

ANALISIS ALGORITMA GREEDY DAN DYNAMIC PROGRAMMING PADA PERMASALAHAN KNAPSACK

DESAIN ANALISA ALGORITMA


1. Fadhilah Nuria Shinta
2. Dhani Aditya Putra E.
3. Reshar Faldi Julianda

DOSEN PENGAMPU :
Dr. Atik Wintarti, M.Kom.

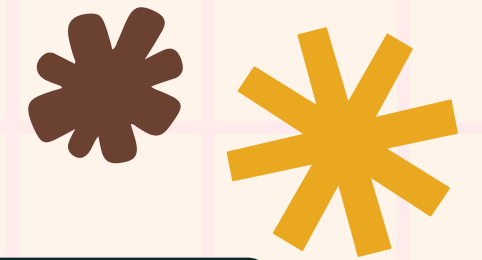




Kontribusi Anggota Kelompok

- 
- Fadhilah Nuria Shinta
Code Greedy
Laporan Akhir
Visualisasi greedy
 - Dhani Aditya Putra E.
Code Greedy
PPT
Visualisasi greedy
 - Reshar Faldi Julianda
Code Dynamic Programming
Visualisasi Dynamic Programming
PPT

Pendahuluan



Penyimpanan barang yang efisien dan efektif memainkan peran penting dalam kehidupan sehari-hari. Dengan menyusun barang dengan baik dan terorganisir, kita dapat menghemat waktu, dan meningkatkan produktivitas. Salah satu permasalahan yang sering dialami adalah permasalahan ranjang (Knapsack Problem) dimana adanya kapasitas suatu ranjang atau tempat penyimpanan yang akan diisi dengan barang barang apa saja yang memiliki keuntungan tertinggi dengan bobot tidak melebihi kapasitas penyimpanan yang dimiliki.

Dalam upaya menangani permasalahan Knapsack, kami menerapkan dua pendekatan algoritma, yakni algoritma greedy dan algoritma dynamic programming. Untuk menentukan performa yang optimal, kami melaksanakan analisis perbandingan antara algoritma greedy dan dynamic programming dalam konteks penyelesaian permasalahan knapsack. Harapan akhir dari penelitian ini adalah memperoleh pemahaman mendalam mengenai perbandingan hasil kinerja kedua algoritma, terfokus pada kompleksitas algoritma serta durasi komputasi masing-masing algoritma.



Tinjauan Pustaka



1. Knapsack Problem

Masalah knapsack (atau knapsack problem) adalah dimana Dalam masalah ini, kita memiliki sebuah ranjang dengan kapasitas tertentu, dan sejumlah item yang memiliki berat dan nilai masing-masing. Tujuan dari masalah knapsack adalah menentukan kombinasi item yang dapat dimasukkan ke dalam ranjang sedemikian rupa sehingga jumlah nilai item tersebut maksimal, tetapi total beratnya tidak melebihi kapasitas ranjang.

Masalah knapsack memiliki relevansi yang signifikan dalam kehidupan sehari-hari, terutama dalam konteks jasa pengangkutan barang seperti pengiriman seperti peti kemas, pengangkutan barang dalam ransel, dll. Dalam aktivitas ini, tujuannya adalah mencapai keuntungan maksimal dengan tetap mematuhi kapasitas yang tersedia.

Masalah knapsack sendiri dapat dibagi menjadi beberapa jenis, antara lain:

1. Knapsack 0-1 (integer knapsack), di mana objek harus dimasukkan sepenuhnya atau tidak dimasukkan sama sekali.

2. Bounded Knapsack, di mana objek dapat dimasukkan sebagian atau seluruhnya ke dalam media penyimpanan.

3. Unbounded knapsack, di mana jumlah dan jenis objek yang dimasukkan ke dalam media penyimpanan tidak terbatas



Tinjauan Pustaka

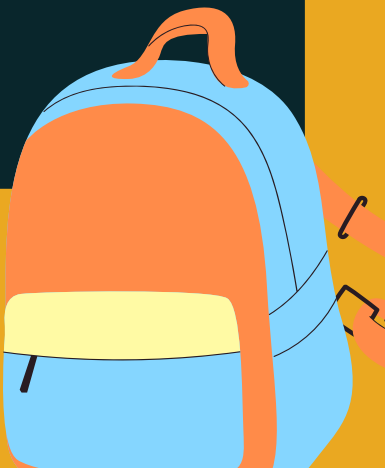


2. Algoritma Greedy

Metode greedy dapat dijelaskan sebagai strategi yang bersifat rakus atau tamak. Algoritma greedy adalah pendekatan permasalahan langkah per langkah, di mana pada setiap langkahnya terdapat beberapa opsi yang harus dipertimbangkan. Oleh karena itu, dalam setiap langkah, keputusan terbaik perlu diambil untuk menentukan opsi yang optimal, dengan harapan bahwa langkah-langkah selanjutnya akan membawa kita menuju solusi global yang optimal (global optimum).

Dalam penyelesaian Knapsack Problem, terdapat tiga jenis algoritma Greedy yang dapat digunakan, yaitu:

1. Greedy by profit: Tempat penyimpanan diisi dengan objek yang memiliki keuntungan terbesar.
2. Greedy by Weights: Tempat penyimpanan diisi dengan objek yang memiliki berat paling ringan.
3. Greedy by density: Tempat penyimpanan diisi dengan objek yang memiliki densitas terbesar pada setiap tahap. Densitas dihitung dengan membagi nilai keuntungan suatu objek dengan beratnya. Pilihan objek dilakukan berdasarkan densitas tertinggi pada setiap langkah.



Tinjauan Pustaka

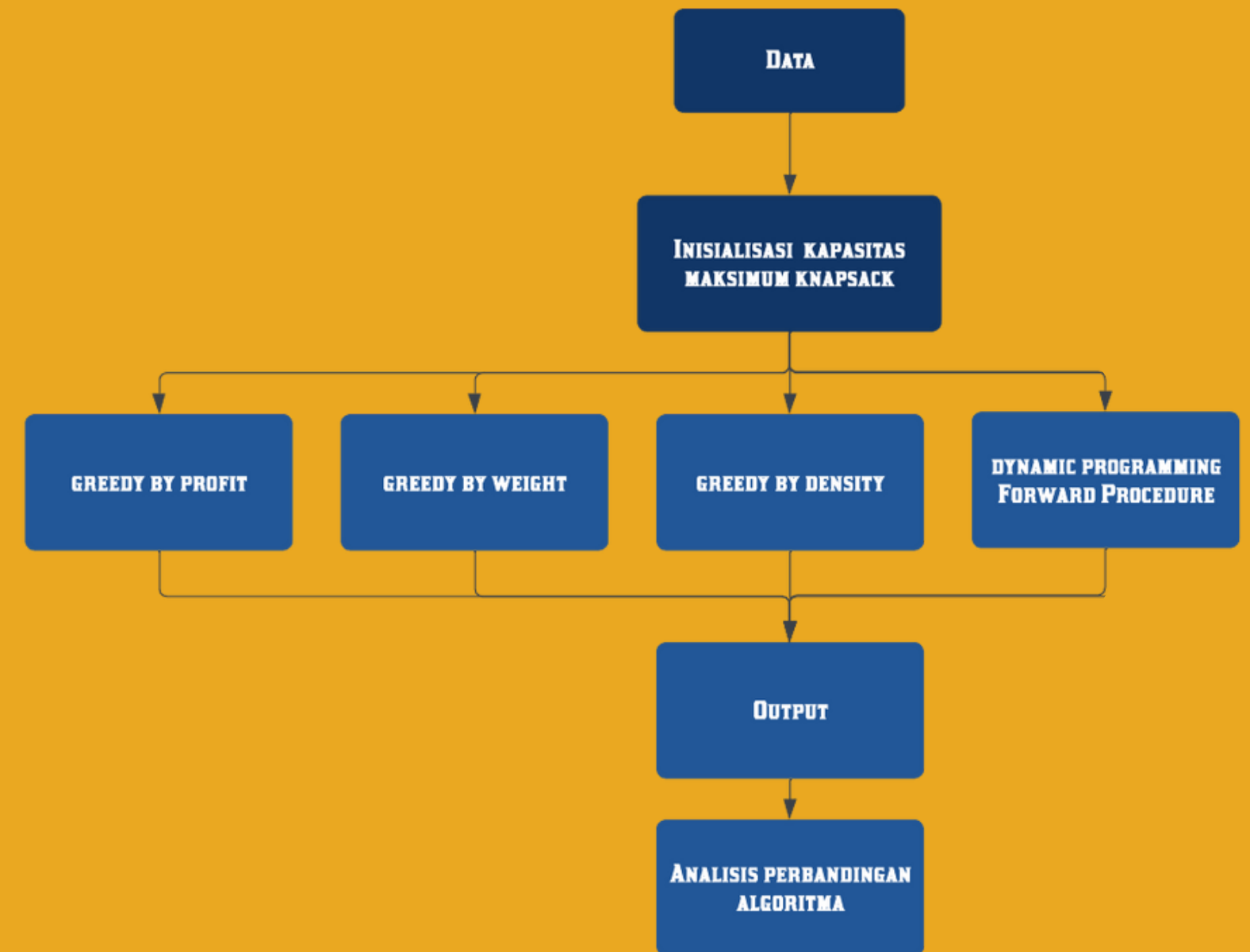
3. Algoritma Dynamic Programming

Dynamic programming adalah suatu pendekatan matematis yang terutama dirancang untuk meningkatkan efisiensi perhitungan dalam penyelesaian masalah pemrograman matematis tertentu. Pendekatan ini mencapai efisiensi dengan memecah masalah besar menjadi bagian-bagian yang lebih kecil, menjadikannya lebih mudah untuk dihitung. Suatu masalah yang diselesaikan menggunakan dynamic programming dibagi menjadi tahap-tahap, dengan setiap tahap merepresentasikan kemungkinan kapasitas dari masalah tersebut.

Langkah-langkah dalam menyelesaikan suatu masalah menggunakan dynamic programming melibatkan prosedur rekursif. Artinya, dalam setiap pengambilan keputusan, perlu mempertimbangkan kondisi yang dihasilkan oleh keputusan optimal sebelumnya, yang kemudian menjadi dasar untuk keputusan optimal berikutnya. Terdapat dua jenis prosedur rekursif dalam penyelesaian masalah, yaitu prosedur maju (forward procedure) dan prosedur mundur (backward procedure).



Diagram Alir Penelitian



DATA MENTAH

No	Nama Barang	Banyak Barang	Harga Beli	Harga Jual
1	PRIMAFUR 2kg	10	16500	24000
2	CALCIUM KUPU2 1kg	10	12500	37500
3	CALCIUM KUMBANG 1kg	10	8500	43500
4	CRASH 1 ltr	10	39500	54500
5	MKP KRISTA 1kg	20	22500	73500
6	NITRAFOS 1kg	10	22500	97500
7	NITRABOR 1kg	3	14000	17000
8	CALCINIT 1kg	10	12500	37500
9	KALI CHILI YARA 2kg	9	19500	42000
10	PLASTIK MULSA 9kg	5	24000	34000
11	PLASTIK MULSA 18kg	5	24000	34000
12	PLASTIK 2KELINCI 5kg	5	95000	105000
13	PLASTIK 2KELINCI 10kg	4	178000	194000
14	AMEGRASS 5kg	4	317500	407500
15	AMEGRASS 20kg	2	63500	67000
16	SIDAMIN 1kg	10	46000	86000
17	SIDAFOS 1kg	10	31500	66500
18	P21 1kg	15	56000	71000
19	BISI 2 1kg	3	42000	43500
20	CIHERANG SS 10kg	3	73000	79000
21	INTANI 10kg	2	73000	87000
22	NPK MUTIARA 5kg	4	339000	383000
23	KCL SHS 5kg	4	255000	275000
24	ZA 5kg	4	70000	90000
25	UREA D/B 5kg	4	89000	93000
26	PHONSKA 5kg	4	115000	119000
27	NPK TAWON 5kg	4	325000	345000
28	KNO.3 MERAH 2kg	5	18900	24400
29	KNO.3 PUTIH 2kg	5	27800	28800
30	REGENT.03G 1kg	10	16250	23750
31	ALPHADIN 1kg	10	12500	17500
32	FURADAN 3G 2kg	5	19500	22000
33	ZK 1kg	5	16000	26000
34	MKP KNO.3 SAPRTAN 3kg	5	22800	33800
35	FERRTERA 2kg	5	77500	90000
36	PRIMA STICK 1kg	5	9000	24000
37	PRIMA STICK 5kg	5	35000	60000

ALGORITMA GREEDY BERDASARKAN BERAT BARANG TERKECIL

Keterangan :

1 = dipilih

0 = tidak dipilih

Menurut perhitungan manual dari tabel disamping, dan perhitungan dengan code, dijelaskan bahwa pada strategi Greedy by weight diperoleh hasil keuntungan maksimum sebesar Rp 676.500,- dengan bobot 474 kg, dan pilihan barang yang diangkut adalah barang ke 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37.

Barang ke-i	w _i	p _i	Status (s)
7	3	3000	1
19	3	1500	1
33	5	10000	1
36	5	15000	1
2	10	25000	1
3	10	35000	1
4	10	15000	1
6	10	75000	1
8	10	25000	1
16	10	40000	1
17	10	35000	1
28	10	5500	1
29	10	1000	1
30	10	7500	1
31	10	5000	1
32	10	2500	1
35	10	12500	1
18	15	15000	1
34	15	11000	1
9	18	22500	1
1	20	7500	1
5	20	50000	1
14	20	90000	1
21	20	14000	1
22	20	44000	1
23	20	20000	1
24	20	20000	1
25	20	4000	1
26	20	4000	1
27	20	20000	1
12	12	10000	1
37	25	25000	1
20	30	6000	1
13	0	0	0
15	0	0	0
10	0	0	0
11	0	0	0
Total	474	676500	

Greedy By Weight

Hasil

jenis barang yang diangkut: [7, 19, 33, 36, 2, 3, 4, 6, 8, 16, 17, 28, 29, 30, 31, 32, 35, 18, 34, 9, 1, 5, 14, 21, 22, 23, 24, 25, 26, 27, 12, 37, 20]

jumlah beban yang diangkut: 474

jumlah keuntungan: 676500

ALGORITMA GREEDY BERDASARKAN KEUNTUNGAN TERBESAR

Keterangan :

1 = dipilih

0 = tidak dipilih

Dari perhitungan manual dari tabel disamping, dan perhitungan dengan code, dijelaskan bahwa pada strategi Greedy by profit diperoleh hasil keuntungan maksimum sebesar Rp 655.500,- dengan bobot 498 kg, dan pilihan barang yang diangkut adalah barang ke 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 21, 22, 23, 24, 27, 34, 35, 36, 37.

Execution time: 0.00021839141845703125 seconds

Greedy By Profit

Hasil

jenis barang yang diangkut: [14, 6, 5, 22, 16, 3, 17, 2, 8, 37, 9, 23, 24, 27, 13, 4, 18, 36, 21, 35, 34, 10, 11, 12]

jumlah beban yang diangkut: 498

jumlah keuntungan: 655000

Barang ke-i	w _i	p _i	Status (s)
14	20	90000	1
6	10	75000	1
5	20	50000	1
22	20	44000	1
16	10	40000	1
3	10	35000	1
17	10	35000	1
2	10	25000	1
8	10	25000	1
37	25	25000	1
9	18	22500	1
23	20	20000	1
24	20	20000	1
27	20	20000	1
13	40	16000	1
4	10	15000	1
18	15	15000	1
36	5	15000	1
21	20	14000	1
35	10	12500	1
34	15	11000	1
10	45	10000	1
11	90	10000	1
12	25	10000	1
33	0	0	0
1	0	0	0
30	0	0	0
20	0	0	0
28	0	0	0
31	0	0	0
15	0	0	0
25	0	0	0
26	0	0	0
7	0	0	0
32	0	0	0
19	0	0	0
29	0	0	0
Total	498	655500	

ALGORITMA GREEDY BERDASARKAN PERBANDINGAN KEUNTUNGAN DAN BERAT TERBESAR

Keterangan :

1 = dipilih

0 = tidak dipilih

Dari perhitungan manual dari tabel disamping, dan perhitungan dengan code, dijelaskan bahwa pada strategi Greedy by density diperoleh hasil keuntungan maksimum sebesar Rp 687.500,- dengan bobot 479 kg, dan pilihan barang yang diangkut adalah barang ke 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 27, 28, 30, 31, 32, 33, 34, 35, 36,

Barang ke- i	$\frac{p_i}{w_i}$	w_i	p_i	Status (s)
6	375	10	75000	1
14	2500	20	90000	1
16	3500	10	40000	1
3	1500	10	25000	1
17	2500	10	35000	1
36	7500	5	15000	1
2	1000	10	25000	1
5	2500	20	50000	1
8	1250	10	25000	1
22	222.22	20	44000	1
33	111.11	5	10000	1
4	400	10	15000	1
9	400	18	22500	1
35	4500	10	12500	1
7	100	3	3000	1
18	4000	15	15000	1
23	3500	20	20000	1
24	1000	20	20000	1
27	500	20	20000	1
37	200	25	25000	1
30	700	10	7500	1
34	2200	15	11000	1
21	1000	20	14000	1
28	1000	10	5500	1
19	200	3	1500	0
31	200	10	5000	0
12	1000	25	10000	0
13	550	40	16000	0
1	100	20	7500	0
32	750	10	2500	0
10	500	0	0	0
20	250	0	0	0
25	2000	0	0	0
26	733.00	0	0	0
11	1250	0	0	0
15	3000	0	0	0
29	1000	0	0	0
Total		479	687500	

Execution time: 0.0006425380706787109 seconds

Greedy By Density

Hasil

Jenis barang yang di angkut: 5, 13, 15, 2, 16, 35, 1, 4, 7, 21, 32, 3, 8, 34, 6, 17, 22, 23, 26, 36, 29, 33, 20, 27, 18, 30, 11, 12, 0, 31, 9

Jumlah beban yang di angkut: 479.0

Jumlah keuntungan: 687500.0

HASIL PERMASALAHAN BERDASARKAN DYNAMIC PROGRAMING

Tujuan dari Pemrograman Dynamic yaitu mencari total keuntungan optimal dari pengangkutan barang dimana total keuntungan optimal dipengaruhi oleh keuntungan dari barang yang dinotasikan dengan P_i

Untuk memperoleh total keuntungan optimal dilakukan perhitungan dengan prosedur rekursif maju. Perhitungan dimulai dengan mencari nilai keuntungan barang ke-1 lalu dilanjutkan mencari nilai keuntungan barang ke-dua dan seterusnya sampai barang ke-n

Dari keterangan disamping didapat hasil maksimal dengan perolehan maksimum sebesar Rp 691.500,- dengan bobot 499 kg

No	NAMA BARANG (i)	BERAT (w_i)	KEUNTUNGAN (p_i)
1	PRIMAFUR 2kg	20	7500
2	CALCIUM KUPU2 1kg	10	25000
3	CALCIUM KUMBANG 1kg	10	35000
4	CRASH 1ltr	10	15000
5	MKP KRISTA 1kg	20	50000
6	NITRAFOS 1kg	10	75000
7	NITRABOR 1kg	3	3000
8	CALCINIT 1kg	10	25000
9	KALI CHILI YARA 2kg	18	22500
10	PLASTIK MULSA 9kg	45	10000
12	PLASTIK 2KELINCI 5kg	25	10000
13	PLASTIK 2KELINCI 10kg	40	16000
14	AMEGRASS 5kg	20	90000
16	SIDAMIN 1kg	10	40000
17	SIDAFOS 1kg	10	35000
18	P21 1kg	15	15000
19	BISI 2 1kg	3	1500
21	INTANI 10kg	20	14000
22	NPK MUTIARA 5kg	20	44000
23	KCL SHS 5kg	20	20000
24	ZA 5kg	20	20000
25	UREA D/B 5kg	20	4000
27	NPK TAWON 5kg	20	20000
28	KNO.3 MERAH 2kg	10	5500
30	REGENT.03G 1kg	10	7500
31	ALPHADIN 1kg	10	5000
32	FURADAN 3G 2kg	10	2500
33	ZK 1kg	5	10000
34	MKP KNO.3 SAPRTAN 3kg	15	11000
35	FERRTERA 2kg	10	12500
36	PRIMA STICK 1kg	5	15000
37	PRIMA STICK 5kg	25	25000
JUMLAH		499	691500

RINGKASAN HASIL ALGORITMA GREEDY

Barang		Status Greedy by			Hasil optimal Greedy by		
<i>i</i>	<i>p_i</i>	<i>Weight</i>	<i>Profit</i>	<i>Density</i>	<i>Weight</i> <i>p_i</i>	<i>Profit</i> <i>p_i</i>	<i>Density</i> <i>p_i</i>
1	7500	1	0	1	7500	0	7500
2	25000	1	1	1	25000	25000	25000
3	35000	1	1	1	35000	35000	35000
4	15000	1	1	1	15000	15000	15000
5	50000	1	1	1	50000	50000	50000
6	75000	1	1	1	75000	75000	75000
7	3000	1	0	1	3000	0	3000
8	25000	1	1	1	25000	25000	25000
9	22500	1	1	1	22500	22500	22500
10	10000	0	1	0	0	10000	10000
11	10000	0	1	0	0	10000	0
12	10000	1	1	1	10000	10000	10000
13	16000	0	1	0	0	16000	16000
14	90000	1	1	1	90000	90000	90000
15	4000	0	0	0	0	0	0
16	40000	1	1	1	40000	40000	000
17	35000	1	1	1	35000	35000	000
18	15000	1	1	1	15000	15000	000
19	1500	1	1	1	1500	0	000
20	6000	1	1	1	6000	0	000
21	14000	1	1	1	14000	14000	000
22	44000	1	1	1	44000	44000	000
23	20000	1	1	1	20000	20000	000
24	20000	1	1	1	20000	20000	000
25	4000	1	1	1	4000	0	000
26	4000	1	1	1	4000	0	000
27	20000	1	1	1	20000	20000	000
28	5500	1	1	1	5500	0	000
29	1000	1	1	1	1000	0	000
30	7500	1	1	1	7500	0	000
31	5000	1	1	1	5000	0	000
32	2500	1	1	1	2500	0	000
33	10000	1	1	1	10000	0	000
34	11000	1	1	1	11000	11000	000
35	12500	1	1	1	12500	12500	000
36	15000	1	1	1	15000	15000	000
37	25000	1	1	1	25000	25000	25000
Total					676500	655500	687500

DATA MENTAH AWAL.

VS

HASIL PILIHAN BARANG MENGGUNAKAN GREEDY

No	Nama Barang	Banyak Barang	Harga Beli	Harga Jual
1	PRIMAFUR 2kg	10	16500	24000
2	CALCIUM KUPU2 1kg	10	12500	37500
3	CALCIUM KUMBANG 1kg	10	8500	43500
4	CRASH 1 ltr	10	39500	54500
5	MKP KRISTA 1kg	20	22500	73500
6	NITRAFOS 1kg	10	22500	97500
7	NITRABOR 1kg	3	14000	17000
8	CALCINIT 1kg	10	12500	37500
9	KALI CHILI YARA 2kg	9	19500	42000
10	PLASTIK MULSA 9kg	5	24000	34000
11	PLASTIK MULSA 18kg	5	24000	34000
12	PLASTIK 2KELINCI 5kg	5	95000	105000
13	PLASTIK 2KELINCI 10kg	4	178000	194000
14	AMEGRASS 5kg	4	317500	407500
15	AMEGRASS 20kg	2	63500	67000
16	SIDAMIN 1kg	10	46000	86000
17	SIDAFOS 1kg	10	31500	66500
18	P21 1kg	15	56000	71000
19	BISI 2 1kg	3	42000	43500
20	CIHERANG SS 10kg	3	73000	79000
21	INTANI 10kg	2	73000	87000
22	NPK MUTIARA 5kg	4	339000	383000
23	KCL SHS 5kg	4	255000	275000
24	ZA 5kg	4	70000	90000
25	UREA D/B 5kg	4	89000	93000
26	PHONSKA 5kg	4	115000	119000
27	NPK TAWON 5kg	4	325000	345000
28	KNO.3 MERAH 2kg	5	18900	24400
29	KNO.3 PUTIH 2kg	5	27800	28800
30	REGENT.03G 1kg	10	16250	23750
31	ALPHADIN 1kg	10	12500	17500
32	FURADAN 3G 2kg	5	19500	22000
33	ZK 1kg	5	16000	26000
34	MKP KNO.3 SAPRTAN 3kg	5	22800	33800
35	FERRTERA 2kg	5	77500	90000
36	PRIMA STICK 1kg	5	9000	24000
37	PRIMA STICK 5kg	5	35000	60000

No	Nama Barang	Berat	Keuntungan
1	PRIMAFUR 2kg	20	7500
2	CALCIUM KUPU2 1kg	10	25000
3	CALCIUM KUMBANG 1kg	10	35000
4	CRASH 1 ltr	10	15000
5	MKP KRISTA 1kg	20	50000
6	NITRAFOS 1kg	10	75000
7	NITRABOR 1kg	3	3000
8	CALCINIT 1kg	10	25000
9	KALI CHILI YARA 2kg	18	22500
10	PLASTIK MULSA 9kg	45	10000
12	PLASTIK 2KELINCI 5kg	25	10000
13	PLASTIK 2KELINCI 10kg	40	16000
14	AMEGRASS 5kg	20	90000
16	SIDAMIN 1kg	10	40000
17	SIDAFOS 1kg	10	35000
18	P21 1kg	15	15000
19	BISI 2 1kg	3	1500
21	INTANI 10kg	20	14000
22	NPK MUTIARA 5kg	20	44000
23	KCL SHS 5kg	20	20000
24	ZA 5kg	20	20000
27	NPK TAWON 5kg	20	20000
28	KNO.3 MERAH 2kg	10	5500
30	REGENT.03G 1kg	10	7500
31	ALPHADIN 1kg	10	5000
32	FURADAN 3G 2kg	10	2500
33	ZK 1kg	5	10000
34	MKP KNO.3 SAPRTAN 3kg	15	11000
35	FERRTERA 2kg	10	12500
36	PRIMA STICK 1kg	5	15000
37	PRIMA STICK 5kg	25	25000
JUMLAH		479	687500

HASIL PERMASALAHAN BERDASARKAN DYNAMIC PROGRAMING

Untuk memperoleh total keuntungan optimal dilakukan perhitungan dengan prosedur rekursif maju. Perhitungan dimulai dengan mencari nilai keuntungan barang ke-1 lalu dilanjutkan mencari nilai keuntungan barang ke-dua dan seterusnya sampai barang ke-n

Dari keterangan disamping didapat hasil maksimal dengan perolehan maksimum sebesar Rp 691.500,- dengan bobot 499 kg

No	NAMA BARANG (i)	BERAT (w_i)	KEUNTUNGAN (p_i)
1	PRIMAFUR 2kg	20	7500
2	CALCIUM KUPU2 1kg	10	25000
3	CALCIUM KUMBANG 1kg	10	35000
4	CRASH 1ltr	10	15000
5	MKP KRISTA 1kg	20	50000
6	NITRAFOS 1kg	10	75000
7	NITRABOR 1kg	3	3000
8	CALCINIT 1kg	10	25000
9	KALI CHILI YARA 2kg	18	22500
10	PLASTIK MULSA 9kg	45	10000
12	PLASTIK 2KELINCI 5kg	25	10000
13	PLASTIK 2KELINCI 10kg	40	16000
14	AMEGRASS 5kg	20	90000
16	SIDAMIN 1kg	10	40000
17	SIDAFOS 1kg	10	35000
18	P21 1kg	15	15000
19	BISI 2 1kg	3	1500
21	INTANI 10kg	20	14000
22	NPK MUTIARA 5kg	20	44000
23	KCL SHS 5kg	20	20000
24	ZA 5kg	20	20000
25	UREA D/B 5kg	20	4000
27	NPK TAWON 5kg	20	20000
28	KNO.3 MERAH 2kg	10	5500
30	REGENT.03G 1kg	10	7500
31	ALPHADIN 1kg	10	5000
32	FURADAN 3G 2kg	10	2500
33	ZK 1kg	5	10000
34	MKP KNO.3 SAPRTAN 3kg	15	11000
35	FERRTERA 2kg	10	12500
36	PRIMA STICK 1kg	5	15000
37	PRIMA STICK 5kg	25	25000
JUMLAH		499	691500

Dynamic Programming Table:

```
Items taken (1 means taken, 0 means not taken): [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1]
Total profit: 691500
Total weight of items taken: 499
```

PERBANDINGAN ALGORITMA GREEDY DAN DYNAMIC PROGRAMMING PADA PERMASALAHAN KNAPSACK

Solusi Optimal dari Perhitungan Kedua Algoritma

No	Algoritma	Pilihan barang	Keuntungan
1	<i>Greedy</i>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37	Rp 687.500,-
2	<i>Dynamic Programming</i>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37	Rp 691.500,-

diketahui bahwa hasil perhitungan dengan menggunakan algoritma Greedy menghasilkan keuntungan yang lebih kecil dibandingkan dengan nilai keuntungan yang dihasilkan oleh algoritma Dynamic Programming. Artinya, penggunaan algoritma Dynamic Programming lebih optimal jika diaplikasikan pada permasalahan knapsack ini



HASIL PILIHAN BARANG MENGGUNAKAN GREEDY

No	Nama Barang	Berat	Keuntungan
1	PRIMAFUR 2kg	20	7500
2	CALCIUM KUPU2 1kg	10	25000
3	CALCIUM KUMBANG 1kg	10	35000
4	CRASH 1 ltr	10	15000
5	MKP KRISTA 1kg	20	50000
6	NITRAFOS 1kg	10	75000
7	NITRABOR 1kg	3	3000
8	CALCINIT 1kg	10	25000
9	KALI CHILI YARA 2kg	18	22500
10	PLASTIK MULSA 9kg	45	10000
12	PLASTIK 2KELINCI 5kg	25	10000
13	PLASTIK 2KELINCI 10kg	40	16000
14	AMEGRASS 5kg	20	90000
16	SIDAMIN 1kg	10	40000
17	SIDAFOS 1kg	10	35000
18	P21 1kg	15	15000
19	BISI 2 1kg	3	1500
21	INTANI 10kg	20	14000
22	NPK MUTIARA 5kg	20	44000
23	KCL SHS 5kg	20	20000
24	ZA 5kg	20	20000
27	NPK TAWON 5kg	20	20000
28	KNO.3 MERAH 2kg	10	5500
30	REGENT.03G 1kg	10	7500
31	ALPHADIN 1kg	10	5000
32	FURADAN 3G 2kg	10	2500
33	ZK 1kg	5	10000
34	MKP KNO.3 SAPRTAN 3kg	15	11000
35	FERRTERA 2kg	10	12500
36	PRIMA STICK 1kg	5	15000
37	PRIMA STICK 5kg	25	25000
JUMLAH		479	687500

HASIL PILIHAN BARANG MENGGUNAKAN DYNAMIC

No	Nama Barang	Berat	Keuntungan
1	PRIMAFUR 2kg	20	7500
2	CALCIUM KUPU2 1kg	10	25000
3	CALCIUM KUMBANG 1kg	10	35000
4	CRASH 1 ltr	10	15000
5	MKP KRISTA 1kg	20	50000
6	NITRAFOS 1kg	10	75000
7	NITRABOR 1kg	3	3000
8	CALCINIT 1kg	10	25000
9	KALI CHILI YARA 2kg	18	22500
10	PLASTIK MULSA 9kg	45	10000
12	PLASTIK 2KELINCI 5kg	25	10000
13	PLASTIK 2KELINCI 10kg	40	16000
14	AMEGRASS 5kg	20	90000
16	SIDAMIN 1kg	10	40000
17	SIDAFOS 1kg	10	35000
18	P21 1kg	15	15000
19	BISI 2 1kg	3	1500
21	INTANI 10kg	20	14000
22	NPK MUTIARA 5kg	20	44000
23	KCL SHS 5kg	20	20000
24	ZA 5kg	20	20000
25	UREA D/B 5kg	20	4000
27	NPK TAWON 5kg	20	20000
28	KNO.3 MERAH 2kg	10	5500
30	REGENT.03G 1kg	10	7500
31	ALPHADIN 1kg	10	5000
32	FURADAN 3G 2kg	10	2500
33	ZK 1kg	5	10000
34	MKP KNO.3 SAPRTAN 3kg	15	11000
35	FERRTERA 2kg	10	12500
36	PRIMA STICK 1kg	5	15000
37	PRIMA STICK 5kg	25	25000
JUMLAH		499	691500

ANALISIS BIG O

- *Greedy by Weight*

Langkah-langkah analisis Big O dari kode *Greedy by Weight*

1. Tentukan variabel dan parameter:
 - `n`: Jumlah barang
 - `m`: Jumlah matriks
 - `M`: Batasan berat
 - `profit`: Array keuntungan setiap barang
 - `weight`: Array berat setiap barang
2. Periksa apakah semua barang dapat terangkut (jika batas berat lebih besar dari total berat barang). Jika ya, keluarkan hasil dan selesai.
3. Jika tidak semua barang dapat terangkut, periksa apakah jumlah elemen dalam matriks profit dan weight sama. Jika tidak, keluarkan pesan kesalahan.
4. Jika jumlah elemen sama, jalankan Greedy by Weight:
 - Buat list indeks barang.
 - Urutkan barang berdasarkan beratnya.
 - Inisialisasi variabel berat dan list barang yang dipilih.
 - Iterasi melalui barang-barang yang diurutkan, tambahkan barang ke dalam list dipilih jika memenuhi batasan berat.
5. Hitung waktu eksekusi dan keluarkan hasilnya, termasuk barang yang terpilih, berat yang diangkut, dan total keuntungan.
6. Analisis kompleksitas waktu (Big O):
 - Jumlah perhitungan dalam loop mencari barang yang dapat dimuat memiliki kompleksitas $O(n)$.
 - Proses pemilihan berdasarkan batasan berat memiliki kompleksitas $O(n \log n)$ karena proses pengurutan barang.

Dengan demikian, kompleksitas waktu totalnya adalah $O(n \log n)$.



ANALISIS BIG O

- *Greedy by Profit*

Langkah-langkah analisis Big O dari kode *Greedy by Profit*:

1. Tentukan variabel dan parameter:
 - `n`: Jumlah barang
 - `m`: Jumlah matriks
 - `M`: Batasan berat
 - `profit`: Array keuntungan setiap barang
 - `weight`: Array berat setiap barang
2. Buat salinan informasi awal untuk keperluan penghitungan kompleksitas waktu.
3. Buat list indeks barang dan urutkan barang berdasarkan keuntungan (descending).
4. Inisialisasi variabel berat dan list barang yang dipilih.
5. Iterasi melalui barang-barang yang diurutkan, tambahkan barang ke dalam list dipilih jika memenuhi batasan berat.
6. Hitung waktu eksekusi dan keluarkan hasilnya, termasuk barang yang terpilih, berat yang diangkut, dan total keuntungan.
7. Analisis kompleksitas waktu (Big O):
 - Jumlah perhitungan dalam loop mencari barang yang dapat dimuat memiliki kompleksitas $O(n)$.
 - Proses pemilihan berdasarkan batasan berat memiliki kompleksitas $O(n \log n)$ karena proses pengurutan barang.

Dengan demikian, kompleksitas waktu totalnya adalah $O(n \log n)$.



ANALISIS BIG O

- *Greedy by Density*

Langkah-langkah analisis Big O dari kode *Greedy by Density*

1. Tentukan variabel dan parameter:
 - `n`: Jumlah barang
 - `m`: Jumlah matriks
 - `M`: Batasan berat
 - `profit`: Array keuntungan setiap barang
 - `weight`: Array berat setiap barang
2. Konversi `weight` dan `profit` ke array NumPy untuk kemudahan pengolahan.
3. Hitung rasio keuntungan-berat (`rasio`) untuk setiap barang.
4. Urutkan rasio keuntungan-berat secara descending dan simpan urutan awalnya (`urutan`).
5. Rekonstruksi array keuntungan (`P`) dan array berat (`W`) berdasarkan urutan dari rasio terurut.
6. Iterasi melalui barang-barang terurut, tambahkan berat dan keuntungan hingga batasan berat tercapai atau semua barang telah dipertimbangkan.
7. Hitung waktu eksekusi dan keluarkan hasilnya, termasuk barang yang terpilih, berat yang diangkut, dan total keuntungan.
8. Analisis kompleksitas waktu (Big O):
 - Perhitungan rasio keuntungan-berat memiliki kompleksitas $O(n)$.
 - Proses pengurutan rasio memiliki kompleksitas $O(n \log n)$.
 - Iterasi melalui barang terurut memiliki kompleksitas $O(n)$.
 - Sehingga, kompleksitas waktu totalnya adalah $O(n \log n)$.

Dengan demikian, kompleksitas waktu total dari algoritma *Greedy by Density* adalah $O(n \log n)$.



ANALISIS BIG O

- *Dynamic Programming*

Berikut langkah-langkah analisis Big O dari kode yang menggunakan algoritma Dynamic Programming untuk masalah Knapsack:

1. Tentukan variabel dan parameter:
 - `'n'`: Jumlah barang
 - `'m'`: Jumlah matriks
 - `'M'`: Batasan berat
 - `'profit'`: Array keuntungan setiap barang
 - `'weight'`: Array berat setiap barang
 - `'z'`: List 2D untuk menyimpan tabel dynamic programming
2. Inisiasi tabel dynamic programming (`'z'`) dengan ukuran `'(n + 1) x (m + 1)'`.
3. Iterasi melalui setiap item dan kapasitas untuk mengisi tabel dynamic programming dengan menggunakan rumus:
 - Jika bobot item dapat muat ke kapasitas knapsack, pilih maksimum antara tidak mengambil item dan mengambil item.
 - Jika bobot item tidak muat, keuntungan maks tetap sama seperti tidak mengambil item.
4. Hitung waktu eksekusi dan keluarkan hasilnya.
5. Loop untuk mencari solusi optimal:
 - Hitung total keuntungan (`'total_profit'`) yang dapat dicapai dengan mempertimbangkan semua item.
 - Inisiasi total bobot item yang diambil (`'total_weight'`) dan daftar item yang diambil (`'items_taken'`).
6. Loop mundur untuk mengidentifikasi item apa saja yang diambil atau tidak diambil.
7. Hitung total bobot item yang diambil dan keluarkan hasilnya.
8. Analisis kompleksitas waktu (Big O):
 - -Dua loop bersarang: $O(n * m)$.
 - Jadi, kompleksitas waktu totalnya adalah $O(n * m)$.

Dengan demikian, kompleksitas waktu dari algoritma Dynamic Programming untuk masalah Knapsack adalah $O(n * m)$.



ANALISIS WAKTU

Dari hasil waktu eksekusi yang diberikan:

1. *Greedy By Weight*: 1.69277e-05 seconds
2. *Greedy By Profit*: 0.00016737 seconds
3. *Greedy By Density*: 0.00033855 seconds
4. *Dynamic Programming*: 0.02177 seconds

Dapat diamati bahwa:

- *Greedy By Weight* memiliki waktu eksekusi yang paling singkat.
- *Greedy By Profit* memiliki waktu eksekusi yang lebih lama dibandingkan *Greedy By Weight*, namun masih cukup efisien.
- *Greedy By Density* memiliki waktu eksekusi yang lebih lama dibandingkan kedua metode sebelumnya, tetapi masih lebih cepat dibanding *Dynamic Programming*.
- *Dynamic Programming* memiliki waktu eksekusi yang paling lama di antara keempatnya.

Hal ini mencerminkan trade-off antara kecepatan dan optimalitas. Metode *Greedy* cenderung lebih cepat tetapi mungkin tidak memberikan solusi optimal, sedangkan *Dynamic Programming* memastikan solusi optimal namun memerlukan lebih banyak waktu komputasi. *Greedy By Density* berada di antara keduanya.



ANALISIS MEMORI

Analisis Memori

1. *Greedy By Weight, Greedy By Profit, Greedy By Density:*

- Penggunaan memori terkait dengan array `profit`, `weight`, dan variabel seperti `items`, `selected_items`, `berat`, dll.
- Array tambahan seperti `rasio` pada *Greedy By Density* juga mempengaruhi penggunaan memori.
- Secara umum, memori yang digunakan oleh algoritma Greedy biasanya lebih kecil karena tidak memerlukan penyimpanan tabel atau matriks seperti pada Dynamic Programming.

2. Dynamic Programming:

- Memori utama terkait dengan matriks `z`, yang memiliki ukuran `(n + 1) x (m + 1)`.
- Tambahan array dan variabel seperti `y`, `total_profit`, `total_weight`, dan `items_taken` juga mempengaruhi penggunaan memori.
- Dynamic Programming cenderung menggunakan lebih banyak memori karena menyimpan informasi untuk setiap kombinasi dari setiap item dan kapasitas.



TERIMA KASIH

