PEMROGRAMAN DINAMIK

Pemrograman dinamik menyelesaikan permasalahan dengan cara menyelesaikan sub-sub permasalahan, kemudian mengabungkannya menjadi penyelesaian permasalahan tersebut. Hal ini mirip dengan metoda *divide-and-conquer*. Pemrograman dinamik sering diterapkan pada permasalahan optimasi, yang mana mempunyai banyak kemungkinan penyelesaian.

Algoritma pemrograman dinamik dapat dibagai kedalam 4 langkah:

1. Menentukan struktur dari suatu penyelesaian optimal.
2. Mendefinisikan secara rekursif nilai dari penyelesaian optimal.
3. Menghitung nilai dari penyelesaian optimal dengan cara bottom-up. Langkah 1 sampai 3 ini merupakan basis dari penyelesaian pemrograman dinamik. Langkah 4 dapat diabaikan apabila hanya diperlukan sebuah solusi optimal.
4. Membentuk suatu penyelesaian optimal dari informasi yang didapat.

Lebih jelasnya kita lihat permasalahan pemrograman dinamik yaitu: perkalian matriks-berantai sehingga total operator perkalian dua skalar yang dilakukannya paling sedikit.

1. Perkalian Matriks-Berantai.

Permasalahan perkalian matriks-berantai adalah mengalikan sederetan matriks $A_1, A_2, ..., A_n$. Ekspresi perkalian tersebut dapat dinyatakan dalam bentuk:

$A_1 A_2 ... A_n$.

Dengan memanfaatkan sifat asosiatif untuk perkalian matriks, kita dapat mengalikan $A_1A_2A_3A_4$ dalam beberapa cara sebagai berikut.

$(A_1(A_2(A_3A_4)))$,
$(A_1((A_2A_3)A_4))$,
$((A_1A_2)(A_3A_4))$,
$((A_1(A_2A_3))A_4)$,
$(((A_1A_2)A_3)A_4)$.

Untuk mengalikan dua buah matriks kita dapat menggunakan algoritma standar pengalian dua buah matriks sebagai berikut.

```
PerkalianMatrik(A,B) {
if ( kolom(A) <> baris(B) )
    return "Kedua matriks tidak kompatibel"
else {
    for i=1 to baris(A) {
         for j=1 to kolom(B) {
               C[i][j] = 0;
               for k=1 to kolom(A) {
                   C[i][j] = C[i][j] + A[i][k] B[k][j];
               }
         }
     }
     return C;
}}
```

Jika matriks *A* berukuran *p* x *q* dan matriks *B* berukuran *q* x *r*, maka algoritma di atas melakukan perkalian skalar sebanyak *pqr*.

Kita akan lihat melalui contoh bahwa, urutan cara pengalian akan mengakibatkan biaya perkalian skalar yang berbeda. Misal matriks A_1 berukuran 10 x 100, matriks A_2 berukuran 100 x 5, dan matriks A_3 berukuran 5 x 50. Kita dapatkan kemungkinan cara urutan perkalian dan banyaknya operasi perkalian skalar sebagai berikut.

Alternatif	Urutan perkalian	Banyaknya operasi perkalian skalar
1	$(A_1(A_2A_3))$	100 x 5 x 50 + 10 x 100 x 50 = 75.000
2	$((A_1A_2)A_3)$	10 x 100 x 5 + 10 x 5 x 50 = 7.500

Alternatif 1 memerlukan operasi perkalian sebanyak 75.000, sedangkan alternatif ke dua memerlukan perkalian 7.500. Sudah barang tentu kita memilih yang alternatif ke dua.

- **Banyaknya Pengurungan**

Sebelum kita menyelasaikan permasalahan perkalian matriks-berantai ini menggunakan pemrograman dinamik, kita harus yakin terlebih dahulu bahwa mengecek semua kemungkinan pengurungan tidak akan menghasilkan algoritma yang efisien. Untuk sederetan *n* matriks, banyaknya alternatif pengurungan kita notasikan dengan *P(n)*. Kita dapat memecah sederetan *n* matriks antara matriks ke *k* dan matriks ke *k+1*, untuk *k = 1, 2, ..., n-1*, dan kemudian mengurung dua subderetan yang dihasilkan secara independen. Kita lihat untuk *n=6.*

$(A_1)(A_2A_3A_4A_5A_6)$ ==>> $p(1)p(6-1)$

$(A_1A_2)(A_3A_4A_5A_6)$ ==>> $p(2)p(6-2)$

$(A_1A_2A_3)(A_4A_5A_6)$ ==>> $p(3)p(6-3)$

$(A_1A_2A_3A_4)(A_5A_6)$ ==>> $p(4)p(6-4)$

$(A_1A_2A_3A_4A_5)(A_6)$ ==>> $p(5)p(6-5)$

$p(6) = p(1)p(6-1) + p(2)p(6-2) + p(3)p(6-3) + p(4)p(6-4) + p(5)p(6-5)$

Dengan demikian kita dapatkan persamaan rekursif sebagai berikut.

$$p(n) = \begin{cases} 1 & jika\ n = 1 \\ \sum_{k=1}^{n} p(k)\, p(n-k) & jika\ n \geq 2 \end{cases}$$

Penyelesaian daripersamaan rekursif di atas adalah *P(n) = C(n-1)*, dengan *C(n)* adalah bilangan *Catalan*, sebagai berikut.

$$C(n) = \frac{1}{n-1}\binom{2n}{n} = \frac{1}{n-1}\frac{(2n)!}{n!\,(2n-n)!}$$

$$C(n) = \frac{1}{n-1}\frac{2n(2n-1)(2n-2)(2n-3)\ldots 3.2.1}{n(n-1)(n-2)\ldots 3.2.1\,\{n(n-1)(n-2)\ldots 3.2.1\}}$$

$$C(n) = \frac{1}{n-1}\frac{2n(2n-1)(2n-2)\dots(2n-(n-1))}{n(n-1)(n-2)\dots 3.2.1}$$

$$C(n) = O\left(\frac{4^n}{n^{3/2}}\right)$$

Terlihat persamaan rekurensi di atas mempunyai penyelesaian exponensial dalam *n*. Dengan kata lain untuk mencari banyaknya alternatif pengurungan diperlukan biaya yang tinggi yaitu: exponensial.

- **Struktur dari suatu pengurungan optimal**

Langkah pertama dari pemrograman dinamik adalah menentukan struktur dari suatu penyelesaian optimal. Untuk permasalahan perkalian matriks-berantai adalah sebagai berikut.

Misal $A_{i..j}$ merupakan notasi untuk matriks hasil perkalian $A_i A_{i+1} \dots A_j$. Suatu penyelesaian optimal dari perkalian $A_1 A_2 \dots A_n$ memisah perkalian antara A_k dengan A_{k+1}, untuk $1 \le k < n$. Ini berarti bahwa untuk suatu nilai *k*, kita hitung dahulu $A_{1..k}$ dan $A_{k+1..n}$, kemudian mengalikan keduanya untuk menghasilkan $A_{1.n}$. Biaya dari pengurungan ini sama dengan biaya untuk menghitung $A_{1..k}$ ditambah biaya untuk menghitung $A_{k+1..n}$, ditambah lagi biaya untuk mengalikan keduanya.

Untuk mendapatkan hasil pengurungan optimal dari $A_1 A_2 \dots A_n$ didapat dari pengurungan $A_1 A_2 \dots A_k$ yang optimal dan pengurungan $A_{k+1} A_{k+2} \dots A_n$ yang optimal.

- **Penyelesaian Rekursif**

Langkah ke dua dari pemrograman dinamik adalah mendefinisikan secara rekursif nilai dari penyelesaian optimal dalam bentuk penyelesaian optimal dari sub permasalahan.

Misal *m[i,j]* merupakan minimum banyaknya perkalian skalar (minimum cost) yang diperlukan untuk menghitung $A_{i..j}$. Sehingga biaya minimum untuk menghitung $A_{1..n}$ adalah *m[1,n]*.

Rumusan *m[i,j]* akan kita definisikan sebagai berikut. Jika *i=j*, maka hanya ada sebuah matrik yaitu A_i. Sehingga tidak diperlukan perkalian skalar. Oleh karena itu *m[i,i]=0* untuk *i=1, 2, ..., n.* Jika $i < j$, maka kita akan memanfaatkan struktur optimal pada langkah 1. Kita misalkan pengurungan optimal dari $A_i A_{i+1} \dots A_j$ memisah A_k dan A_{k+1} untuk $i \le k < j$. Maka dari itu didapatkan:

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j \ .$$

Pada persamaan rekursif di atas kita asumsikan mengetahui nilai *k*, pada hal nilai *k* ini tidak diketahui. Akan tetapi nilai *k* hanya ada *j-i* kemungkinan, yaitu *k=i, i+1, ... , j-1.* Karena pengurungan optimal harus mengunakan salah satu dari nilai *k* di atas, maka kita hanya perlu melakukan pengecekan pada daerah *k* ini. Sehingga definisi untuk biaya minimum pengurungan $A_1 A_2 \dots A_n$ menjadi:

$$m[i,j] = \begin{cases} 0, & \text{jika } i = j \\ \min\limits_{i \le k < j} m[i,k] + m[k+1,j] + p_{i-1}p_k p_j & \text{jika } i < j \end{cases}$$

Nilai-nilai *m[i,j]* diatas memberikan penyelesaian optimal untuk sub permasalahan. Misal didefinisikan *s[i,j]* adalah sebuah nilai *k* yang mana kita dapat memisah $A_i A_{i+1} \ldots A_j$ untuk mendapatkan pengurungan optimal.

Dengan kata lain, $s[i,j] = k$ sehingga $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$.

- **Penghitungan biaya optimal**

Langkah ke tiga dari pemrograman dinamik adalah menghitung nilai dari penyelesaian optimal dengan cara bottom-up. Hal ini sama dengan artinya menghitung persamaan recursif untuk *m[i,j]* di atas.

Algoritma dibawah ini memakai matriks A_i dengan ukuran p_{i-1} x p_i untuk $i = 1, 2, \ldots, n$.

Input : Sederetan $<p_0, p_1, \ldots, p_n>$ atau disingkat *p* dimana *length(p)=n+1*.

Output : Array *m[1..n, 1..n]* untuk menyimpan nilai *m[i,j]* dan array *s[1..n, 1..n]* untuk menyimpan indek *k* yang mana saat penghitungan *m[i,j]* didapatkan hasil optimal.

Algoritma:

```
UrutanMatrik-Berantai(p)
   n=length(p)-1;
   for i=1 to n do
       m[i,i]=0;
   for h=2 to n do
       for i=1 to n-h+1 do
           j=i+h-1;
           m[i,j] = ∞;
           for k=i to j-1 do
               q= m[i,k] + m[k+1,j] + p_{i-1}p_k p_j;
               if (q <m[i,j] ) then
                   m[i,j]=q;
                   s[i,j]=k;
               endif;
           endfor;
       endfor;
   endfor;
end; UrutanMatriks-Berantai.
```

Untuk melihat cara kerja algoritma diatas mari kita lihat contoh untuk perkalian sederetan 6 matriks $A_1 A_2 A_3 A_4 A_5 A_6$, dengan ukuran matrik sebagai berikut.

Matriks	Ukuran
A_1	30 x 35
A_2	35 x 15
A_3	15 x 5
A_4	5 x 10
A_5	10 x 20

A_6	20 x 25					
m	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0
s	1	2	3	4	5	6
1		1	1	3	3	3
2			2	3	3	3
3				3	3	
4					4	5
5						5
6						

Algoritma di atas memerlukan paling banyak n^3 perkalian skalar, dan memerlukan *space* sebesar n^2 untuk meyimpan *m* dan *s.*

- **Membentuk suatu penyelesaian optimal**

  Algoritma *urutanMatrik-Berantai* di atas hanya menentukan banyaknya perkalian skalar yang optimal pada perkalian matriks-berantai, namun tidak menentukan bagaimana perkalian matriknya. Untuk itu, kita masuk ke langkah 4 yaitu:membentuk suatu penyelesaian optimal dari informasi yang didapat.

  Tabel *s* kita pakai untuk menentukan cara terbaik untuk mengalikan matriks. Elemen *s[i,j]* memuat nilai *k* sehingga pengurungan optimal dari perkalian $A_i\ A_{i+1}\ ...\ A_j$ memisah A_k dan A_{k+1}. Sehingga untuk menghitung hasil akhir $A_{1..n}$ secara optimal dilakukan pemisahan perkalian $A_{1..s[1,n]}A_{s[1,n]+1..n}$. Algoritma berikut ini menghitung hasil perkalian matrik berantai $A_{i..j}$ dengan input sederetan matriks $A=<A_1,A_2, ...\ A_n>$, tabel *s* hasil dari algoritma *urutanMatriks-Berantai*, dan indeks *i*, dan *j*.

```
PerkalianMatrik-Berantai(A,s,i,j)
   if (i > j) then
       X = PerkalianMatrik-Berantai(A,s,i,s[i,j]);
       Y = PerkalianMatrik-Berantai(A,s,s[i,j]+1,j);
       return PerkalianMatriks(X,Y);
   else return Ai;
```

  Pemanggilan awal dari algoritma ini adalah *PerkalianMatrik-Berantai(A,s,1,n),* untuk contoh sebelumnya dipanggil dengan *PerkalianMatrik-Berantai(A,s,1,6)* akan melakukan penghitungan perkalian menurut pengurungan $((A_1(A_2A_3))((A_4,A_5)A_6))$.

2. Pengalokasian Uang ke Perusahaan

Permasalahan

Bagaimana mnginvestasikan $b ke dalam n buah perusahaan, bila diketahui keuntungan atau hasil dari setiap perusahaan. Permasalahan ini akan kita formulasikan dengan menggunakan **pemrograman dinamik.**

x_i = besarnya $ yang diinvestkan pada perusahaan ke i

$f_i(x_i)$ = hasil atau keuntungan yang didapat dari perusahaan ke I bila diinvestkan $ x_i ke perusahaan tersebut.

Permasalahan diatas dapat juga ditulis dalam bentuk:

$$\max \; f_1(x_1) + f_2(x_2) + ... + f_n(x_n)$$

dengan konstrain $x_1 + x_2 + ... + x_n \le b$

$x_i \ge 0$ & integer

$f_i(0) = 0$

1. Struktur optimal

- Menentukan x_i secara tepat, sebagai gambaran :

  misal dipunyai $1000 yang akan diinvestkan ke dalam 10 perusahaan dan seandainya sudah menginvestkan $450 ke dalam perusahaan 1,2,3 dan 4 sisa $550 harus diinvestkan ke perusahaan 6,7,8,9,10 secara optimal. Kalau tidak, hasil yang didapat tidak optimal. Dari sini sudah kelihatan bahwa sub permasalahannya, yaitu : “Harus memutuskan untuk menggunakan sisa $550 secara optimal”.

- Sekarang buat symbol :

  $F_k(x)$ = hasil max yang di dapat dari menginvestkan $x ke dalam k buah perusahaan yang pertama.

  $F_n(b)$ = Dicari / ditanyakan

2. Formulasi secara rekursif

$F_1(x) = f_1(x)$

$F_2(x)$ = Ditanya (harus memutuskan untuk diinvestkan ke perusahaan ke-2 sebesar berapa?

Misal x_2 diinvestkan ke perusahaan ke-2, sehingga ada sisa $x - x_2$ harus ditaruh di perusahaan yang pertama secara optimal.

Nilai x_2 bisa 0,1,2,…,x

$$F_2(x) = \max_{0 \le x_2 \le x} \{f_2(x_2) + F_1(x - x_2)\}$$

Dengan cara yang sama :

$$F_3(x) = \max_{0 \le x_3 \le x} \{f_3(x_3) + F_2(x - x_3)\}$$

$$F_4(x) = \max_{0 \le x_4 \le x} \{f_4(x_4) + F_3(x - x_4)\}$$

M

M

$$F_n(x) = \max_{0 \le x_n \le x} \{f_n(x_n) + F_{n-1}(x - x_n)\}$$

Contoh soal:

Dipunyai \$5 untuk diinvestkan pada perusahaan 1,2,3,4 hasil dari invest di masing-masing perusahaan adalah sebagai berikut :

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
1	11	0	2	20
2	12	5	10	21
3	13	10	30	22
4	14	15	32	23
5	15	20	40	24

Tentukan besarnya uang yang harus diinvestkan di tiap perusahaan secara optimal $F_4(5)$?

F k x	$F_1(x)$	x_1	$F_2(x)$	x_2	$F_3(x)$	x_3	$F_4(x)$	x_4
1	11	1	11	0	11	0	20	1
2	12	2	12	0	13	1	31	1
3	13	3	16	2	30	3	33	1
4	14	4	21	3	41	3	50	1
5	15	5	26	4	43	4	61	1

$$F_1(x) = f_1(x) \rightarrow x_1 = x$$

$$F_2(1) = \max_{0 \le x_2 \le 1} \{ f_2(x_2) + f_1(1 - x_2) \}$$
$$= \{ f_2(0) + f_1(1); f_2(1) + f_1(0) \}$$
$$= \{ 0 + 11; 0 + 0 \} = 11 \rightarrow x_2 = 0$$

$$F_2(2) = \max_{0 \le x_2 \le 2} \{ f_2(x_2) + f_1(2 - x_2) \}$$
$$= \{ f_2(0) + f_1(2); f_2(1) + f_1(1); f_2(2) + f_1(0) \}$$
$$= \{ 0 + 12; 0 + 11; 5 + 0 \} = 12 \rightarrow x_2 = 0$$

$$F_2(3) = \max_{0 \le x_2 \le 3} \{ f_2(x_2) + f_1(3 - x_2) \}$$
$$= \{ f_2(0) + f_1(3); f_2(1) + f_1(2); f_2(2) + f_1(1); f_2(3) + f_1(0) \}$$
$$= \{ 0 + 13 ; 0 + 12 ; 5 + 11 ; 10 + 0 \} = 16 \rightarrow x_2 = 2$$

$$F_2(4) = \max_{0 \le x_2 \le 4} \{ f_2(x_2) + f_1(4 - x_2) \}$$
$$= \{ f_2(0) + f_1(4); f_2(1) + f_1(3); f_2(2) + f_1(2); f_2(3) + f_1(1); f_2(4) + f_1(0) \}$$
$$= \{ 0 + 14 ; 0 + 13 ; 5 + 12 ; 10 + 11 ; 15 + 0 \} = 21 \rightarrow x_2 = 3$$

$$F_2(5) = \max_{0 \le x_2 \le 5} \{ f_2(x_2) + f_1(5 - x_2) \}$$
$$= \{ f_2(0) + f_1(5); f_2(1) + f_1(4); f_2(2) + f_1(3); f_2(3) + f_1(2); f_2(4) + f_1(1); f_2(5) + f_1(0) \}$$
$$= \{ 0 + 15 ; 0 + 14 ; 5 + 13 ; 10 + 12 ; 15 + 11 ; 20 + 0 \} = 26 \rightarrow x_2 = 4$$

$$F_3(1) = \max_{0 \le x_3 \le 1} \{ f_3(x_3) + f_2(1 - x_3) \}$$
$$= \{ f_3(0) + f_2(1);\ f_3(1) + f_2(0) \}$$
$$= \{ 0 + 11;\ 2 + 0 \} = 11 \rightarrow x_3 = 0$$

$$F_3(2) = \max_{0 \le x_3 \le 2} \{ f_3(x_3) + f_2(2 - x_3) \}$$
$$= \{ f_3(0) + f_2(2);\ f_3(1) + f_2(1);\ f_3(2) + f_2(0) \}$$
$$= \{ 0 + 12;\ 2 + 11;\ 10 + 0 \} = 13 \rightarrow x_3 = 1$$

$$F_3(3) = \max_{0 \le x_3 \le 3} \{ f_3(x_3) + f_2(3 - x_3) \}$$
$$= \{ f_3(0) + f_2(3);\ f_3(1) + f_2(2);\ f_3(2) + f_2(1);\ f_3(3) + f_2(0) \}$$
$$= \{ 0 + 16\ ;\ 2 + 12\ ;\ 10 + 11\ ;\ 30 + 0 \} = 30 \rightarrow x_3 = 3$$

$$F_3(4) = \max_{0 \le x_3 \le 4} \{ f_3(x_3) + f_2(4 - x_3) \}$$
$$= \{ f_3(0) + f_2(4);\ f_3(1) + f_2(3);\ f_3(2) + f_2(2);\ f_3(3) + f_2(1);\ f_3(4) + f_2(0) \}$$
$$= \{ 0 + 21;\ 2 + 16\ ;\ 10 + 12\ ;\ 30 + 11;\ 32 + 0 \} = 41 \rightarrow x_3 = 3$$

$$F_3(5) = \max_{0 \le x_3 \le 5} \{ f_3(x_3) + f_2(5 - x_3) \}$$
$$= \{ f_3(0) + f_2(5);\ f_3(1) + f_2(4);\ f_3(2) + f_2(3);\ f_3(3) + f_2(2);\ f_3(4) + f_2(1);\ f_3(5) + f_2(0) \}$$
$$= \{ 0 + 26\ ;\ 2 + 21\ ;\ 10 + 16;\ 30 + 12\ ;\ 32 + 11\ ;\ 40 + 0 \} = 43 \rightarrow x_3 = 4$$

$$F_4(1) = \max_{0 \le x_4 \le 1} \{ f_4(x_4) + f_3(1 - x_4) \}$$
$$= \{ f_4(0) + f_3(1);\ f_4(1) + f_3(0) \}$$
$$= \{ 0 + 11;\ 20 + 0 \} = 20 \rightarrow x_4 = 1$$

$$F_4(2) = \max_{0 \le x_4 \le 2} \{ f_4(x_4) + f_3(2 - x_4) \}$$
$$= \{ f_4(0) + f_3(2);\ f_4(1) + f_3(1);\ f_4(2) + f_3(0) \}$$
$$= \{ 0 + 13;\ 20 + 11;\ 21 + 0 \} = 31 \rightarrow x_4 = 1$$

$$F_4(3) = \max_{0 \le x_4 \le 3} \{ f_4(x_4) + f_3(3 - x_4) \}$$
$$= \{ f_4(0) + f_3(3);\ f_4(1) + f_3(2);\ f_4(2) + f_3(1);\ f_4(3) + f_3(0) \}$$
$$= \{ 0 + 30\ ;\ 20 + 13\ ;\ 21 + 11\ ;\ 22 + 0 \} = 33 \rightarrow x_4 = 1$$

$$F_4(4) = \max_{0 \le x_4 \le 4} \{ f_4(x_4) + f_3(4 - x_4) \}$$
$$= \{ f_4(0) + f_3(4);\ f_4(1) + f_3(3);\ f_4(2) + f_3(2);\ f_4(3) + f_3(1);\ f_4(4) + f_3(0) \}$$
$$= \{ 0 + 41;\ 20 + 30\ ;\ 21 + 13\ ;\ 22 + 11;\ 23 + 0 \} = 50 \rightarrow x_4 = 1$$

$$F_4(5) = \max_{0 \le x_4 \le 5} \{ f_4(x_4) + f_3(5 - x_4) \}$$
$$= \{ f_4(0) + f_3(5);\ f_4(1) + f_3(4);\ f_4(2) + f_3(3);\ f_4(3) + f_3(2);\ f_4(4) + f_3(1);\ f_4(5) + f_3(0) \}$$
$$= \{ 0 + 43\ ;\ 20 + 41\ ;\ 21 + 30;\ 22 + 13\ ;\ 23 + 11;\ 24 + 0 \} = 43 \rightarrow x_4 = 1$$

Sehingga di dapat hasil optimal $F_4(5) = 61$ dengan cara meletakkan di perusahaan ke-4 sebesar \$1 atau $x_4 = 1$ &

$F_3(5 - x_4) = F_3(5-1) = F_3(4) = 41 \rightarrow x_3 = 3$
$F_2(5 - x_4 - x_3) = F_2(5-1-3) = F_2(1) = 11 \rightarrow x_2 = 0$
$F_1(5 - x_4 - x_3 - x_2) = F_1(5-1-3-0) = F_1(1) = 11 \rightarrow x_1 = 1$

3. KNAPSACK PROBLEM

Knapsack merupakan bagian dari Dinamik Programming. Knapsack sendiri ada dua yaitu:

1. Knapsack satu dimensi
2. Knapsack dua dimensi

Knapsack satu Dimensi :

Mempunyai satu parameter atau constrain, artinya yang menjadi focus penyelesaian adalah satu objek.

Contoh :

Seorang pendaki gunung ingin membawa bekal atau barang yang dibawa kapasitasnya harus sesuai dengan kapasitas tas.. Jadi yang menjadi focus masalah diatas adalah berat atau bobot bekal yang akan dibawa.

Penyelesaian :

Misal

- b = kemampuan angkat pendaki gunung (kapasitas), maka pendaki gunung harus memperkirakan benda 1,2,…n yang dibawa.
- n = jenis barang yang akan dibawa, maka masing-masing jenis barang mempunyai berat atau bobot dan nilai atau value
- w_i = berat / bobot barang / item ke i
- v_i = nilai / value barang / item ke i
- x_i = banyaknya barang / item ke I yang akan dibawa

Problem diatas dapat di formulasikan sebagai berikut :

$$\begin{aligned} \max \quad & v_1x_1 + v_2x_2 + ...+ v_nx_n \\ \text{constrain} \quad & w_1x_1 + w_2x_2 + ...+ w_nx_n \le b \\ & x_i = 0,1,2,3,..... \end{aligned}$$

Catatan :

1. Linier Program

$$\max \sum_{i=1}^{n} v_i x_i \qquad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \le b$$

$$1 \le i \le m \;\; ; \;\; x_i \;\; \text{riil}$$

2. Linier Program x_i integer disebut Integer programming. Integer Programming untuk constrain tunggal disebut Knapsack programming.

Disini akan diselesaikan dengan Dinamik Programmming

1. Struktur Optimal : "Ingin membawa barang / item sebanyak-banyaknya dengan nilia semaksimal mungkin dan dengan total bobot lebih kecil dari kapasitas pendaki."
2. Formulasi Struktur Optimal

   Misal didefinisikan

   $F_k(y)$ = nilai optimal yang didapat apabila membawa k item yang pertama dan kapasitas membawanya adalah y

   1,2,3,…k,k+1,…….n

$$F_k(y) = \max \sum_{i=1}^{k} v_i x_i \quad (0 \le k \le n)$$

$$\text{dengan batasan} \sum_{i=1}^{k} w_i x_i \quad (0 \le y \le b)$$

$$\text{untuk} \quad k = 0 \rightarrow F_0(y) = 0 \quad (0 \le y \le b)$$

$$y = 0 \rightarrow F_k(0) = 0 \quad (0 \le k \le n)$$

$$\text{untuk} \quad k = 1 \rightarrow F_1(y) = v_1 \left[\frac{y}{w_1} \right] \rightarrow \text{ingin membawa item 1 sebanyak mungkin}$$

$$\text{untuk} \quad k = 2,3,...n \rightarrow F_k(y) = \max\{F_{k-1}(y), F_k(y - w_k) + v_k\}$$

Dari rumus diatas kelihatan bahwa untuk penghitung $F_k(y)$ ada k x b buah nilai.

Contoh Soal :

Kapasitas pendaki gunung b = 10 dan ada 4 macam barng dengan nilai dan bobot sebagai berikut :

Item	v_i	w_i
1	1	2
2	3	3
3	5	4
4	9	7

Ditanya : $F_4(10)$

Jawab :

Untuk k = 1

$$F_1(1) = v_1\left\lfloor \frac{y}{w_1} \right\rfloor = 1\left\lfloor \frac{1}{2} \right\rfloor = 0$$

$$F_1(2) = 1\left\lfloor \frac{2}{2} \right\rfloor = 3$$

$$F_1(3) = 1\left\lfloor \frac{3}{2} \right\rfloor = 1$$

$$F_1(4) = 1\left\lfloor \frac{4}{2} \right\rfloor = 3$$

$$F_1(5) = 1\left\lfloor \frac{5}{2} \right\rfloor = 1$$

$$F_1(6) = 1\left\lfloor \frac{6}{2} \right\rfloor = 4$$

$$F_1(7) = 1\left\lfloor \frac{7}{2} \right\rfloor = 2$$

$$F_1(8) = 1\left\lfloor \frac{8}{2} \right\rfloor = 4$$

$$F_1(9) = 1\left\lfloor \frac{9}{2} \right\rfloor = 2$$

$$F_1(10) = 1\left\lfloor \frac{10}{2} \right\rfloor = 5$$

Untuk k = 2

$$F_2(1) = \max\{F_1(1), F_2(1-3)+v_2\} = \max\{0, -\infty+3\} = 0$$
$$F_2(2) = \max\{F_1(2), F_2(2-3)+v_2\} = \max\{1, -\infty+3\} = 1$$
$$F_2(3) = \max\{F_1(3), F_2(3-3)+v_2\} = \max\{1, 0+3\} = 3$$
$$F_2(4) = \max\{F_1(4), F_2(4-3)+v_2\} = \max\{2, 0+3\} = 3$$
$$F_2(5) = \max\{F_1(5), F_2(5-3)+v_2\} = \max\{2, 1+3\} = 4$$
$$F_2(6) = \max\{F_1(6), F_2(6-3)+v_2\} = \max\{3, 3+3\} = 6$$
$$F_2(7) = \max\{F_1(7), F_2(7-3)+v_2\} = \max\{3, 3+3\} = 6$$
$$F_2(8) = \max\{F_1(8), F_2(8-3)+v_2\} = \max\{4, 4+3\} = 7$$
$$F_2(9) = \max\{F_1(9), F_2(9-3)+v_2\} = \max\{4, 6+3\} = 9$$
$$F_2(10) = \max\{F_1(10), F_2(10-3)+v_2\} = \max\{5, 6+3\} = 9$$

Untuk k = 3

$$F_3(1) = \max\{F_2(1), F_3(1-4)+v_3\} = \max\{0, -\infty+5\} = 0$$
$$F_3(2) = \max\{F_2(2), F_3(2-4)+v_3\} = \max\{1, -\infty+5\} = 1$$
$$F_3(3) = \max\{F_2(3), F_3(3-4)+v_3\} = \max\{3, -\infty+5\} = 3$$
$$F_3(4) = \max\{F_2(4), F_3(4-4)+v_3\} = \max\{3, 0+5\} = 5$$
$$F_3(5) = \max\{F_2(5), F_3(5-4)+v_3\} = \max\{4, 0+5\} = 5$$
$$F_3(6) = \max\{F_2(6), F_3(6-4)+v_3\} = \max\{6, 1+5\} = 6$$
$$F_3(7) = \max\{F_2(7), F_3(7-4)+v_3\} = \max\{6, 3+5\} = 8$$
$$F_3(8) = \max\{F_2(8), F_3(8-4)+v_3\} = \max\{7, 5+5\} = 10$$
$$F_3(9) = \max\{F_2(9), F_3(9-4)+v_3\} = \max\{9, 5+5\} = 10$$
$$F_3(10) = \max\{F_2(10), F_3(10-4)+v_3\} = \max\{9, 6+5\} = 11$$

Untuk k = 4

$$F_4(1) = \max\{F_3(1), F_4(1-7)+v_4\} = \max\{0, -\infty+9\} = 0$$
$$F_4(2) = \max\{F_3(2), F_4(2-7)+v_4\} = \max\{1, -\infty+9\} = 1$$
$$F_4(3) = \max\{F_3(3), F_4(3-7)+v_4\} = \max\{3, -\infty+9\} = 3$$
$$F_4(4) = \max\{F_3(4), F_4(4-7)+v_4\} = \max\{5, -\infty+9\} = 5$$
$$F_4(5) = \max\{F_3(5), F_4(5-7)+v_4\} = \max\{5, -\infty+9\} = 5$$
$$F_4(6) = \max\{F_3(6), F_4(6-7)+v_4\} = \max\{6, -\infty+9\} = 6$$
$$F_4(7) = \max\{F_3(7), F_4(7-7)+v_4\} = \max\{8, 0+9\} = 9$$
$$F_4(8) = \max\{F_3(8), F_4(8-7)+v_4\} = \max\{10, 0+9\} = 10$$
$$F_4(9) = \max\{F_3(9), F_4(9-7)+v_4\} = \max\{10, 1+9\} = 10$$
$$F_4(10) = \max\{F_3(10), F_4(10-7)+v_4\} = \max\{11, 3+9\} = 12$$

Setelah perhitungan diatas dapat dimasukkan pada table $F_k(y)$ sebagai berikut :

F y k	1	2	3	4	5	6	7	8	9	10
1	0	1	1	2	2	3	3	4	4	5
2	0	1	3	3	4	6	6	7	9	9
3	0	1	3	5	5	6	8	10	10	11
4	0	1	3	5	5	6	9	10	10	12

Dari table diatas dapat diketahui $F_4(10)$ = 12 yang artinya nilai maksimal yang didapat apabila membawa 4 item yang pertama dengan kapasitas membawa 10. Untuk menghitung berapa banyak (x_i) dari masing-masing item yang harus dibawa adalah dengan menghitung index terlebih dahulu yaitu didefinisikan :

$i(k, y)$ = index max dalam penghitungan $F_k(y)$

Misal $i(k, y) = j(1 \le j \le k)$

Item ke j dibawa paling tidak 1

$i(0, y) = 0$; $i(k,0) = 0$

$$i(k, y) = \begin{cases} k & \text{jika } F_{k-1}(y) < F_k(y - w_k) + v_k \\ i(k-1, y) & \text{jika } F_{k-1}(y) \ge F_k(y - w_k) + v_k \end{cases}$$

Untuk k = 1

$$i(1,1) = \begin{cases} 1 & \text{jika } F_0(1) < F_1(1-2) + 1 \\ i(0,1) & \text{jika } F_0(1) \ge F_1(1-2) + 1 \end{cases} = \begin{cases} 1 & \text{jika } 0 < -\infty + 1 \\ i(0,1) & \text{jika } 0 \ge -\infty + 1 \end{cases}$$

$$i(1,1) = i(0,1) = 0$$

$$i(1,2) = \begin{cases} 1 & \text{jika } F_0(2) < F_1(2-2) + 1 \\ i(0,2) & \text{jika } F_0(2) \ge F_1(2-2) + 1 \end{cases} = \begin{cases} 1 & \text{jika } 0 < 0 + 1 \\ i(0,2) & \text{jika } 0 \ge 0 + 1 \end{cases}$$

$$i(1,2) = 1$$

$$i(1,3) = \begin{cases} 1 & \text{jika } F_0(3) < F_1(3-2) + 1 \\ i(0,3) & \text{jika } F_0(3) \ge F_1(3-2) + 1 \end{cases} = \begin{cases} 1 & \text{jika } 0 < 0 + 1 \\ i(0,3) & \text{jika } 0 \ge 0 + 1 \end{cases}$$

$$i(1,3) = 1$$

$$\vdots$$

$$\vdots$$

$$i(1,10) = \begin{cases} 1 & \text{jika } F_0(10) < F_1(10-2) + 1 \\ i(0,10) & \text{jika } F_0(10) \ge F_1(10-2) + 1 \end{cases} = \begin{cases} 1 & \text{jika } 0 < 5 + 1 \\ i(0,10) & \text{jika } 0 \ge 5 + 1 \end{cases}$$

$$i(1,10) = 1$$

Untuk k = 2

$$i(2,1)=\begin{cases}2 & \text{jika } F_1(1)<F_1(1-3)+3\\ i(1,1) & \text{jika } F_1(1)\geq F_1(1-3)+3\end{cases}=\begin{cases}2 & \text{jika } 0<-\infty+3\\ i(1,1) & \text{jika } 0\geq-\infty+3\end{cases}$$

$$i(2,1)=i(1,1)=0$$

$$i(2,2)=\begin{cases}2 & \text{jika } F_1(2)<F_1(2-3)+3\\ i(1,2) & \text{jika } F_1(2)\geq F_1(2-3)+3\end{cases}=\begin{cases}2 & \text{jika } 1<-\infty+3\\ i(1,2) & \text{jika } 1\geq-\infty+3\end{cases}$$

$$i(2,2)=i(1,2)=1$$

$$i(2,3)=\begin{cases}2 & \text{jika } F_1(3)<F_1(3-3)+3\\ i(1,3) & \text{jika } F_1(3)\geq F_1(3-3)+3\end{cases}=\begin{cases}2 & \text{jika } 1<0+3\\ i(1,3) & \text{jika } 1\geq0+3\end{cases}$$

$$i(2,3)=2$$

$$\vdots$$

$$\vdots$$

$$i(2,10)=\begin{cases}2 & \text{jika } F_1(10)<F_1(10-3)+3\\ i(1,10) & \text{jika } F_1(10)\geq F_1(10-3)+3\end{cases}=\begin{cases}2 & \text{jika } 5<5+3\\ i(1,10) & \text{jika } 5\geq5+3\end{cases}$$

$$i(2,10)=2$$

Untuk k = 3

$$i(3,1)=\begin{cases}3 & \text{jika } F_2(1)<F_3(1-4)+5\\ i(2,1) & \text{jika } F_2(1)\geq F_3(1-4)+5\end{cases}=\begin{cases}3 & \text{jika } 0<-\infty+5\\ i(2,1) & \text{jika } 0\geq-\infty+5\end{cases}$$

$$i(3,1)=i(2,1)=0$$

$$i(3,2)=\begin{cases}3 & \text{jika } F_2(2)<F_3(2-4)+5\\ i(2,2) & \text{jika } F_2(2)\geq F_3(2-4)+5\end{cases}=\begin{cases}3 & \text{jika } 1<-\infty+5\\ i(2,2) & \text{jika } 1\geq-\infty+5\end{cases}$$

$$i(3,2)=i(2,2)=1$$

$$i(3,3)=\begin{cases}3 & \text{jika } F_2(3)<F_3(3-4)+5\\ i(2,3) & \text{jika } F_2(3)\geq F_3(3-4)+5\end{cases}=\begin{cases}3 & \text{jika } 1<-\infty+5\\ i(2,3) & \text{jika } 1\geq-\infty+5\end{cases}$$

$$i(3,3)=2$$

$$\vdots$$

$$\vdots$$

$$i(3,10)=\begin{cases}3 & \text{jika } F_2(10)<F_3(10-4)+5\\ i(2,10) & \text{jika } F_2(10)\geq F_3(10-4)+5\end{cases}=\begin{cases}3 & \text{jika } 9<5+5\\ i(2,10) & \text{jika } 9\geq5+5\end{cases}$$

$$i(3,10)=3$$

Untuk k = 4

$$i(4,1)=\begin{cases}4 & \text{jika } F_3(1)<F_4(1-7)+9\\ i(3,1) & \text{jika } F_3(1)\geq F_4(1-7)+9\end{cases}=\begin{cases}4 & \text{jika } 0<-\infty+9\\ i(3,1) & \text{jika } 0\geq-\infty+9\end{cases}$$

$$i(4,1)=i(3,1)=0$$

$$i(4,2)=\begin{cases}4 & \text{jika } F_3(2)<F_4(2-7)+9\\ i(3,2) & \text{jika } F_3(2)\geq F_4(2-7)+9\end{cases}=\begin{cases}4 & \text{jika } 1<-\infty+9\\ i(3,2) & \text{jika } 1\geq-\infty+9\end{cases}$$

$$i(4,2)=i(3,2)=1$$

$$i(4,3)=\begin{cases}4 & \text{jika } F_3(3)<F_4(3-7)+9\\ i(3,3) & \text{jika } F_3(3)\geq F_4(3-7)+9\end{cases}=\begin{cases}4 & \text{jika } 1<-\infty+9\\ i(3,3) & \text{jika } 1\geq-\infty+9\end{cases}$$

$$i(4,3)=i(3,3)=2$$

$$\vdots$$

$$\vdots$$

$$i(4,10)=\begin{cases}4 & \text{jika } F_3(10)<F_4(10-7)+9\\ i(3,10) & \text{jika } F_3(10)\geq F_4(10-7)+9\end{cases}=\begin{cases}4 & \text{jika } 11<3+9\\ i(3,10) & \text{jika } 11\geq 3+9\end{cases}$$

$$i(4,10)=4$$

Dari perhitungan diatas dapat dibuat table indexnya sebagai berikut:

i y k	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	1	1	1	1
2	0	1	2	2	2	2	2	2	2	2
3	0	1	2	3	3	3	3	3	3	3
4	0	1	2	3	3	3	4	3	4	4

Pada $i(4,10)=4$ mempunyai arti item / barang ke 4 terbawa minimal 1.

Selanjutnya dilakuakan perhitungan secara Buttom-up

$i(4,10)=4$

$i(4,10-w_4)=i(4,10-7)=i(4,3)=2 \rightarrow$ item / barang ke 2 terbawa minimal 1 Jadi

$i(4,3-w_2)=i(4,3-3)=i(4,0)=0 \rightarrow$ sudah tidak ada item / barang yang dapat dibawa

barang yang di bawa $x_2=1$ & $x_4=1$ dengan nilai maksimal adalah 12

4. PERMASALAHAN KNAPSACK DUA – DIMENSI

Materi yang diambil untuk knapsack adalah batasan berat / beban. Dengan kata lain, materi terbatas oleh satu parameter berat / beban. Versi ini dari suatu masalah knapsack dapat dikenal sebagai masalah knapsack dimensional. Di dalam bagian ini, kita mempertimbangkan suatu versi masalah knapsack dua – dimensi di mana masing-masing item terbatas oleh dua parameter yaitu panjang dan lebar.

Untuk merumuskan versi ini menyangkut masalah knapsack, kita mempertimbangkan permasalahan dalam memotong suatu papan besar menjadi segiempat kecil dengan ukuran panjang yang diberikan.

Asumsikan bahwa kita diberi sebuah papan yang besar dan kemudian kita akan memotong papan yang besar menjadi segiempat kecil kemudian dijual di pasar. Jika kita mengetahui harga dari bermacam-macam segiempat kecil di pasar, bagaimana cara kita memotong papan yang besar menjadi sedemikian rupa sehingga kita mendapatkan laba yang maksimum ? Versi satu – dimensi terbatas dari masalah ini (juga disebut masalah

stock-cutting) akan dihubungkan dengan suatu masalah knapsack. Di sini, masing – masing item ditandai oleh dua parameter, panjang dan lebar, dan knapsack adalah setara dengan papan yang besar. Di dalam ilmu pengetahuan komputer, masing – masing item merupakan suatu pekerjaan yang mana memerlukan beberapa waktu tertentu dan suatu jumlah tertentu dari ruang memori, dan kita bermaksud menyelesaikan semua pekerjaan di dalam kapasitas memori dari komputer dalam periode waktu 24 jam.

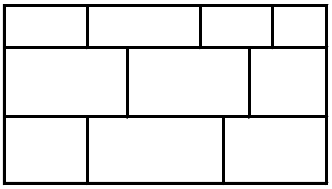
Jika papan segiempat kecil mewakili suatu pekerjaan dengan waktu dan ruang yang diberikan, kemudian segiempat tidak bisa diputar secara pas pada pembatasan ukuran dari papan besar. Jika kita memotong suatu kepingan besar dari gelas / kaca, orientasi dari segiempat kecil berkenaan dengan kepingan yang besar tidak relevan. Jika perputaran dari segiempat memenuhi, dan setara dengan mempunyai dua macam segiempat dengan harga yang sama (masing – masing dengan orientasi yang ditentukan). Mulai sekarang, kita akan berasumsi bahwa perputaran dari segiempat dengan panjang yang kecil tidak memenuhi. Masalah stock-cutting memenuhi papan yang besar menjadi potongan dengan cara berubah-ubah.

Versi stock-cutting ini masih terlalu rumit, dan kita akan menggunakan suatu cara yang terbatas dalam memotong. Ini masalah terbatas untuk memenuhi papan yang besar agar dipotong sebagai berikut:

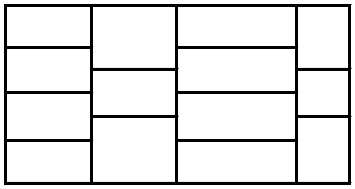
(1) dengan garis mendatar semua cara berlaku untuk digunakan oleh potongan tegak pada masing – masing potongan mendatar seperti yang ditunjukkan pada gambar (a).

(2) dengan garis tegak semua cara berlaku untuk digunakan oleh potongan mendatar pada masing – masing potongan tegak seperti yang ditunjukkan pada gambar (b).

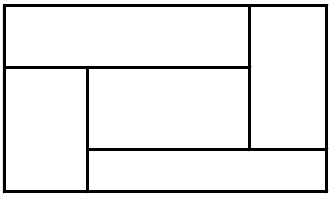
Pola pada gambar (a) dan (b) disebut potongan tingkat – dua karena pola ini dapat dicapai pertama oleh potongan sepanjang satu sumbu koordinat dan kemudian dipotong sepanjang koordinat yang lain sehingga menghasilkan beberapa potongan. Potongan pola ditunjukkan pada gambar (c) dan tidak memenuhi karena tidak bisa dicapai secara berulang potongan segiempat besar menjadi dua segiempat. Gambar (d) tidak memenuhi karena memerlukan lebih dari dua langkah potongan.



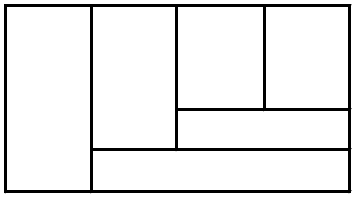
(a)



(b)



(c)



(d)

Andaikan bahwa kita diberi nilai – nilai v_i dari n segiempat, dimana l_i adalah panjang (yang mendatar) dari segiempat dan w_i adalah lebar (yang tegak) dari segiempat. Kita akan memotong papan besar menjadi beberapa segiempat sehingga menghasilkan nilai maksimum. Jika suatu segiempat menghasilkan tidak persis l_i x w_i untuk semua i, kita asumsikan bahwa segiempat mempunyai nilai yang sama dengan nilai maksimum dari semua segiempat sehingga dapat pas di dalamnya. Dengan kata lain, kita tidak perlu memotong segiempat menjadi ukuran yang tepat l_i x w_i.

Sebagai contoh, ambil papan besar dengan ukuran L = 14, W = 11 dan menjadi segiempat kecil dengan ukuran :

$v_1 = \$ 6$	$l_1 = 7$	$w_1 = 2$
$v_2 = \$ 7$	$l_2 = 5$	$w_2 = 3$
$v_3 = \$ 9$	$l_3 = 4$	$w_3 = 5$

Satu cara untuk potongan dengan garis mendatar dan kemudian garis tegak ditunjukkan dalam gambar (e) dengan total laba $ 63. Ketika cara untuk potongan dengan garis tegak dan kemudian garis mendatar ditunjukkan pada gambar (f) dengan total laba $ 64.

Catatan bahwa potongan pada gambar (f) tidak bisa dicapai dalam dua langkah, untuk melengkapi langkah ketiga sangat diperlukan. Bagaimanapun juga, kita tidak

melakukan kelengkapan langkah dan sederhananya membuat 5 x 5 dan 4 x 6 segiempat sebagaimana adanya dengan nilai $ 9.

Untuk mendapatkan jumlah optimal dari potongan mendatar dan kemudian tegak, kita mempertimbangkan permasalahan dari potongan papan dengan panjang 14 dan mengabaikan pembatasan lebar. Ini adalah masalah knapsack satu – dimensi, yakni :

maks $6 x_1 + 7 x_2 + 9 x_3$

dengan konstrain $7 x_1 + 5 x_2 + 4 x_3 \leq 14$ (1)

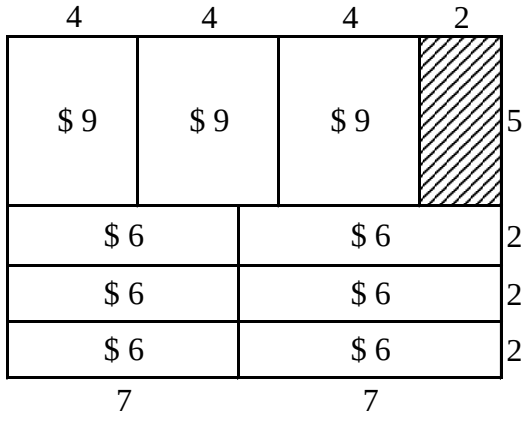
dimana x_i menandakan banyaknya item ke – i dari segiempat kecil yang dihasilkan. Jika papan besar mempunyai lebar 2, kemudian hanya segiempat pertama yang dapat dihasilkan. Jika papan besar mempunyai lebar 3, hanya segiempat pertama dan kedua yang dapat dihasilkan.

Di sini kita definisikan :

$$F_k (x) = \text{maks} \sum^{k} v_i x_i$$

dengan konstrain $\sum_{i=1}^{k} l_i x_i \leq x$ (2)

$x_i \geq 0$ integer.



(e)

(f)

Masalah (2) dapat dipecahkan dengan algoritma knapsack. Hasil dari komputasi ini ditunjukkan pada tabel (g).

Nilai dari $F_k (x)$:

- Nilai awal $F_k (x)$

  $k = 0$ maka $F_0 (x) = 0$

  $x = 0$ maka $F_k (x) = 0$

- Untuk $k = 1$ dan $1 \leq x \leq 14$

  $$F_1 (1) = v_1 \left\lfloor \frac{x}{l_k} \right\rfloor = 6 \left\lfloor \frac{1}{7} \right\rfloor = 0$$

$F_1(6) = v_1 \left\lfloor \frac{x}{l_k} \right\rfloor = 6 \left\lfloor \frac{6}{7} \right\rfloor = 0$

$F_1(7) = v_1 \left\lfloor \frac{x}{l_k} \right\rfloor = 6 \left\lfloor \frac{7}{7} \right\rfloor = 6$

$F_1(13) = v_1 \left\lfloor \frac{x}{l_k} \right\rfloor = 6 \left\lfloor \frac{13}{7} \right\rfloor = 6$

$F_1(14) = v_1 \left\lfloor \frac{x}{l_k} \right\rfloor = 6 \left\lfloor \frac{14}{7} \right\rfloor = 12$

- Untuk $k = 2$ dan $1 \le x \le 14$

  $F_k(x) = \max \{ F_{k-1}(x) ; F_k(x - l_k) + v_k \}$

  $F_2(1) = \max \{ F_{2-1}(1) ; F_2(1 - l_2) + v_2 \}$

  $= \max \{ 0 ; -\infty + 7 \} = 0$

  $F_2(14) = \max \{ F_{2-1}(14) ; F_2(14 - l_2) + v_2 \}$

  $= \max \{ 12 ; 7 + 7 \} = 14$

- Untuk $k = 3$ dan $1 \le x \le 14$

  $F_k(x) = \max \{ F_{k-1}(x) ; F_k(x - l_k) + v_k \}$

  $F_3(1) = \max \{ F_{3-1}(1) ; F_3(1 - l_3) + v_3 \}$

  $= \max \{ 0 ; -\infty + 9 \} = 0$

  $F_3(14) = \max \{ F_{3-1}(14) ; F_3(14 - l_3) + v_3 \}$

  $= \max \{ 14 ; 18 + 9 \} = 27$

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$F_1(x)$	0	0	0	0	0	0	6	6	6	6	6	6	6	12
$F_2(x)$	0	0	0	0	7	7	7	7	7	14	14	14	14	14
$F_3(x)$	0	0	0	9	9	9	9	18	18	18	18	27	27	27

(g)

Nilai dari $G_k(y)$:

- Nilai awal $G_k (y)$

  k = 0 maka $G_0 (y) = 0$

  y = 0 maka $G_k (y) = 0$

  $(\pi_1, \pi_2, \pi_3) = (12, 14, 27)$

- Untuk k = 1 dan $1 \le y \le 11$

  $G_1 (1) = \pi_1 \left\lfloor \frac{y}{w_k} \right\rfloor = 12 \left\lfloor \frac{1}{2} \right\rfloor = 0$

  $G_1 (11) = \pi_1 \left\lfloor \frac{y}{w_k} \right\rfloor = 12 \left\lfloor \frac{11}{2} \right\rfloor = 60$

- Untuk k = 2 dan $1 \le y \le 11$

  $G_k (y) = \max \{ G_{k-1} (y) ; G_k (y - w_k) + \pi_k \}$

  $G_2 (1) = \max \{ G_{2-1} (1) ; G_2 (1 - w_2) + \pi_2 \}$

  $= \max \{ 0 ; - \infty + 14 \} = 0$

  $G_2 (11) = \max \{ G_{2-1} (11) ; G_2 (11 - w_2) + \pi_2 \}$

  $= \max \{ 60 ; 48 + 14 \} = 62$

- Untuk k = 3 dan $1 \le y \le 11$

  $G_k (y) = \max \{ G_{k-1} (y) ; G_k (y - w_k) + \pi_k \}$

  $G_3 (1) = \max \{ G_{3-1} (1) ; G_3 (1 - w_3) + \pi_3 \}$

  $= \max \{ 0 ; - \infty + 27 \} = 0$

  $G_3 (11) = \max \{ G_{3-1} (11) ; G_3 (11 - w_3) + \pi_3 \}$

  $= \max \{ 62 ; 36 + 27 \} = 63$

Y	1	2	3	4	5	6	7	8	9	10	11
$G_1(y)$	0	12	12	24	24	36	36	48	48	60	60
$G_2(y)$	0	12	14	24	26	36	38	48	50	60	62
$G_3(y)$	0	12	14	24	27	36	39	48	51	60	63

(h)

Pertanyaan sekarang menjadi " Berapa banyak potongan yang mempunyai harga $ 12, $ 14, $ 27 yang harus dihasilkan ?". Dengan satu – dimensi masalah knapsack, yakni :

maks $12\ y_1 + 14\ y_2 + 27\ y_3$

dengan konstrain $2\ y_1 + 3\ y_2 + 5\ y_3 \leq 11$ (3)

$y_i \geq 0$ integer.

Di sini y_i menandakan banyaknya potongan yang dihasilkan. (Pada riset operasi, harga $ 12, $ 14 dan $ 27 dapat ditafsirkan dari harga perkiraan untuk lebar).

Kita dapat menyelesaikan masalah dengan :

mendefinisikan $G_k (y) = \sum_{i=1}^{k} \pi_1\ y_i$

dengan konstrain $\sum_{i=1}^{k} w_i\ y_i \leq y$ (4)

$y_i \geq 0$ integer

dimana (π_1, π_2, π_3) = (12, 14, 27).

Hasil komputasi pada (4) ditunjukkan pada tabel (h) dan hasil potongan mencapai $ 63 ditunjukkan pada gambar (e).

Jika papan besar yang asli mempunyai L = 13, W = 11, kemudian kita gunakan (π_1, π_2, π_3) pada (6, 14, 27) dalam (4).

Secara umum, kita akan membuat : $w_1 < w_2 < \ldots < w_n$. Kemudian $F_k (x)$ adalah nilai maksimum yang dapat diperoleh dalam potongan dengan lebar kurang dari atau sama dengan w_k. Kemudian kita selesaikan (4) dengan $\pi_i = F_i(L)$, di mana L adalah panjang papan besar yang akan dipotong.

Atau dilakukan indexing berdasarkan tabel $G_k(y)$ kemudian indexing ke tabel $F_k(x)$.

- Indexing dari tabel $G_k(y)$

  Didefinisikan : i (k,y) = index maksimum dari pola ke-k

$$i(k,y) = \begin{cases} K, & \text{jika } G_{k-1}(y) < G_k(y-l_i) + \pi_k \\ i(k-1,y), & \text{jika } G_{k-1}(y) \geq G_k(y-l_i) + \pi_k \end{cases}$$

  Hasilnya :

Y	1	2	3	4	5	6	7	8	9	10	11
$I_1(y)$	0	1	1	1	1	1	1	1	1	1	1
$I_2(y)$	0	1	2	1	2	1	2	1	2	1	2
$I_3(y)$	0	1	2	1	3	1	3	1	3	1	3

Nilai maksimal $ 63 terletak pada (k=3, y=11) pada tabel $G_k(y)$. Pada tabel index baris dan kolom yang bersesuaian (k=3, y=11) bernilai (index) 3. Artinya kita memotong secara horisontal dengan lebar 5 m.

Pencarian index :

i(3,11-5) = i(3,6) = 1 (dipotong horisontal dengan lebar 2m)

i(3, 6-2) = i(3,4) = 1 (dipotong horisontal dengan lebar 2m)

i(3, 4-2) = i(3,2) = 1 (dipotong horisontal dengan lebar 2m)

i(3, 2-2) = i(3,0) = 0

Papan besar dengan panjang 11m dipotong 5m dan dipotong 2m sebanyak 3 kali.

- Indexing dari tabel $F_k(x)$

  Didefinisikan : i (k,y) = index maksimum dari pola ke-k

$$i(k,y) = \begin{cases} K, & \text{jika } F_{k-1}(x) < F_k(x-w_i) + v_k \\ i(k,x), & \text{jika } F_{k-1}(x) \geq G_k(x-w_i) + v_k \end{cases}$$

Hasilnya :

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$I_1(x)$	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$I_2(x)$	1	1	1	1	2	2	2	2	2	2	2	2	2	2
$I_3(x)$	1	1	3	3	3	3	3	3	3	3	3	3	3	3

Karena dari index tabel sebelumnya yang muncul pertama kali pemotongan horisontal dengan panjang 5m, maka untuk index tabel $F_k(x)$ dimulai dari maksimum k=3 (k=3, y=14).

Selanjutnya pencarian index :

i(3,14-4) = i(3,10) = 3 (pemotongan vertikal dengan panjang 4m)

i(3,10-4) = i(3,6) = 3 (pemotongan vertikal dengan panjang 4m)

i(3, 6-4) = i(3,2) = 2 (pemotongan vertikal dengan panjang 7m)

i(3, 2-7) = i(3,-∞) = 0

Pada akhirnya potongan horizontal pertama dengan lebar 5m akan dipotong vertikal dengan panjang 4m sebanyak tiga kali dan ketiga papan potongan horisontal 2m akan dipotong masing-masing 7m. Hasil potongan ditunjukkan pada gambar (e).

Kita dapat memperoleh jumlah maksimum dari potongan secara tegak dan dan secara mendatar dalam cara yang serupa. Di sini kita pertama memesan lagi segiempat menurut panjangnya mereka.

$v'_1 = \$\ 9$ $l'_1 = 4$ $w'_1 = 5$

$v'_2 = \$\ 7$ $l'_2 = 5$ $w'_2 = 3$

$v'_3 = \$\ 6$ $l'_3 = 7$ $w'_3 = 2$

Masalah satu – dimensi menjadi :

$$\text{maks} \quad 9\,x'_1 + 7\,x'_2 + 6\,x'_3$$

$$\text{dengan konstrain} \quad 5\,x'_1 + 3\,x'_2 + 2\,x'_3 \le 11 \qquad (5)$$

$$x'_i \ge 0 \ \text{integer.}$$

di sini x'_i menandakan banyaknya potongan mendatar yang dihasilkan.

Masalah (5) dapat diselesaikan dengan mendefinisikan :

$$F'_k(y) = \text{maks} \sum_{i=1}^{k} v'_i\,x'_i$$

$$\text{dengan konstrain} \sum_{i=1}^{k} w'_i\,x'_i \le y \le 11 \qquad (6)$$

$x'_i \geq 0$ integer.

Hasil (6) ditunjukkan pada tabel (i). Juga dalam tabel (i), kita lihat bahwa potongan tegak dengan panjang 4 x 11, 5 x 11, 7 x 11 berharga $ 18, $ 23 dan $ 31, berturut-turut.

Kemudian masalah sekarang menjadi :

maks $18\ y'_1 + 23\ y'_2 + 31\ y'_3$

dengan konstrain $4\ y'_1 + 5\ y'_2 + 7\ y'_3 \leq 14$ (7)

$y'_i \geq 0$ integer.

Di sini y'_i menandakan banyaknya potongan tegak yang dihasilkan. Lagi (7) dapat diselesaikan dengan pendefinisian :

$$G'_k(x) = \sum_{i=1}^{k} \pi'_i\, y'_i$$

dengan konstrain $\sum_{i=1}^{k} l'_i\, y'_i \leq x$ (8)

$y'_i \geq 0$ integer

Hasil dari (8) ditunjukkan dalam tabel (j) dan pola mencapai $ 64 ditunjukkan pada gambar (f). Nilai dari $F'_k(y)$ dan $G'_k(x)$:

y	1	2	3	4	5	6	7	8	9	10	11
$F_1'(y)$	0	0	0	0	9	9	9	9	9	18	18
$F_2'(y)$	0	0	7	7	9	14	14	16	21	21	23
$F_3'(y)$	0	6	7	12	13	18	19	24	25	30	31

(i)

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$G_1'(x)$	0	0	0	18	18	18	18	36	36	36	36	54	54	54
$G_2'(x)$	0	0	0	18	23	23	23	36	41	46	46	54	54	64
$G_3'(x)$	0	0	0	18	23	23	31	36	41	46	49	54	54	64

(j)