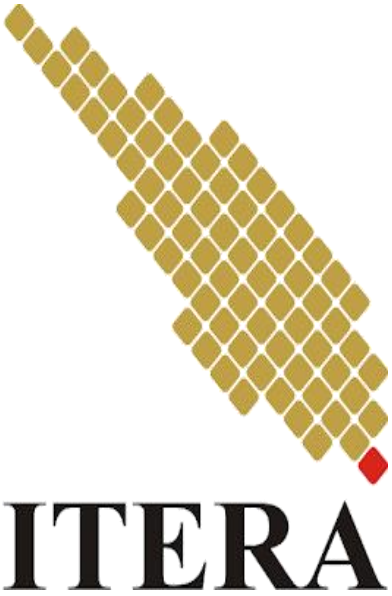
Penerapan Algoritma *Dynamic Programming* Pada *Knapsack Problem* Untuk Optimasi Maksimasi Keuntungan Dalam Sebuah Pengiriman Barang

Tugas Besar Algoritma Strategi



Kelompok 6 (RB)

Ketua Kelompok	120450006	Arfyani Deiastuti
Anggota 1	120450026	Akbar Fadhillah IU
Anggota 2	120450108	Ribka Gabriela S.
Anggota 3	120450008	Sophia Yolanda RI
Anggota 4	120450084	Angga Pramana p

INTRODUCTION

Pada setiap perusahaan ataupun badan usaha, baik dalam bentuk jasa maupun barang diperlukan sistem kerja yang baik agar dapat memperoleh keuntungan secara maksimal. Akan tetapi, masih ada pihak kurang peduli pada beberapa aspek dalam sistem kerja, seperti pada bagian penyimpanan dalam pengiriman barang. Banyak barang yang dikemas secara sembarang seperti pengemasan dalam boks tanpa memperdulikan berat barang tersebut dapat menyebabkan kerugian serta kerusakan barang. Oleh karena itu dibutuhkan sebuah optimasi untuk mengatur sistem kerja tersebut. Optimasi dibagi menjadi 2 bagian, yaitu maksimisasi merupakan optimasi dengan menggunakan input tertentu untuk mendapatkan keuntungan maksimal dan minimisasi merupakan optimasi untuk menghasilkan output tertentu dengan menggunakan input yang minimal. Adanya sistem optimasi ini kita dapat mencapai target dengan cara yang lebih efisien.

Optimasi yang digunakan dalam permasalahan ini merupakan permasalahan knapsack. Masalah Knapsack (Knapsack Problem) adalah suatu permasalahan tentang bagaimana cara memilih objek dari beberapa objek yang akan diinput ke media penyimpanan dengan setiap objek memiliki bobot serta total bobot dari objek yang dipilih tidak boleh melebihi kapasitas penyimpanan, agar diperoleh keuntungan atau hasil yang maksimal. Permasalahan knapsack dapat diselesaikan dengan berbagai algoritma, salah satunya adalah *Dynamic Programming. Dynamic Programming* merupakan salah satu teknik perancangan algoritma yang dikembangkan untuk menyelesaikan permasalahan untuk mencari hasil terbaik, baik maksimal maupun minimal, dari sebuah solusi. Kelebihan *Dynamic Programming* adalah dapat diaplikasikan untuk berbagai macam masalah pemrograman matematik, karena *Dynamic Programming* cenderung lebih fleksibel daripada teknik optimasi lain dan dapat menyesuaikan sistematik perhitungannya menurut ukuran masalah yang tidak selalu tetap dengan melakukan perhitungan satu persatu secara lengkap dan menyeluruh. Oleh karena itu, kami tertarik untuk melakukan penelitian tentang penerapan *Dynamic Programming* Pada Knapsack Problem untuk Optimasi Maksimasi Keuntungan Dalam Sebuah Penyimpanan Barang.

PROBLEM STATEMENT

Masalah penyimpanan untuk pengiriman barang ditemukan pada *brand Brooks Running Shoes* asal Amerika merupakan sebuah brand olahraga yang memfokuskan diri untuk memproduksi sepatu khusus lari sejak 1914, dengan berbagai macam item yang dijual brand ini tentunya memiliki jenis dan ukuran yang beragam. Banyaknya jenis barang yang diproduksi menimbulkan masalah dalam pengemasan untuk pengiriman barang. Barang yang akan dikirim pastinya akan dikemas dalam *box-box* yang memiliki keterbatasan ruang, dengan kapasitas boks yang kurang maksimal, dapat mengakibatkan brand mengalami kerugian dan apabila kapasitas nya melewati batas maksimal, maka akan mengakibatkan barang rusak karena boks tidak dapat menampung dengan baik.

Oleh karena itu, petugas harus memasukkan barang sebaik dan semaksimal mungkin kedalam tempat barang tersebut ketika menerima permintaan pengiriman ke konsumen. Barang dimasukkan kedalam wadah berdasarkan berat, harga, serta tingkat kepentingan barang tersebut. Petugas akan memilih barang yang sesuai dengan tempat atau wadah dengan pertimbangan berat barang tidak melebihi kapasitas maksimum sehingga dapat mengoptimalkan tempat yang digunakan. Melalui proses analisis ini diharapkan pengiriman jumlah barang dapat dimaksimalkan sehingga mendapatkan keuntungan yang sebesar-besarnya.

Perumusan masalah pada penelitian ini yaitu barang yang memiliki profit yang tinggi dengan berat yang lebih ringan itu akan lebih diutamakan untuk dikemas dan dikirim, dan harus memenuhi kapasitas yang ada. Tujuan penelitian ini adalah untuk membantu *brand Brooks Running Shoes* dalam mendapatkan data-data barang yang dapat dikemas dengan profit dan weight yang paling maksimal untuk mendapatkan keuntungan yang tinggi dengan Metode *Dynamic Programming* pada Knapsack Problem. Batasan masalah yang digunakan adalah analisis ini menggunakan dua variabel dari data yang ada yaitu ***price*** sebagai *profit* dan ***weight*** sebagai *weight*, tanpa adanya penambahan atau pengurangan dari data tersebut. Analisis ini juga menggunakan asumsi bahwa wadah atau tempat barang yang digunakan untuk mengemas barang memiliki kapasitas yang sama dan benar.

DATA DESCRIPTION

Pada tugas RBL ini, penulis mengambil data dari http://brooksrunning.com/ tahun 2020. Data dikumpulkan secara manual menangkap bagian spesifikasi dari setiap halaman sepatu yang berlaku dalam situs *web Books Running*. Data yang digunakan yaitu data barang penjualan sepatu pada *brand Brooks Running Shoes* yang mencakup 26 model dengan rincian sebagai berikut :

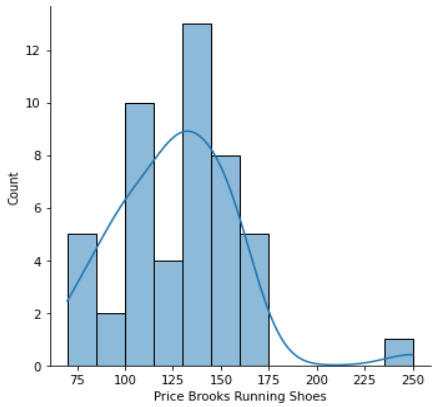
- **Name** : Nama sepatu
- **Type** : Tipe (man's/Women's/unisex)
- **Price** : Harga Sepatu (Tahun 2020)
- **Midsole Drop** : Penurunan midsole dalam mm
- **Weight** : Berat dalam gr

Tabel 1. Data Sepatu Lari Brooks Tahun 2020

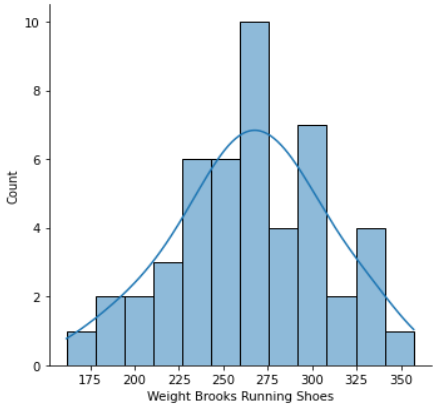
Name	Type	Price	Midsole Drop(mm)	Weight
Addiction 14	Men's	130	12	357.2
Adrenaline GTS 20	Men's	130	12	300.5
Anthem 3	Men's	69.95	10	212.6
Asteria	Men's	110	8	235.3
Beast 20	Men's	160	12	331.7
Bedlam 3	Men's	150	8	306.2
Caldera 4	Men's	140	4	283.5
Cascadia 15	Men's	130	8	311.8

Cascadia 15 GTX	Men's	160	8	331.7
Divide	Men's	100	8	292
Dyad 11	Men's	130	10	328.9
Ghost 13	Men's	130	12	286.3
Glycerin 18	Men's	150	10	289.2
Hyperion	Men's	130	10	181.4
Hyperion Tempo	Men's	150	8	207
Hyperion Elite	Unisex	250	8	195.6
Launch 7	Men's	100	10	255.1
Levitate 4	Men's	150	8	292
PureFlow 7	Men's	90	4	252.3
PureGrit 8	Men's	78	4	263.7
Ravenna 11	Men's	110	10	266.5
Revel 4	Men's	100	8	252.3
Ricochet 2	Men's	120	8	275
Ricochet 2 LE	Men's	120	8	275
Transcend 7	Men's	160	10	303.3
Ghost 13	Women's	130	12	249.5
Adrenaline GTS 20	Women's	130	12	266.5
Revel 4	Women's	100	8	224
Glycerin 18	Women's	150	10	255.1
Levitate 4	Women's	150	8	260.8
Bedlam 3	Women's	150	8	269.3
Addiction 14	Women's	130	12	323.2
PureFlow 7	Women's	90	4	212.6
Ravenna 11	Women's	110	10	238.1
Transcend 7	Women's	160	10	269.3
Ricochet 2	Women's	120	8	238.1
Cascadia 15	Women's	130	8	283.5
Divide	Women's	100	8	260.8
Hyperion Tempo	Women's	150	8	189.9
Caldera 4	Women's	140	4	252.3
Ricochet 2 LE	Women's	120	8	275
Cascadia 15 GTX	Women's	160	8	294.8
Anthem 3	Women's	69.95	10	235.3
Hyperion	Women's	130	10	161.6
Launch 7	Women's	100	10	226.8
Dyad 10	Women's	84.5	10	297.7
Ariel '18	Women's	104	12	334.5
PureGrit 8	Women's	78	4	232.5

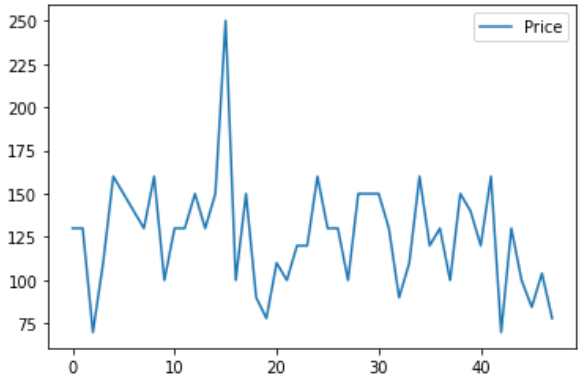
Visualisasi Data



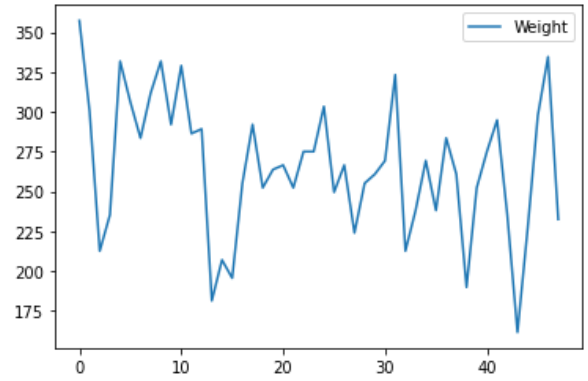
Gambar 1. Grafik Price Running Shoes



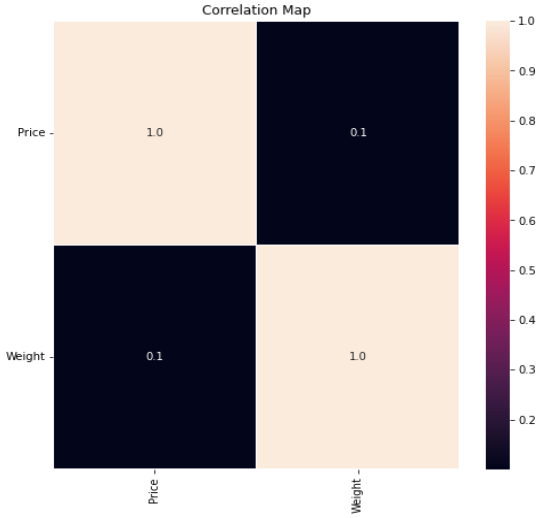
Gambar 2. Grafik Weight Running Shoes



Gambar 3. Plot Data Price



Gambar 4. Plot Data Weight



Gambar 5. Correlation Map dari Price dan Weight

Data Processing

Dari lima variabel yang ada pada data yang digunakan hanya dua variabel yang akan digunakan sebagai parameter yaitu price dan weight dengan jumlah data adalah 48. Dengan parameter profit

dan weight yang berbeda, dimana barang tersebut akan ditentukan keuntungan mana yang paling optimal dan akan dilakukan penyimpanan dalam *knapsack* untuk dilakukan pengiriman. Pada kasus ini, diketahui 48 item yang akan dicari keuntungan yang optimal berdasarkan berat dan harga. Berat masing-masing barang dimulai dari 161,6-357,2 gr dengan berat rata-rata adalah 265,36 gr. Harganya juga yang bervariasi yang mulai dengan harga paling murah yaitu $69,95 dan paling mahal $250, dengan harga rata-rata adalah $125,72.

Tabel 2. Nilai Statistik dari Data

	Price	Weight
count	48.000000	48.000000
mean	125.716667	265.360417
std	31.617932	43.060066
min	69.950000	161.600000
25%	100.000000	237.400000
50%	130.000000	266.500000
75%	150.000000	292.700000
max	250.000000	357.200000

METHODS

Dynamic Programming

Pemrograman dinamis (*Dynamic Programming*) adalah metode pengoptimalan matematika yang memiliki prosedur sistematis dengan tujuan penyelesaian masalah dengan cara menguraikan sub-sub masalah yang lebih kecil yang terkait satu sama lain dengan tetap memperhatikan kondisi dan batasan permasalahan tersebut. Pemrograman dinamis biasanya digunakan untuk masalah optimisasi, dimana suatu permasalahan memiliki banyak solusi dan setiap solusi memiliki nilai masing-masing yang ditemukan dengan nilai yang optimum (maksimal atau minimal). Solusi optimal dari masalah tersebut dapat dipandang sebagai suatu deret keputusan.

Pemrograman dinamis dapat dibagi menjadi empat tahap berurutan sebagai berikut:

1. Karakterisasi struktur pada solusi optimasi
2. Mendefinisikan nilai solusi optimal secara rekursif
3. Menghitung nilai solusi optimal pada model bottom-up
4. Menyusun solusi optimal dari informasi hasil perhitungan

Langkah 1-3 merupakan dasar dari pemrograman dinamis dalam menemukan solusi dari suatu permasalahan, dan langkah ke-4 dapat dilakukan jika nilai solusi optimalnya diperlukan.

Pemrograman dinamis memiliki prinsip-prinsip dasar, yaitu:

- Prinsip pertama dalam model pemrograman dinamis adalah bahwa masalah dapat dibagi menjadi bagian-bagian masalah yang lebih kecil. Masalah yang lebih kecil atau sub masalah ini tersebut sebagai tahap keputusan (stage). Setiap masalah uang akan diselesaikan, terlebih dahulu dibagi-bagi menjadi beberapa masalah kecil dengan maksud memudahkan evaluasi masalah untuk mendapatkan keputusan optimal dari tiap-tiap tahap yang pada akhirnya akan menghasilkan satu keputusan yang optimal. Oleh karena itu model Dynamic programming disebut juga model multistage programming (model multi tahap).
- Prinsip kedua dalam model pemrograman dinamis adalah tentang status (state). Pengertian status (*state*) dalam pemrograman dinamis adalah arus informasi dari suatu tahap ke tahap berikutnya. Arus informasi yang masuk ke suatu tahap disebut status input, sedangkan arus informasi yang keluar dari suatu tahap disebut stats output. Status input penting, karena keputusan pada tahap berikutnya tergantung dari status input sebelumnya. Jadi, status input untuk tahap keputusan n-1 merupakan status output dari tahap keputusan sebelumnya, yaitu tahap keputusan n. Sedangkan status output dari tahap keputusan n akan menjadi status input untuk tahap keputusan berikutnya, yaitu keputusan n-1.
- Prinsip ketiga dalam pemrograman dinamis adalah tentang variabel keputusan yang dinyatakan dalam berbagai bentuk keputusan alternatif yang dapat dipilih pada saat pengambilan keputusan pada tahap tertentu. Berbagai alternatif keputusan yang dapat diambil dalam setiap tahap keputusan dapat dibatasi dalam sejumlah persyaratan yang dikenalkan dalam struktur masalah.
- Prinsip keempat dalam model pemrograman dinamis adalh tentang fungsi transformasi yang memberikan penjelasan tentang bagaimana hubungan antara tahap keputusan yang satu dengan tahap keputusan yang lain dalam formulasi pemrograman dinamis. Selain itu fungsi transformasi juga menyatakan tentang hubungan fungsional nilai status pada setiap tahap keputusan yang bersifat berulang.

Programming rekursif maju yaitu :

$fi\ (y)$ = maks $\{fi{-}1\ (y), fi{-}1\ (y - wi\ ) + pi\ \}$ (1)

Jika $fi{-}1\ (y - wi\ ) + pi > fi{-}1\ (y)$ (2)

maka objek pada tahap-i dimasukkan ke dalam knapsack, dan sebaliknya untuk

$fi{-}1\ (y - wi\ ) + pi < fi{-}1\ (y)$ (3)

maka objek pada tahap-i tidak dimasukkan ke dalam knapsack.

Perhitungan Rekursif Mundur :

$fi\ (y)$ = maks $\{fi{+}1\ (y), fi{+}1\ (y - wi\ ) + pi\ \}$ (4)

Jika $fi{+}1\ (y - wi\ ) + pi > fi{+}1\ (y)$ (5)

maka objek pada tahap-i dimasukkan ke dalam knapsack, dan sebaliknya untuk :

$fi{+}1\ (y - wi\ ) + pi < fi{+}1\ (y)$ (6)

maka objek pada tahap-i tidak dimasukkan ke dalam knapsack.

Keterangan :

i = 1, 2, 3, … , n, dan $i = n, n-1, n-2, n-3, \cdots, 1$

fi = nilai optimal dari permasalahan yang diselesaikan pada tahap i,

y = kapasitas knapsack pada tahap i,

$fi\ (y)$ = nilai optimal dari permasalahan knapsack 0-1 yang diselesaikan pada tahap i dengan kapasitas knapsack sebesar y.

Knapsack Problem

Knapsack Problem dapat digambarkan sebagai tas, ransel, karung atau kantong. *Knapsack problem* merupakan penyelesaian masalah optimasi kombinasi yang bertujuan untuk mendapatkan nilai maksimal dari barang-barang yang dimasukkan ke dalam *knapsack* atau wadah atau tempat dengan berat seminimal mungkin dan tidak melewati kapasitasnya. Dengan *knapsack problem* kita menentukan bagaimana memilih dari sekian banyak barang yang ada untuk dimasukkan kedalam *knapsack* sehingga mendapat penyimpanan yang optimal dengan mempertimbangkan barang yang terdiri dari n barang yang memiliki berat (W_n) dan nilai profit (p_n) dengan memperhatikan juga kapasitas dari tempat penyimpanan (W) dan nilai probabilitas dari setiap barang (X_n). Masalah knapsack merupakan permasalahan optimisasi kombinatorial yang diklasifikasikan sebagai NP-complete problem. Tujuannya adalah untuk memperoleh keuntungan maksimum tanpa melebihi kapasitas knapsack dengan objek yang dipilih. Inti dari knapsack 0-1 adalah objek yang tersedia harus diambil secara keseluruhan atau tidak sama sekali. Persoalan Knapsack 0-1 pada wadah terbuka ditulis sebagai berikut :

Maksimum :

$\boldsymbol{Z} = \sum_{i=1}^{n} n\, pixi$ **(1)**

dengan kendala :

$\sum_{i=1}^{n} n\, wixi \leq \boldsymbol{M}$ **(2)**

Keterangan :

i = 1, 2, 3, … , n,

z = keuntungan total,

n = banyaknya jenis objek,

pi = keuntungan per satuan objek ke-i,

wi = berat objek ke-i, dan

M = kapasitas maksimal knapsack.

xi = { 0, jika objek − i tidak dimasukkan

1, jika objek − i dimasukkan

Algoritma / Pseudocode

```
for w ← 0 to W do
        c[0,w] ←0
end for
for i ← 1 to n do
        c[i,0] ← 0
end for
for i ← 1 to n do
   for w=0 to W do
    if w[i] ≤ w then
     if v[i] + c[ i,w-w[i] ] > c[i – 1, w ] then
       c[i,w] ←v[i] + c[i-1, w-w[i] ]
         else
       c[i,w] ← c [i–1,w]
     end if
    else
       c[i,w] ← c[i-1,w]
   end if
  end for
end for
```

Gambar 6. Algoritma *Dynamic Programming* pada 0-1 Knapsack

RESULTS AND DISCUSSIONS

Diketahui terdapat 48 jenis barang (n = 48) yang dinotasikan dengan x = (x1 x2, …,x48). Berat barang i dinotasikan dengan wi yaitu w = (w1, w2, … , w48) = (130, 130, 70, …, 78). Kapasitas angkut dinotasikan dengan M = 8.000. Tujuan dari penggunaan Dynamic Programming yaitu mencari total keuntungan yang maksimal dari pengangkutan barang. Data yang awalnya bertipe float diubah menjadi data bertipe integer supaya masalah bisa diselesaikan dengan metode *Dynamic Programming.*

Tabel 3. Data Jenis Barang, Berat, dan Harga

Item	Price	Weight(g)
1	130	357
2	130	301
3	70	213
4	110	235
5	160	332
...	...	...
44	130	162
45	100	227
46	85	298
47	104	335
48	78	233

```
vals = df['Price'].astype(int)
wts = df['Weight'].astype(int)
capacity = 8000
```

Gambar 7. Pengubahan Tipe Data

Perhitungan rekursif maju tahap ke-1 : Keuntungan maksimal yang diperoleh untuk i = 0 adalah $f0\ (y) = 0$. Berdasarkan Tabel 3, berat barang 1 (w1) sebesar 130gr dan keuntungannya (p1) sebesar 357, dengan menggunakan Persamaan (1) maka $f1\ (y)$ = maks $\{f0\ (y), f0\ (y - w1\ ) + p1\ \}$ $f1\ (y)$ = maks$\{f0\ (y), f0\ (y - 130) + 357\ \}$, dilanjutkan dengan tahap-tahap selanjutnya. Perhitungan rekursif mundur tahap ke-1 : Untuk i = 48, keuntungan maksimal yang diperoleh adalah $f49(y) = 0$ Berdasarkan Tabel 3 , diketahui jenis barang ke-48 memiliki berat barang sebesar 78 gr dan keuntungan sebesar 233, dengan menggunakan Persamaan (4) Untuk i = 48, maka $f48(y)$ = maks $\{f48(y), f48(y - 78) + 233\}$, dilanjutkan dengan tahap-tahap selanjutnya.

```
table = [[0 for x in range(w)] for y in range(h)]
print(table)
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Gambar 8. Hasil Barang yang Dipilih

```
print(max(solution_arr))
```

```
4310
```

```
max([x for y in table for x in y])
```

```
4310
```

Gambar 9. Solusi terbaik dari Algoritma *Dynamic Programming*

Solusi yang diperoleh dapat dilihat dari Tabel 4 dimana barang yang dipilih ditandai dengan angka (1) sedangkan barang yang tidak dipilih ditandai dengan angka (0). Sehingga diperoleh nilai profit terbaik dari Algoritma *Dynamic Programming* adalah 4310 dengan weight 8000 dan total barang yang diambil adalah 31. Solusi knapsacknya adalah (0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0). Dengan rincian barang yang diambil ialah barang ke- (4-7, 9, 12-16, 18, 22-31, 34-37, 39-42, 44, dan 45).

Tabel 4. Data Jenis Barang, Berat, dan Harga dari Solusi Terbaik

Item	Price	Weight(g)	Optimal
1	130	357	0
2	130	301	0
3	70	213	0
4	110	235	1
5	160	332	1
6	150	306	1
7	140	284	1
8	130	312	0
9	160	332	1
10	100	292	0
11	130	329	0
12	130	286	1
13	150	289	1
14	130	181	1
15	150	207	1
16	250	196	1
17	100	255	0
18	150	292	1
19	90	252	0
20	78	264	0
21	110	267	0
22	100	252	1
23	120	275	1
24	120	275	1
25	160	303	1
26	130	250	1
27	130	267	1
28	100	224	1
29	150	255	1
30	150	261	1
31	150	269	1
32	130	323	0
33	90	213	0
34	110	238	1
35	160	269	1
36	120	238	1
37	130	284	1
38	100	261	0

39	150	190	1
40	140	252	1
41	120	275	1
42	160	295	1
43	70	235	0
44	130	162	1
45	100	227	1
46	85	298	0
47	104	335	0
48	78	233	0
Total Bobot			8000
Total Keuntungan			4310
Total Barang			31

CONCLUSION

Hasil penyelesaian knapsack problem menggunakan Algoritma *Dynamic Programming* Pada Knapsack Problem dengan data penjualan pada *brand Brooks Running Shoes* menunjukkan bahwa Algoritma tersebut dapat menyelesaikan permasalahan knapsack dan menghasilkan solusi optimal. Solusi optimal yang didapatkan dengan nilai profit $4310 dan weight 8000 gr serta barang yang dipilih ada 31 barang.

Penelitian ini tidaklah sepenuhnya baik dan efektif maka dari itu penelitian selanjutnya perlu diperbaiki dan dikembangkan algoritma tersebut supaya lebih baik dan efektif untuk menyelesaikan integer knapsack problem. Masalah knapsack 0-1 merupakan permasalahan optimisasi yang dapat dikerjakan dengan berbagai macam metode dan algoritma. Masih terbuka bagi peneliti berikutnya untuk menerapkan algoritma metaheuristic yang lain dan membandingkan hasil dari setiap algoritma tersebut, sehingga didapatkan metode penyelesaian terbaik untuk menyelesaikan permasalahan knapsack 0-1. Peneliti selanjutnya bisa menggunakan permasalahan yang berbeda atau sudah dikembangkan guna menyelesaikan permasalahan pada realita yang ada.

REFERENCES

1. Referensi ilmiah yang digunakan
   Penerapan Algoritma Dynamic Programming Pada Permasalahan Knapsack 0-1. Journal homepage: (Irmeilyana, Putra Bahtera Jaya Bangun, Dian Pratamawati, Winda Herfia Septiani)

   Resume :

   Judul : Penerapan Algoritma Dynamic Programming Pada Permasalahan Knapsack 0-1

Penelitian ini menggunakan data sekunder pada penelitian yang diperoleh dari UD. Subur Tani Makmur pada bongkar muat pupuk dan kebutuhan pertanian. Diketahui terdapat 19 jenis barang (n =19) yang dinotasikan dengan x = (x1 x2,...,x19). Berat barang i dinotasikan dengan wi yaitu w = (w1, w2, ... , w19) = (1.000, 750, 750, ..., 6). Kapasitas angkut dinotasikan dengan M = 6.000. Tujuan dari penggunaan Dynamic Programming yaitu mencari total keuntungan yang maksimal dari pengangkutan barang.

Hasil :

Dalam Keuntungan di UD. Subur Tani Makmur berdasarkan algoritma Dynamic Programming perhitungan rekursif maju yaitu sebesar Rp 118.096.500 dengan total berat barang yang diangkut adalah 5.981 kg sehingga memenuhi 99,683 % dari kapasitas truk. Keuntungan berdasarkan algoritma Dynamic Programming perhitungan rekursif mundur yaitu sebesar Rp 86.246.500 dengan total berat barang yang diangkut adalah 5.881 kg sehingga memenuhi 98,017 % dari kapasitas truk.

Penggunaan algoritma Dynamic Programming perhitungan rekursif maju menghasilkan solusi keuntungan yang sama dengan metode Branch and Bound dan merupakan solusi paling maksimal dibanding algoritma Greedy. Algoritma Dynamic Programming perhitungan rekursif mundur dan algoritma Greedy by Weight menghasilkan solusi keuntungan yang sama dan merupakan solusi yang paling minimal.

2. Sumber data yang digunakan

   https://www.kaggle.com/hannahcollins/2020-brooks-running-shoes

   - desc : Kumpulan data ini mencakup 26 model sepatu Brooks Running terkini yang diperoleh dari http://brooksrunning.com/. Data terdiri dari : Nama sepatu, Tipe, Harga (per 29 Agustus 2020), Dukungan, Pengalaman, Permukaan, Penurunan midsole dalam mm, Berat dalam g, Jenis lengkungan, dan Fitur sepatu tambahan.

APPENDIX

1. Link Source Code , dapat berupa link Github atau Gdrive
   Link : https://colab.research.google.com/drive/1Vks_UrJwYTSKQr575r4ihSFnYMywY3ns

Tabel Pembagian Tugas

No	Nama	NIM	Job desk	Deskripsi Tugas
1	Arfyani Deiastuti	120450006	Ketua Kelompok	Mencari data dan mengaplikasikan Algoritma di google colab menggunakan data yang ada, Data Description, Methods, Results and Discussion, dan PPT.
2	Akbar Fadhillah IU	120450026	Anggota 1	Introduction, Problem Statement.

3	Ribka Gabriela S.	120450108	Anggota 2	Mencari data dan mengaplikasikan Algoritma di google colab menggunakan data yang ada, Methods, dan PPT.
4	Sophia Yolanda RI	120450008	Anggota 3	Introduction, Problem Statement.
5	Angga Pramana p	120450084	Anggota 4	Conclusion dan Mencari referensi.