobile*. *Python* diciptakan oleh Guido van Rossum di Belanda pada tahun 1990. Van Rossum mengembangkan *Python* sebagai hobi, kemudian *Python* menjadi bahasa pemrograman yang dipakai secara luas dalam industri dan pendidikan karena sederhana, ringkas, sintaks intuitif dan memiliki pustaka yang luas [3].

5. Algoritma *Knapsack*
Knapsack Problem merupakan salah satu dari persoalan klasik yang banyak ditemukan dalam literatur-literatur lama dan hingga kini permasalahan tersebut masih sering ditemukan dalam kehidupan sehari-hari [4].
Contoh nyata dari *Knapsack Problem* ini misalnya, jika ada seorang Pebisnis ingin membeli barang untuk dijual kembali dengan tujuan untuk memperoleh keuntungan. Tentu Pebisnis ini memiliki anggaran (*budget*) dengan besaran tertentu. Yang menjadi masalah adalah jika besaran *budget* tersebut terbatas, sehingga ia tidak bisa membeli barang sesuka hatinya. Dan berakibat ia memiliki keterbatasan dalam memilih barang yang akan dibeli dengan pertimbangan total harga barang yang dibeli tidak boleh melebihi *budget* yang ia miliki. Pada situasi ini akan melahirkan banyak alternatif dalam pemiihan barang. Dan algoritma *Knapsack* menawarkan solusi matematis dalam memperhitungkan barang apa saja yang akan dipilih demi mendapatkan laba yang optimal jika barang tersebut dijual kembali.
Adapun pendekatan-pendekatan yang digunakan pada algoritma tersebut dalam mengolah data adalah:
a. Greedy By Weight (GBW)
Perhitungan laba dengan berfokus pada beban (*Weight*) atau nilai yang harus dikeluarkan atau dibayar atau dikorbankan. Dengan kata lain mencari nilai manfaat, bedasarkan *Weight* terkecil. Jadi semakin kecil *Weight* suatu barang maka akan semakin diprioritaskan.
b. Greedy By Profit (GBP)
Perhitungan laba dengan berfokus pada keuntungan (*Profit*) atau nilai yang akan diperoleh. Dengan kata lain

mencari nilai manfaat, bedasarkan *Profit* terbesar. Jadi semakin besar *Profit* suatu barang maka akan semakin diprioritaskan.

c. *Greedy By Density (GBD)*
Perhitungan laba dengan berfokus pada rasio perbadingan *Profit* dibagi *Weight* (R=P/W). Dengan kata lain mencari nilai manfaat, bedasarkan nilai Rasio terbesar. Jadi semakin besar Rasio suatu barang maka akan semakin diprioritaskan.

Menurut Yulikuspartono [5] dalam beberapa literatur, istilah lain dari fungsi tujuan dapat disebut sebagai fungsi utama atau juga fungsi objektif, yaitu fungsi yang menjadi penyelesaian permasalahan dengan mendapatkan solusi yang optimal. Solusi yang dimaksud adalah menemukan nilai/profit yang maksimal untuk jumlah item yang dibayarkan oleh *budget* sehingga bisa memenuhi sesuai besaran *budget*.
Adapun Fungsi Tujuan Maksimum adalah :

$$\sum_{i=1}^{n} Pi.Xi \quad (1)$$

Fungsi pembatas adalah fungsi subyektif yang bertujuan untuk memberikan batas maksimal dari setiap *item* untuk bisa dibayarkan oleh *budget*, sehingga alokasinya tidak melebihi dari jumlah *budget*.
Fungsi Pembatas

$$\sum_{i=1}^{n} Wi.Xi \leq M \quad (2)$$

dimana $0 \leq Xi \leq 1$; $Pi > 0$; $Wi > 0$
Dengan kata lain kita harus mencari nilai probabilitas antara 0 dan 1, $0 \leq Xi \leq 1$ untuk setiap *item*, agar diperoleh solusi optimal untuk laba tertinggi.

Dengan berdasar tahapan-tahapan diatas, maka penelitian ini dapat berlangsung dengan lebih sistematis dan terstruktur.

HASIL DAN PEMBAHASAN

Pada bab ini akan dijabarkan tahapan-tahapan pengembangan sistem.

a. Analisa Kebutuhan Sistem

Berdasarkan data yang berhasil dikumpulkan, kebutuhan user terhadap sistem yang akan dibangun adalah:

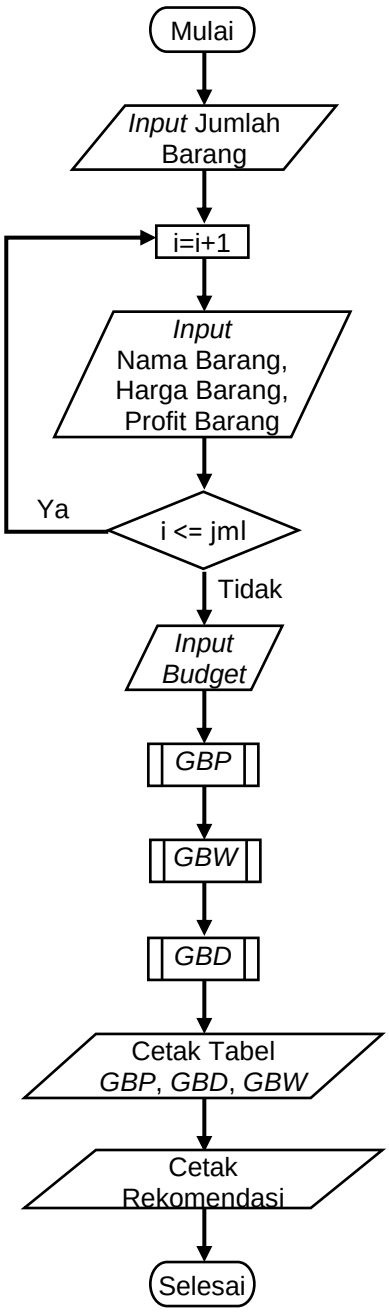
1. *User* membutuhkan sistem yang sederhana dan mudah digunakan.
2. *User* membutuhkan sistem yang dapat membantu memberikan pilihan dalam pengambilan keputusan.
3. Aplikasi bisa dijalankan tanpa membutuhkan spesifikasi perangkat yang tinggi.

b. Desain

Berdasarkan hasil yang diperoleh dari poin Analisa Kebutuhan Sistem dan dipadukan dengan bagaimana algoritma *Knapsack* itu bekerja, maka *pseudocode* dan *flowchart* logika program secara garis besar adalah sebagai berikut:

- *Input* jumlah barang.
- *Input* nama, harga, profit barang. Lakukan berulang sebanyak jumlah barang.
- *Input budget*
- Proses dengan metode *Greedy By Profit*
- Proses dengan metode *Greedy By Weight*
- Proses dengan metode *Greedy By Density*
- Tampilkan tabel dari semua metode
- Cetak rekomendasi berdasarkan nilai tertinggi

Urutan *pseudocode* diatas adalah tahapan penggunaan program dari awal *input* data hingga hasil pengolahan algoritma *Knapsack* ditampilkan dalam bentuk tabel dan diakhiri dengan skenario alokasi *budget* yang direkomendasikan.



Gambar 1. *Flowchart* Logika Program

c. Code Generation

Berikut adalah kode program yang ditampilkan berdasarkan urutan simbol-simbol *flowchart* diatas:

```
#BEGIN
nama=[]
W=[]
P=[]
if by=="by Herryansyah":
    acc=1
else:
    acc=0
#Modal=20

banyak=int(input("Input banyak barang : "))
for i in range(banyak):
    print("DATA KE-"+str(i+1)+" : ")
    nb=input("Nama Barang : ")
#    pn=16-len(nb)
    nama.append(nb)
    W.append(int(input("Harga Barang : ")))
    P.append(int(input("Profit Barang : ")))
    print("="*50)
Modal=int(input("Budget Anda : "))
```

Gambar 2. *Main Program*

Gambar diatas menunjukkan *listing* program dari saat peng-*input*-an banyak barang yang akan dipilih, hingga peng-*input*-an besaran *budget*.

```
#Sorting By Profit
def SortProfit(name,weight,profit):
    for i in range(len(profit)-1,0,-1):
        Max=0
        maxpro=0
        maxname=0
        for l in range(1,i+1):
            if float(profit[l])<float(profit[Max]):
                Max = l
                maxpro=l
                maxname=l
        temp = weight[i]
        temppro = profit[i]
        tempname = name[i]
        weight[i] = weight[Max]
        profit[i] = profit[maxpro]
        name[i] = name[maxname]
        weight[Max] = temp
        profit[maxpro] = temppro
        name[maxname] = tempname
```

Gambar 3. *Greedy By Profit (GBP)*

Gambar diatas adalah *listing* program yang mengurutkan data berdasarkan Profit, secara menurun dari besar ke kecil (*descending*)

```python
#Sorting By Weight
def SortWeight(name,weight,profit):
    for i in range(len(weight)-1,0,-1):
        Max=0
        maxpro=0
        maxname=0
        for l in range(1,i+1):
            if float(weight[l])>float(weight[Max]):
                Max = l
                maxpro=l
                maxname=l
        temp = weight[i]
        temppro = profit[i]
        tempname = name[i]
        weight[i] = weight[Max]
        profit[i] = profit[maxpro]
        name[i] = name[maxname]
        weight[Max] = temp
        profit[maxpro] = temppro
        name[maxname] = tempname
```

Gambar 4. *Greedy By Weight* (*GBW*)

Gambar diatas adalah *listing* program yang mengurutkan data berdasarkan *Weight*, secara berurut dari kecil ke besar (*ascending*).

```python
#Sorting By Density
def SortDensity(name,weight,profit):
    global R
    R=[]
    for i in range(len(profit)):
        R.append(float(profit[i])/float(weight[i]))
    for i in range(len(R)-1,0,-1):
        Max=0
        maxpro=0
        maxname=0
        maxR=0
        for l in range(1,i+1):
            if float(R[l])<float(R[Max]):
                Max = l
                maxpro=l
                maxname=l
                maxR=l
        temp = weight[i]
        temppro = profit[i]
        tempname = name[i]
        tempR = R[i]
        weight[i] = weight[Max]
        profit[i] = profit[maxpro]
        name[i] = name[maxname]
        R[i] = R[maxR]
        weight[Max] = temp
        profit[maxpro] = temppro
        name[maxname] = tempname
        R[maxR] = tempR
```

Gambar 5. *Greedy By Density (GBD)*

Gambar diatas adalah *listing* program yang mengurutkan data berdasarkan Rasio *Density*, secara menurun dari besar ke kecil (*descending*).

```python
def Greedy(name,weight,profit,M):
    global wixi
    global pixi
    global X
    wixi=0
    pixi=0
    X=[]
    for i in range(len(weight)):
        if M<=0:
            X.append(0)
        elif M>W[i]:
            X.append(1)
            M=M-W[i]
        else:
            X.append(float(M)/float(W[i]))
            M=0
        wixi=float(wixi)+(float(X[i])*float(W[i]))
        pixi=float(pixi)+(float(X[i])*float(P[i]))
    print("M =",Modal)
    print("="*80)
    print("{:<16}".format("NAMA BARANG")+
          "{:>16}".format("HARGA (W)")+
          "{:>16}".format("PROFIT (P)")+
          "{:>14}".format("RASIO")+
          "{:>12}".format("X"))
    print("="*80)
    for i in range(len(weight)):
        print("{:<16}".format(nama[i])+
              "{:>16}".format(str(W[i]))+
              "{:>16}".format(str(P[i]))+
              "{:>14.3f}".format((float(P[i])/float(W[i])))+
              "{:>14.3f}".format(X[i]))
    print("="*80)
    print("WiXi =",wixi)
    print("PiXi =",pixi)
    selisih=pixi-wixi
    print("PiXi-WiXi = "+str(selisih)+"\n\n")
```

Gambar 6. *Listing Knapsack*

Gambar diatas adalah penerapan algoritma *Knapsack* kedalam bahasa *Python* yang nantinya akan mengolah data-data yang masuk untuk dijadikan rekomendasi pemilihan barang.

```python
SortProfit(nama,W,P)
print("{:^80}".format("GREEDY BY PROFIT"))
print("{:^80}".format("Prioritas berdasarkan PROFIT TERBESAR"))
Greedy(nama,W,P,Modal)

SortWeight(nama,W,P)
print("{:^80}".format("GREEDY BY WEIGHT"))
print("{:^80}".format("Prioritas berdasarkan WEIGHT TERKECIL"))
Greedy(nama,W,P,Modal)

SortDensity(nama,W,P)
print("{:^80}".format("GREEDY BY DENSITY"))
print("{:^80}".format("Prioritas berdasarkan RASIO TERBESAR"))
Greedy(nama,W,P,Modal)
```

Gambar 7. Cetak Tabel

Gambar diatas adalah *listing* untuk menyajikan hasil olahan data kedalam bentuk tabel dengan urutan-urutan barang berdasarkan pendekatan *Greedy By Profit*, *Greedy By Weight*, dan *Greedy By Density*.

```python
Rekomendasi.sort(key=lambda row: (row[1], row[1]),reverse=True )
print("Untuk memperoleh laba yang optimal "+
      "dari budget sebesar {}\n".format(Modal)+
      "sebaiknya anda melakukan alokasi biaya "+
      "menggunakan pendekatan {}".format(Rekomendasi[0][0]))
```

Gambar 8. Rekomendasi

Gambar diatas adalah *listing* untuk menampilkan hasil akhir dari algoritma *Knapsack*, yang berupa rekomendasi alokasi *budget* untuk memperoleh laba optimal.

d. Testing

Pada tahap ini program diuji untuk memastikan sesuai dengan tujuannya. Uji coba ini menggunakan metode *Black Box Testing* yang menguji logika program melalui kasus atau masalah yang diberikan, agar menghasilkan *output* yang sesui dengan kebutuhan. Adapun langkah-langkah pengujian adalah sebagai berikut:

Tabel 1. Pengujian *Input* Data

Event	Kode Program	*Output*	Hasil Uji
Input jumlah barang	banyak=int(input("Input banyak barang : ")) for i in range(banyak): print("DATA KE-"+str(i+ 1)+" : ")	Muncul perintah untuk meng-*input* nama, harga, profit barang secara berulang sebanyak jumlah barang yang diisi	*Valid*
Input nama, harga, profit barang	nb=input ("Nama Barang : ") nama.append(nb) W.append(int(input ("Harga Barang : "))) P.append(int(input ("Profit Barang : "))) print("="*50)	Muncul perintah untuk meng-*input* besaran *budget*	*Valid*
Input budget	Modal=int(input ("Budget Anda : "))	Menjalankan algoritma *Knapsack*	*Valid*

Tabel 2. Pengujian Pengurutan Data

Event	Kode Program	*Output*	Hasil Uji
Proses Mengurutkan data *GBP*	#Sorting By Profit def SortProfit (name,weight,profit):	Data terurut *descending* berdasarkan variabel Profit	*Valid*
Proses Mengurutkan data *GBW*	#Sorting By Weight def SortWeight (name,weight,profit):	Data terurut *ascending* berdasarkan variabel *Weight*	*Valid*
Proses Mengurutkan data *GBD*	#Sorting By Density def SortDensity (name,weight,profit):	Data terurut *descending* berdasarkan variabel Rasio	*Valid*

Tabel 3. Pengujian Algoritma *Knapsack*

Event	Kode Program	*Output*	Hasil Uji
Proses Pengolahan data dgn *Knapsack*	def Greedy (name,weight,profit,M) :	Data diolah menggunakan algoritma *Knapsack*, untuk memperoleh nilai X, ∑WiXi, dan ∑PiXi, serta nilai laba	*Valid*

Pada pengujian di Tabel 2 dan Tabel 3 tidak ada *output* berupa tampilan pada monitor, melainkan *output* hasil pengolahan data hanya tersimpan pada memory saja.

Tabel 4. Pengujian Hasil Penghitungan

Event	Kode Program	*Output*	Hasil Uji
Jalankan fungsi *Greedy* terhadap *GBP*	SortProfit (nama,W,P) print("{:^80}" .format("GREEDY BY PROFIT")) print("{:^80}" .format("Prioritas berdasarkan PROFIT TERBESAR")) Greedy (nama,W,P,Modal) Rekomendasi[0][0]=" Greedy By Profit" Rekomendasi[0][1]= selisih	Tampil pada monitor tabel *GBP* beserta besaran labanya	*Valid*
Jalankan fungsi *Greedy* terhadap *GBW*	SortWeight (nama,W,P) print("{:^80}" .format("GREEDY BY WEIGHT")) print("{:^80}" .format("Prioritas berdasarkan WEIGHT TERKECIL")) Greedy (nama,W,P,Modal) Rekomendasi[1][0]=" Greedy By Weight" Rekomendasi[1][1]= selisih	Tampil pada monitor tabel *GBW* beserta besaran labanya	*Valid*
Jalankan fungsi *Greedy* terhadap *GBD*	SortDensity (nama,W,P) print("{:^80}" .format("GREEDY BY DENSITY")) print("{:^80}" .format("Prioritas berdasarkan RASIO TERBESAR")) Greedy (nama,W,P,Modal) Rekomendasi[2][0]=" Greedy By Density" Rekomendasi[2][1]= selisih	Tampil pada monitor tabel *GBW* beserta besaran labanya	*Valid*

Tabel 5. Pengujian Tampil Rekomendasi

Event	Kode Program	*Output*	Hasil Uji
Jalankan fungsi *sort* untuk menentukan nilai laba terbesar	Rekomendasi.sort (key=lambda row: (row[1], row[1]), reverse=True) print("Untuk memperoleh laba yang optimal "+ "dari budget sebesar {}\n".format(Modal)+ "sebaiknya anda melakukan alokasi biaya "+ "menggunakan pendekatan {}" .format(Rekomendasi[0][0])):	Rekomendasi alokasi *budget* dengan laba terbesar tampil pada monitor. Bentuk rekomendasi bisa berupa pendekatan *GBP*, *GBW*, atau *GBD*	*Valid*

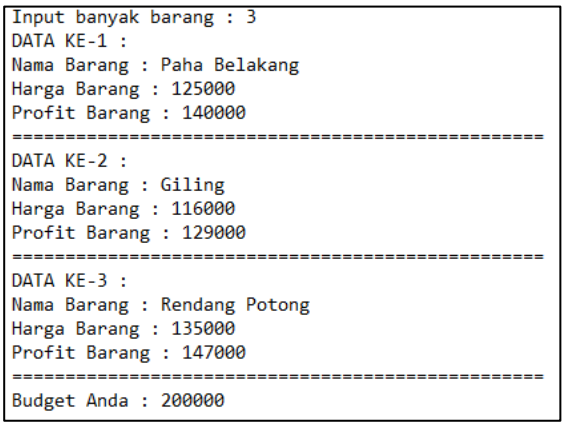
Adapun data yang digunakan untuk tahap *Black Box Testing* diatas, diambil dari contoh kasus dibawah ini.

Contoh Kasus:
Seorang pedagang ingin membeli beberapa jenis potong daging sapi untuk dijual kembali. Adapun jenis daging tersebut adalah:

1. **Daging Paha Belakang**
   Harga Beli per kg : Rp. 125.000
   Harga Jual per kg : Rp. 140.000
2. **Daging Giling**
   Harga Beli per kg : Rp. 116.000
   Harga Jual per kg : Rp. 129.000
3. **Daging Rendang Potong**
   Harga Beli per kg : Rp. 135.000
   Harga Jual per kg : Rp. 147.000

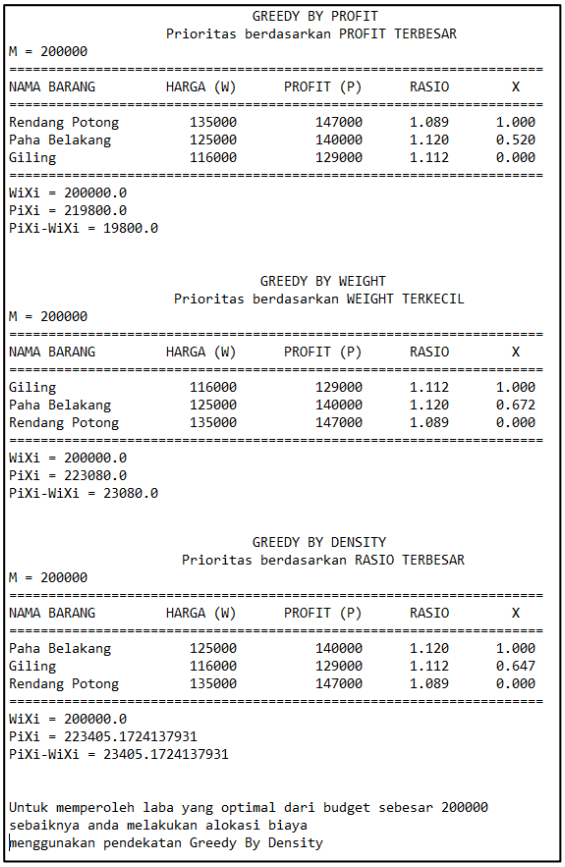
Untuk bisa membeli ketiga jenis daging potong tersebut Si Pedagang harus membayar total harga sebesar Rp. 376.000, sedangkan *budget* yang tersedia hanya sebesar Rp. 200.000 saja. Sudah tentu tidak mungkin semua jenis daging potong bisa terbeli.
Tapi, bagimanakah cara Si Pedagang mensiasati *budget* terbatas tersebut, untuk memperoleh laba yang optimal, walaupun hanya sebagian daging yang terbeli?

Data-data dari kasus diatas akan coba diolah dengan Sistem Penunjang Keputusan yang sudah dibuat. Adapun proses pemakaian programnya adalah sebagai berikut:



Gambar 9. Halaman *Input* Data

Gambar 9 menunjukkan bahwa data-data di-*input* sebanyak 3 kali, sesuai jumlah *item* yang ada pada contoh kasus. Mulai dari nama *item*, harga beli dan profit (harga jual kembali).



Gambar 10. Halaman Tabel & Rekomendasi

Gambar 10 menampilkan rincian alokasi budget jika dilakukan dengan pendekatan:

1. *Greedy By Profit*
- Daging Rendang Potong, beli 1 kg
- Daging Paha Belakang, beli hanya 0,52 kg
- Daging Giling, tidak terbeli sama sekali

Dengan skenario ini, laba yang diperoleh adalah sebesar **Rp. 19.800**

2. *Greedy By Weight*
- Daging Giling, beli 1 kg
- Daging Paha Belakang, beli hanya 0,672 kg
- Daging Rendang Potong, tidak terbeli sama sekali

Dengan skenario ini, laba yang diperoleh adalah sebesar **Rp. 23.080**

3. *Greedy By Density*
- Daging Paha Belakang, beli 1 kg
- Daging Giling, beli hanya 0,647 kg
- Daging Rendang Potong, tidak terbeli sama sekali

Dengan skenario ini, laba yang diperoleh adalah sebesar **Rp. 23.405,1724**

Jadi kesimpulannya, untuk kasus sederhana diatas, rekomendasi yang cocok untuk Si Pedagang adalah pendekatan *Greedy By Density.* Karena menghasilkan nilai laba yang paling besar yaitu **Rp. 23.405,1724.**

Adapun rekomendasi yang akan tampil bisa berbeda-beda tergantung dari kasusnya.

Dengan demikian kesimpulan yang diperoleh dari tahap *Testing* ini bahwa kode Python yang digunakan untuk menerapkan algoritma *Knapsack* sudah berfungsi dengan semestinya dan *output* yang dihasilkan sesuai dengan kebutuhan dan siap digunakan sebagai Sistem Penunjang Keputusan untuk mem-peroleh laba optimal dari keterbatasan anggaran.

e. Support

Pada tahap ini *user* belum mengalami kendala apapun. Dan sejauh ini program sudah cukup memenuhi kebutuhan mereka dalam mengoptimalkan budget bisnis mereka.

Adapun pengembangan sistem ini ke-tahap yang lebih baik, akan dibahas pada penelitian dan kesempatan berikutnya.

PENUTUP

Kesimpulan dan saran yang ingin Penulis sampaikan dalam penelitian kali adalah:

a. Penerapan Algoritma Knapsack pada bahasa pemrograman Python dapat menghasilkan sebuah program berupa Sistem Penunjang keputusan yang sederhana tapi sangat membantu dalam suatu proses bisnis. Dalam hal ini adalah mengoptimalkan perolehan laba.
b. Walaupun sederhana, Sistem Penunjang Keputusan ini sudah memenuhi kebutuhan dasar *user* yang tertera dalam poin-poin Analasisa Kebutuhan Sistem.
c. Program ini mudah untuk digunakan kapan saja dan tidak membutuhkan spesifikasi komputer yang tinggi.
d. Harapan kedepannya, bahwa Sistem Penunjang Keputusan ini bisa terus dikembangkan sehingga bisa memberikan solusi dari kondisi yang lebih detail dan lebih rumit lagi.
e. Semoga Sistem Penunjang Keputusan ini, suatu saat nanti bisa dikembangkan kedalam tampilan *Graphical User Interface (GUI)* dan bisa dijalankan pada *multi-platform* seperti Android, Linux, *Web*, dan lain sebagainya.

UCAPAN TERIMAKASIH

Terimakasih yang sebesar-besarnya kepada:

1. Kedua Orang Tua, saudara kandung penulis, istri dan putra-putri Penulis yang selalu memberikan *support* yang tidak pernah terputus hingga saat ini.
2. Rekan-rekan dosen dan staff Universitas Bina Sarana Informatika yang selalu membantu Penulis dalam memenuhi kegiatan Tri Dharma Perguruan Tinggi.
3. Bapak/Ibu *Reviewer* dan *Staff Editor* Jurnal Ilmiah *FLASH* yang berkenan memberikan kritik membangun dan masukan positif yang membangun hingga memberikan kesempatan untuk diterbitkannya hasil penelitian ini.
4. Bapak/Ibu peneliti sebelumnya yang karya tulisnya menjadi referensi bagi Penulis dalam menyelesaikan penelitian ini.
5. Para rekan-rekan pengusaha UMKM yang telah menyediakan waktunya untuk menjadi narasumber pada penelitian kali ini.
6. Semua pihak yang tidak bisa disebutkan satu-persatu yang terlibat dalam penelitian kali ini.

DAFTAR PUSTAKA

[1] A. S. Rosa, Shalahuddin, “Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek”, Bandung: Informatika, 2015.

[2] T. P. Dinar, H. Kuswara, F. E. Schaduw, A. S. F. Utami, “Sistem Pendukung Keputusan Dalam Pemilihan Sim Card Provider Menggunakan Metode Simple Additive Weighting (Saw) (Studi Kasus: Master Cell)”, Journal Speed - Sentra Penelitian Engineering dan Edukasi, vol. 12, no. 2, 2020.

[3] M. Romzi, B. Kurniawan, “Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma”, Jurnal Teknik Informatika Mahakarya, vol. 3, no. 2, 2020.

[4] D. Supriadi, “Perbandingan Penyelesaian Knapsack Problem Secara Matematika, Kriteria Greedy Dan Algoritma Greedy”, Indonesian Journal on Computer and Information Technology, vol. 1, no. 2, 2016.

[5] Yulikuspartono, “Pengantar Logika dan Algoritma”. Yogyakarta: Andi, 2001