PERMASALAHAN OPTIMASI *KNAPSACK* 0-1 DENGAN MENGGUNAKAN ALGORITMA *DYNAMIC PROGRAMMING*

Skripsi

Disusun untuk melengkapi syarat-syarat

guna memperoleh gelar Sarjana Sains

ANGGITA DYAH AYU PITALOKA

3125121978

PROGRAM STUDI MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI JAKARTA

2017

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

PERMASALAHAN OPTIMASI *KNAPSACK* 0-1 DENGAN MENGGUNAKAN ALGORITMA *DYNAMIC PROGRAMMING*

Nama : Anggita Dyah Ayu Pitaloka

No. Registrasi : 3125121978

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: Prof. Dr. Suyono, M.Si.	.....	.....
	NIP. 19671218 199303 1 005		
Wakil Penanggung Jawab			
Pembantu Dekan I	: Dr. Muktiningsih, M.Si.	.....	.....
	NIP. 19640511 198903 2 001		
Ketua	: Drs. Mulyono, M.Kom.	.....	.....
	NIP. 19660517 199403 1 003		
Sekretaris	: Dr. Lukita Ambarwati, S.Pd, M.Si.	.....	.....
	NIP. 19721026 200112 2 001		
Penguji	: Ir. Fariani Hermin, M.T.	.....	.....
	NIP. 19600211 198703 2 001		
Pembimbing I	: Ratna Widyati, S.Si, M.Kom.	.....	.....
	NIP. 196750925 200212 2 002		
Pembimbing II	: Med Irzal, M.Kom.	.....	.....
	NIP. 19770615 200312 1 001		

Dinyatakan lulus ujian skripsi tanggal: 24 Januari 2017

ABSTRACT

ANGGITA DYAH AYU PITALOKA, 3125121978. 0-1 Knapsack Optimization Problems Using Dynamic Programming Algorithm. Thesis. Faculty of Mathematics and Natural Science Jakarta State University. 2016.

In this thesis, the author uses deterministic algorithm that is Dynamic Programming Algorithm on 0-1 knapsack problems, it is an issue for the election of items from a collection of items given where each item has a weight and a profit different, so that with available capacity is expected election of such items has the maximum profit. On the 0-1 knapsack, status of items is divided into two parts, namely the items are given a value of 1 if that items are elected and the items are given a value of 0 if that items are not elected. Dynamic programming resolve this problem by breaking the solution into a set of steps or stages, so that there are a series of decisions relating, which in this case study uses two recursive approach, that is forward and backward to obtain optimal results.

Keywords *: Knapsack Problems, Otimization, Dynamic Programming Algorithm, Forward and Backward.*

ABSTRAK

ANGGITA DYAH AYU PITALOKA, 3125121978. Permasalahan Optimasi *Knapsack* 0-1 dengan Menggunakan Algoritma *Dynamic Programming*. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2016.

Pada skripsi ini penulis menggunakan algoritma deterministik yaitu Algoritma *Dynamic Programming* pada permasalahan *knapsack* 0-1, yaitu suatu permasalahan cara pemilihan barang dari sekumpulan barang yang diberikan di mana setiap barang tersebut mempunyai berat dan keuntungan yang berbeda-beda, sehingga dengan kapasitas yang tersedia diharapkan pemilihan barang tersebut mempunyai keuntungan yang maksimal. Pada *knapsack* 0-1, status barang dibagi menjadi dua yaitu barang tersebut diberi nilai 1 jika barang tersebut dipilih dan diberi nilai 0 jika barang tersebut tidak dipilih. *Dynamic programming* menyelesaikan permasalahan ini dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan sehingga terdapat serangkaian keputusan yang berkaitan, di mana pada studi kasus ini menggunakan dua pendekatan rekursif yaitu pendekatan maju (*forward*) dan pendekatan mundur (*backward*) untuk memperoleh hasil yang optimal.

Kata kunci : *Knapsack Problem*, optimasi, Algoritma *Dynamic Programming*, pendekatan maju (*forward*) dan pendekatan mundur (*backward*).

PERSEMBAHANKU...

"*Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari suatu urusan), tetaplah bekerja keras (untuk urusan yg lain), dan hanya kepada Tuhanmulah engkau berharap. (Q.S Al Insyirah: 6-8)*"

"*Jika ragu dalam melakukan sesuatu, sebaiknya tanya kepada diri sendiri, apa yang kita inginkan esok hari dari apa yang telah kita lakukan sebelumnya. - Jonh Lubbock*"

"*Tak seorangpun pernah dihormati karena apa yang dia terima. Kehormatan adalah penghargaan bagi orang yang telah memberikan sesuatu yang berarti. - Calvin Coolidge*"

Skripsi ini kupersembahkan untuk Ayah, Ibu dan Ebot.

"Terima kasih atas dukungan, do'a, serta kasih sayang kalian".

KATA PENGANTAR

Puji syukur kepada Allah SWT atas pengetahuan dan kemampuan sehingga penulis dapat menyelesaikan skripsi yang berjudul "Permasalahan Optimasi *Knapsack* 0-1 dengan Menggunakan Algoritma *Dynamic Programming*" yang merupakan salah satu syarat dalam memperoleh gelar Sarjana Jurusan Matematika Universitas Negeri Jakarta.

Skripsi ini berhasil diselesaikan tidak terlepas dari adanya bantuan dari berbagai pihak. Oleh karena itu, dalam kesempatan ini penulis ingin menyampaikan terima kasih terutama kepada:

1. Ibu Ratna Widyati, S.Si, M.Kom selaku Dosen Pembimbing I dan Bapak Med Irzal, M.Kom selaku Dosen Pembimbing II, yang telah meluangkan waktunya dalam memberikan bimbingan, saran, nasihat, dan arahan sehingga skripsi ini dapat menjadi lebih baik dan terarah.

2. Ibu Ir. Fariani Hermin, MT., selaku Pembimbing Akademik, Ibu Dr. Lukita Ambarwati, S.Pd, M.Si., selaku Ketua Prodi Matematika FMIPA UNJ. Terima kasih atas segala bantuan dan kerja sama Ibu selama perkuliahan dan pengerjaan skripsi ini.

3. Dosen-dosen Matematika yang sudah memberikan ilmunya kepada penulis sehingga penulis dapat membagikan kembali ilmunya kepada orang lain dengan kemampuan yang penulis miliki.

4. Seluruh karyawan/karyawati FMIPA UNJ. Terima kasih atas segala kemudahan dan informasi yang diberikan kepada penulis selama perkuliahan dan pengerjaan skripsi ini.

5. Ayah Teguh Harjono dan Ibu Siti Nurjanah selaku kedua orangtua penulis yang sangat penulis sayang dan menjadi alasan penulis agar skripsi ini harus cepat selesai yang selalu mendoakan, mendukung, mempercayai, memberi motivasi, dan setia membantu penulis dengan penuh cinta dan kasih sayang yang tulus.

6. Adik laki-laki penulis, Rezza Dendi Irawan (Ebot) yang terus memberi doa, semangat, dan selalu menghibur ketika penulis mengalami kesulitan dalam penulisan skripsi ini. Semangat juga sekolahnya agar yang kamu cita-citakan terkabul.

7. Teman-teman MATEMATIKA 2012 yang selama ini menjadi keluarga kedua yang bersedia menjadi penyemangat disaat penulis mengalami kejenuhan dalam menulis skripsi ini terutama Alphien, Bobby, Ela, Irma dan Dewanti yang sudah berjuang bersama dan selalu menemani bimbingan penulis.

8. Teman Kesayangan (TKS) terima kasih untuk Vinna, Acit, Yuli, Fara, Ibeth, Mega, Jennyfer, dan Atun yang selama ini selalu menjadi teman baik penulis dari semester 1 sampai saat ini dan insya Allah selamanya, sudah senantiasa mendukung dan saling mengingatkan dalam menyelesaikan skripsi ini.

9. Yogi Takbul Perminta yang sudah memberi doa, mengingatkan, dan menyemangati penulis dalam menyelesaikan skripsi ini, semoga dipermudah juga untuk Yogi menyelesaikan Tugas Akhirnya.

10. Teman-teman KKN penulis, Acit, Ode, Vira, Eka, Ogie, Hanun, Sharon, Opik, dan Danu, serta orang-orang Subang yang sudah bersedia menjadi bagian dari perjalanan kuliah penulis, bersedia menjadi keluarga kedua

penulis selama di Subang. Terima Kasih Bapak dan Ibu Kades Cimayasari, A'Apip, Ian, dan keluarga besar di sana yang tidak bisa penulis sebutkan satu persatu namun tidak luput dari ucapan terima kasih penulis, serta keluarga besar SDN Karya Utama yang telah memberikan kesempatan kepada penulis untuk mengajar di sana.

11. Keluarga besar SDN Sukatani VII, SMPN 11 Depok, dan SMAN 99 Jakarta, terima kasih telah menjadi bagian dari tempat di mana penulis menuntut ilmu sehingga ilmu yang diberikan bermanfaat bagi orang lain, selalu menjadi sekolah yang terbaik, dan memberikan pelajaran yang terbaik bagi semua muridnya.

12. Teman-teman Paskibra SMA 99 Jakarta (ELEMENTAL), Aldi, Resti, Galuh, Endah, Bertie, Fathia, Aghita, Listyan, Erwin, Reza, Icad, Ulfa, Tiwi, Irun, Axel, dan Fatrian yang selalu menjadi penghibur penulis disaat jenuh dan selalu mendoakan agar cepat lulus.

13. Nita, Andre, Ady, Widya, Yevi, dan teman-teman SDN Sukatani VII yang tidak bisa penulis sebutkan satu persatu yang selalu mengingatkan untuk cepat menyelesaikan skripsi penulis.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Masukan dan kritikan akan sangat berarti. Semoga skripsi ini dapat bermanfaat bagi pembaca sekalian.

Depok, Januari 2017

Anggita Dyah Ayu Pitaloka

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Produksi merupakan suatu kegiatan yang dilakukan untuk menambah nilai guna suatu barang atau menciptakan barang baru sehingga barang tersebut dapat lebih dimanfaatkan untuk memenuhi kehidupan. Sedangkan tempat untuk melakukan kegiatan produksi dan berkumpulnya faktor-faktor produksi disebut perusahaan.

Pada kegiatan ekonomi, modal merupakan hal yang sangat dibutuhkan. Modal merupakan salah satu dari faktor produksi berupa barang atau jasa. Setiap perusahaan yang memproduksi suatu barang menginginkan keuntungan yang sebesar-besarnya dengan modal yang sekecil-kecilnya. Oleh karena itu, perusahaan harus menentukan barang-barang yang akan dipilih untuk menjamin kelancaran usahanya. Misalkan dalam masalah pengangkutan barang, dengan kapasitas yang tersedia diharapkan dapat mengangkut barang yang memiliki keuntungan yang maksimal dari barang-barang tersebut. Masalah seperti ini disebut masalah *knapsack* (*knapsack problem*).

Knapsack merupakan salah satu permasalahan yang dapat diselesaikan dengan cara yang berbeda-beda. *Knapsack* merupakan suatu permasalahan optimalisasi, di mana dihadapkan pada persoalan optimasi pemilihan objek ke dalam suatu wadah yang memiliki keterbatasan ruang atau daya tampung. Adanya optimasi dalam pemilihan objek yang akan di masukan ke dalam wadah tersebut diharapkan dapat menghasilkan keuntungan yang maksimal. Pada

perkembangan teknologi saat ini, pencarian optimal dapat dilakukan dengan menggunakan komputer dalam penyelesaiannya. Oleh karena itu, untuk menjalakan suatu program di komputer, membutuhkan metode yang tepat agar hasil yang diperoleh sesuai dengan yang diharapkan. Masalah *knapsack* dapat diselesaikan dengan beberapa algoritma seperti algoritma *Greedy*, algoritma Genetika, algoritma *Branch and Bound*, algoritma *Dynamic Programming*, *Brute Force*, dan lain sebagainya.

Pada penelitian sebelumnya yang berjudul "Penerapan Algoritma Genetika *Hybrid* pada Permasalahan *Bounded Knapsack*" ditulis oleh Ivan (2015). Dalam tulisannya, penulis mencari peranan dan probabilitas algoritma dalam pencarian nilai optimum. Selain itu, dilakukan penelitian oleh Iwan dan Djoko yang berjudul "Menentukan Lintasan Terpendek (*Shortest Path*) dengan 0/1 *Knapsack Problem* dan Pendekatan Algoritma *Dynamic Programming*" yang menuliskan bahwa dalam menyelesaikan suatu persoalan *shortest path* dengan *knapsack* untuk menentukan lintasan terpendek dan biaya termurah dari sumber-sumber s sampai ke tujuan t pada suatu lintasan, dapat diselesaikan dengan metode mundur yang merupakan salah satu persoalan dari permasalahan *knapsack* dan persoalan ini juga merupakan persoalan 0-1 *Dynamic Programming*.

Berdasarkan permasalahan di atas, maka penulis menggunakan algoritma *Dynamic Programming* dalam menyelesaikan masalah *knapsack* agar keuntungan yang diperoleh menjadi maksimal. *Knapsack* yang akan digunakan penulis yaitu jenis *knapsack* 0-1, di mana jika jenis barang tersebut dipilih, maka bernilai 1. Namun, jika jenis barang tersebut tidak dipilih, maka bernilai 0. Dengan demikian, dalam penulisan skripsi ini penulis mengangkat judul "**Permasalahan Optimasi *Knapsack* 0-1 dengan Menggunakan Algoritma *Dynamic Programming***".

1.2 Perumusan Masalah

Rumusan masalah yang dikaji berdasarkan latar belakang di atas adalah sebagai berikut:

1. Bagaimana cara Algoritma *Dynamic Programming* menyelesaikan permasalahan *knapsack* 0-1?

2. Bagaimana model optimasi yang diperoleh untuk permasalahan *knapsack* 0-1 dengan menggunakan Algoritma *Dynamic Programming*?

1.3 Pembatasan Masalah

Pembatasan masalah dalam penulisan ini adalah sebagai berikut:

1. Metode optimasi yang digunakan yaitu metode optimasi untuk memaksimalkan keuntungan.

2. Algoritma optimasi yang digunakan yaitu algoritma optimasi deterministik.

3. Studi kasus yang digunakan merupakan data dari 72 jenis minuman ringan yang diambil dari suatu agen minuman.

4. Data penelitian yang digunakan dalam kasus ini berupa data harga beli tiap karton barang, harga jual barang, berat barang, dan kapasitas maksimal alat pengangkut barang tersebut (mobil).

5. Alat angkut (mobil) yang digunakan merupakan mobil Carry Pick Up dengan daya tampung sebesar 1.800 kg.

1.4 Tujuan Penulisan

Tujuan dari penulisan skripsi ini adalah mengetahui cara Algoritma *Dynamic Programming* menyelesaikan permasalahan *knapsack* 0-1 dan mencari model optimasi untuk permasalahan *knapsack* 0-1 dengan menggunakan Algoritma *Dynamic Programming*.

1.5 Manfaat Penulisan

Manfaat yang diharapkan dari penulisan skripsi ini adalah memberikan solusi kepada pembaca, perusahaan, dan usaha dagang dalam menentukan pemilihan barang yang akan dibeli agar keuntungan yang diperoleh usaha tersebut menjadi maksimal.

1.6 Metode Penelitian

Teknik pengumpulan data yang dilakukan dalam penulisan skripsi ini adalah berupa kajian pustaka dengan mengumpulkan literatur bacaan berupa jurnal, internet, dan *textbook* yang mendukung penulisan ini. Metode yang digunakan adalah metode optimasi untuk mencari keuntungan dengan menggunakan Algoritma *Dynamic Programming*.

BAB II

LANDASAN TEORI

2.1 Optimasi

Setiap perusahaan atau organisasi memiliki keterbatasan atas sumber dayanya, baik keterbatasan dalam jumlah bahan baku, mesin dan peralatan kerja, ruang, tenaga kerja, jam kerja, maupun modal yang dimiliki perusahaan atau organisasi tersebut. Dengan keterbatasan tersebut, maka perlu adanya rencana atau strategi yang dapat mengoptimalkan hasil yang ingin dicapai, baik untuk memaksimalkan keuntungan atau meminimalkan biaya. Dengan demikian, teknik optimasi dibutuhkan dalam menyelesaikan permasalahan tersebut.

Optimasi adalah suatu usaha memperoleh solusi yang terbaik diantara banyak solusi yang ada dengan mempertimbangkan syarat-syarat yang diberikan dalam suatu permasalahan. Optimasi diperlukan dalam melakukan suatu usaha secara efektif dan efisien untuk mencapai target hasil yang ingin dicapai. Optimasi merupakan bagian dari program linier. Di mana di dalamnya terdapat fungsi tujuan dan fungsi kendala. Fungsi tujuan yaitu fungsi yang menggambarkan suatu tujuan atau sasaran yang ingin dicapai oleh suatu perusahaan dengan menggunakan sumber daya yang dimiliki untuk memperoleh keuntungan yang maksimal atau biaya yang minimal. Pada umumnya nilai yang akan dioptimalkan dilambangkan dengan "Z". Sedangkan yang dimaksud dengan fungsi kendala yaitu fungsi yang menggambarkan suatu batasan-batasan kapasitas yang tersedia yang dihadapi oleh suatu perusahaan untuk mencapai tujuan atau optimalitas tersebut.

Berikut ini adalah beberapa persoalan optimasi yang biasanya ditemukan dalam bentuk program linier (Munir, 2005):

1. Menentukan lintasan terpendek dari suatu tempat ke tempat yang lainnya, agar perjalanan menjadi lebih cepat dan efisien.

2. Menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya untuk pekerja dapat diminimalkan dan hasil dari produksi tersebut tetap maksimal.

3. Mengatur rute perjalanan kendaraan umum agar semua lokasi dapat dijangkau oleh kendaraan tersebut.

4. Pemilihan barang dari sekumpulan barang yang tersedia pada permasalahan *knapsack* dengan tujuan untuk mencari keuntungan yang maksimal.

2.2 Algoritma Optimasi

Algoritma adalah setiap prosedur komputasi yang terdefinisi dengan baik yang mengambil beberapa nilai atau seperangkat nilai-nilai, sebagai masukan (*input*) dan menghasilkan beberapa nilai atau seperangkat nilai-nilai sebagai pengeluaran (*output*).

Algoritma optimasi dapat didefinisikan sebagai algoritma yang digunakan untuk mencari nilai x sedemikian hingga menghasilkan fungsi x ($f(x)$) yang mempunyai nilai yang terkecil (minimum) atau yang terbesar (maksimum) untuk suatu fungsi f yang diberikan, yang umumnya disertai dengan beberapa batasan pada x. Di mana nilai x tersebut dapat berupa skalar atau vektor dari nilai-nilai kontinu maupun diskrit.

Berdasarkan metode operasinya, algoritma optimasi dapat dibagi menjadi dua, yaitu algoritma deterministik (*deterministic*) dan probabilistik (*probabilistic*). Pada setiap langkah eksekusi dalam algoritma deterministik, terdapat satu jalan untuk diproses. Jika tidak ada jalan lain, maka algoritma tersebut artinya sudah selesai. Beberapa algoritma yang termasuk ke dalam algoritma deterministik yaitu *State Space Search*, *Dynamic Programming*, dan *Branch and Bound*. Algoritma deterministik menghasilkan solusi yang tetap untuk suatu *input* yang diberikan. Algoritma ini biasanya digunakan untuk masalah yang ruang solusinya tidak terlalu besar. Sedangkan untuk permasalahan dengan ruang solusi yang sangat besar bahkan tidak terbatas biasanya algoritma yang digunakan yaitu algoritma probabilisitk. Algoritma optimasi yang termasuk ke dalam algoritma probabilistik yaitu metode Monte Carlo. Metode Monte Carlo berstandar pada proses pengambilan sampel secara acak yang berulang-ulang untuk menghasilkan solusi. Metode Monte Carlo dilakukan apabila permasalahan tersebut tidak dapat diselesaikan dengan algoritma deterministik. Algoritma-algoritma yang termasuk ke dalam Monte Carlo salah satunya yaitu *Swarm Intelligence* (SI). Algoritma optimasi yang termasuk ke dalam kelas *Swarm Intelligence* (SI) diantaranya adalah *Particle Swarm Optimization*(PSO), *Ant Colony Optimization* (ACO), *Cat Swarm Optimization* (CSO), dan lain sebagainya.

2.3 *Knapsack Problem*

Knapsack problem dipelajari secara intensif sejak akhir tahun 50-an yang dirintis oleh Dantzig, baik karena diaplikasikan langsung dalam industri dan manajemen keuangan mereka untuk alasan teoritis, sebagai masalah *knapsack* yang sering terjadi dari berbagai permasalahan *integer programming (Pisinger,*

1995).

Knapsack merupakan optimasi pengangkutan suatu barang atau disebut juga optimasi kombinatorial. *Knapsack problem* adalah suatu masalah cara pemilihan barang dari sekumpulan barang yang diberikan di mana setiap barang mempunyai berat (*weight*) dan keuntungan (*profit*) yang berbeda-beda, sehingga dengan kapasitas yang tersedia diharapkan pemilihan barang tersebut dapat menghasilkan keuntungan yang maksimal tanpa melebihi kapasitas muat dari *knapsack* tersebut.

Knapsack terdiri dari beberapa persoalan yaitu sebagai berikut(Martello,2006):

1. *Knapsack 0-1*(*Integer Knapsack*)

   Barang yang di masukan ke dalam *knapsack* 0-1 harus dimasukkan semua atau tidak sama sekali. Jika diberikan satu set n barang dan satu *knapsack* dengan:

   r_k = Keuntungan (*profit*) dari barang ke-k

   w_k = Berat (*weight*) dari barang ke-k

   c = Kapasitas *Knapsack*

   Maka, untuk mencari keuntungan yang maksimal persamaan yang memenuhi adalah sebagai berikut:

   maksimum:

   $$\Sigma_{k=1}^{n} r_k x_k \tag{2.1}$$

   kendala:

   $$\Sigma_{k=1}^{n} w_k x_k \leq c \tag{2.2}$$

   Di mana nilai $x_k = 0$ atau $1; k = 1, 2, 3, ..., n$. Jika x_k bernilai 1 maka barang ke-k terpilih. Sebaliknya, jika x_k bernilai 0 maka barang ke-k tidak terpilih.

2. *Knapsack* terbatas (*Bounded Knapsack*)

   Barang yang di masukan ke dalam *knapsack* terbatas bisa dimasukkan sebagian atau seluruhnya. Jika diberikan n barang dan satu *knapsack*, dengan:

   r_k = Keuntungan (*profit*) dari barang ke-k

   w_k = Berat (*weight*) dari barang ke-k

   b_k = Batas atas yang tersedia dari barang ke-k

   c = Kapasitas *Knapsack*

   Maka, untuk mencari keuntungan yang maksimal persamaan yang memenuhi adalah sebagai berikut:

   maksimum:

$$\Sigma_{k=1}^{n} r_k x_k \tag{2.3}$$

   kendala:

$$\Sigma_{k=1}^{n} w_k x_k \leq c \tag{2.4}$$

   Di mana $0 \leq x_k \leq b_k$ dan *integer*; $k = 1, 2, 3, ..., n$.

3. *Knapsack* tak terbatas (*Unbounded Knapsack*)

   Barang yang dimasukkan ke dalam *knapsack* jumlahnya tidak terbatas. Jika diberikan n barang dan satu *knapsack* sejumlah $m(m \leq n)$, dengan

   r_k = Keuntungan (*profit*) dari barang ke-k

   w_k = Berat (*weight*) dari barang ke-k

   c = Kapasitas *Knapsack*

   sehingga,

   Maksimum:

$$\Sigma_{k=1}^{n} r_k x_k \tag{2.5}$$

kendala:

$$\Sigma_{k=1}^{n} w_k x_k \leq c \tag{2.6}$$

Di mana $x_k \geq 0$ *integer*; $k = 1, 2, 3, ...n$.

Secara umum, *knapsack* dapat dirumuskan sebagai berikut:
fungsi tujuan memaksimumkan/meminimumkan:

$$\Sigma_{k=1}^{n} r_k x_k \tag{2.7}$$

kendala:

$$\Sigma_{k=1}^{n} w_k x_k \leq c \tag{2.8}$$

dengan,
w_k = berat masing-masing jenis barang k ; $k = 1, 2, 3, ..., n$,
r_k = nilai atau keuntungan pada setiap barang k ; $k = 1, 2, 3, ..., n$,
x_k = jumlah barang ke-k ; $k = 1, 2, 3, ..., n$,
c = kapasitas pada *knapsack*.

di mana $x_1, x_2, x_3, ..., x_n$ merupakan bilangan bulat *non-negative*. Karena x_k bernilai *integer*, maka permasalahan *knapsack* 0-1 dinamakan sebagai program *integer* (*integer programing*). Pada permasalahan *integer knapsack*, barang yang diangkut harus diangkut semua atau tidak sama sekali. Barang yang diangkut semua diberi nilai 1, sedangkan yang lainnya diberi nilai 0. Misalkan terdapat himpunan barang:

$$X = \{x_1, x_2, x_3, ..., x_n\}$$

Jika barang yang diangkut semua adalah barang ke-2 dan ke-3 saja, maka solusi yang didapatkan dinotasikan sebagai $X = \{0, 1, 1, ..., 0\}$.

Contoh 2.3.1. Misalkan terdapat lima objek yang akan diberi nomer 1, 2, 3, 4, dan 5. Masing-masing objek tersebut memiliki berat dan keuntungan sebagai berikut:

$$w_1 = 5; r_1 = 30$$

$$w_2 = 2; r_2 = 10$$

$$w_3 = 3; r_3 = 20$$

$$w_4 = 2; r_4 = 35$$

$$w_5 = 6; r_5 = 30$$

Kapasitas *knapsack* 0-1 = 15

Dilihat dari kapasitas *knapsack* 0-1 yang diberikan, maka harus dilakukan suatu pemilihan objek yang akan dipilih semua atau tidak sama sekali di mana objek tersebut memiliki keuntungan maksimal dan tidak melebihi kapasitas yang telah diberikan. Oleh karena itu, permasalahan di atas dapat diselesaikan sebagai berikut :

Diketahui :

Banyaknya jenis objek $(n) = 5$

Berat tiap jenis objek ke-k (w_k) dan keutungan tiap jenis objek ke-k (r_k):

$w_1 = 5; r_1 = 30$

$w_2 = 2; r_2 = 10$

$w_3 = 3; r_3 = 20$

$w_4 = 2; r_4 = 35$

$w_5 = 6; r_5 = 30$

Kapasitas maksimal *knapsack* $(c) = 15$

Ditanya :

1. Objek apa sajakah yang dipilih agar dapat memaksimalkan keuntungan dengan keterbatasan kapasitas yang diberikan?

2. Berapa besar keuntungan yang diperoleh dari pemilihan barang tersebut?

Penyelesaian:

Tabel 2.1: Langkah Pencarian Solusi *Integer Knapsack*

Himpunan Bagian	Total Berat	Total Keuntungan
{}	0	0
{1}	5	30
{2}	2	10
{3}	3	20
{4}	2	35
{5}	6	30
{1,2}	7	40
{1,3}	8	50
{1,4}	7	65
{1,5}	11	60
{1,2,3}	10	60
{1,2,4}	9	75
{1,2,5}	13	70
{1,3,4}	10	85
{1,3,5}	14	80
{1,4,5}	13	95
{1,2,3,4}	12	95
{1,2,3,5}	16	Tidak Layak
{1,2,4,5}	**15**	**105**
{1,2,3,4,5}	18	Tidak Layak

Dari tabel di atas maka diketahui himpunan bagian objek yang memberikan keuntungan maksimal adalah {1,2,4,5} dengan total keuntungan 105. Dengan demikian, solusi persoalan *Integer Knapsack* di atas adalah $X = \{1, 1, 0, 1, 1\}$.

2.3.1 Beberapa Metode Pemecahan Masalah *Knapsack 0-1*

Algoritma optimasi dibagi menjadi dua, yaitu algoritma deterministik (*deterministic*) dan algoritma probabilistik (*probabilistic*). Pada permasalahan

knapsack 0-1 terdapat beberapa algoritma yang mampu menyelesaikan permasalahan ini di antaranya seperti algoritma *Branch and Bound, Brute Force*, Algoritma Genetika, Algoritma *Dynamic Programming*, Algoritma *Greedy*, dan lain sebagainya. Dari algoritma-algoritma tersebut mempunyai caranya tersendiri dalam menyelesaikan masalah *knapsack* 0-1. Adapun cara dari beberapa algoritma adalah sebagai berikut:

1. *Branch and Bound*

   Branch and Bound adalah metode algoritmik general untuk menemukan solusi optimal dari berbagai masalah optimasi, khususnya pada diskrit dan optimasi kombinatorial. Dasarnya adalah dengan pendekatan enumerasi dengan cara mematikan *search space* yang tidak mengarah kepada solusi.

2. *Brute Force*

   Brute Force adalah sebuah pendekatan langsung (*straight forward*) untuk memecahkan suatu masalah, yang biasanya didasarkan pada pernyataan masalah dan definisi konsep yang dilibatkan. Pada dasarnya algoritma *Brute Force* mempunyai alur penyelesaian yang sederhana dan tidak memerlukan pemikiran yang lama.

3. Algoritma Genetika

   Algoritma Genetika adalah algoritma komputasi yang terinspirasi dari teori evolusi untuk mencari solusi suatu permasalahan dengan cara yang alamiah. Proses yang dilakukan Algoritma Genetika yaitu dengan cara proses reduksi, *crossover*, dan mutasi. Algoritma Genetika bergerak dan mencari sejumlah solusi sekaligus, selama solusi tersebut masih bersifat *feasible*. Dengan parameter yang tepat, diharapkan salah satu dari seki-

an solusi yang dibangkitkan oleh Algoritma Genetika merupakan solusi optimum global.

4. Algoritma *Dynamic Programming*

   Algoritma *Dynamic Programming* adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang berkaitan.

5. Algoritma *Greedy*

   Algoritma *Greedy* merupakan suatu algoritma yang lazim untuk memecahkan masalah optimasi meskipun hasilnya tidak selalu optimum. Prinsip utama Algoritma *Greedy* yaitu dengan mengambil sebanyak mungkin apa yang diperoleh sekarang. Algoritma ini sering digunakan untuk menyelesaikan masalah optimum dengan teknik umum menggunakan *heuristic* untuk mencari solusi suboptium sehingga diharapkan solusi optimum.

2.4 Program *Dynamic* (*Dynamic Programming*)

Dynamic Programming adalah metoda rancangan algoritma yang dapat dipakai bila atau pada saat pemecahan masalah yang mungkin dipandang sebagai hasil dari suatu rangkaian keputusan-keputusan (Suryadi, 1996). Istilah *Dynamic Programming* terjadi karena adanya kecenderungan metode ini dalam menganalisa hasil perhitungan pada setiap tahapnya menggunakan beberapa tabel sehingga solusi yang diperoleh dapat diketahui detailnya dengan mudah dan dapat lebih mudah dimengerti. Program ini dikembangkan oleh Richard

Bellman dan G. B Dantzing pada tahun 1940-1950. *Dynamic Programming* dapat diaplikasikan pada pengelolaan persediaan, jaringan, penjadwalan kerja untuk karyawan, pengendalian produksi, perencanaan penjualan, dan lain sebagainya.

Dynamic Programming adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang berkaitan. Dalam menentukan total keuntungan yang optimal, maka terlebih dahulu dilakukan pencarian keuntungan pada setiap kelompok barang dengan kemungkinan kapasitas muat barang yang tersedia. Perhitungan disetiap kelompok barang dilakukan dengan menggunakan persamaan rekursif dan selanjutnya menghasilkan penyelesaian optimal yang mungkin bagi seluruh barang (Taha, 1996). Secara rekursif artinya bahwa pada saat mengambil keputusan harus terlebih dahulu memperlihatkan keadaan yang dihasilkan oleh keputusan optimal dari tahap sebelumnya dan keputusan tersebut merupakan landasan bagi keputusan optimal pada tahap selanjutnya. Jadi, pada program *dynamic* keputusan yang optimal dibuat dengan menggunakan prinsip optimalitas, yaitu jika solusi total optimal maka bagian solusi sampai tahap ke-k juga optimal.

2.4.1 Konsep Dasar dalam *Dynamic Programming*

Pada *Dynamic Programming* terdapat beberapa konsep dasar di antaranya adalah sebagai berikut:

1. Dekomposisi

   Pada konsep dekomposisi *Dynamic Programming* membagi persoalan menjadi subpersoalan atau tahapan yang lebih kecil dan berurutan. Se-

tiap tahap yang ada memiliki suatu keputusan tersendiri, di mana keputusan yang ada pada setiap tahap akan mempengaruhi keputusan pada tahap selanjutnya. Artinya, keputusan dari tahap sekarang tergantung dari keputusan tahap sebelumnya.

2. Status

   Status akhir pada setiap tahap dipengaruhi oleh status awal dan keputusan yang dibuat pada tahap tersebut. Selanjutnya, status akhir pada suatu tahap merupakan *input* pada tahap selanjutnya.

3. Variabel Keputusan dan Akhir

   Keputusan yang dibuat pada setiap tahap merupakan keputusan yang mengakibatkan tingkat minimal atau maksimal.

4. Fungsi Transisi

   Pada fungsi transisi menjelaskan bahwa setiap tahapan-tahapan yang ada mempunyai suatu hubungan satu antar lainnya, yaitu hubungan antar status pada setiap tahapan yang berurutan.

5. Optimasi Tahap

   Optimasi tahap pada *Dynamic Programming* yaitu menentukan nilai yang optimal pada setiap tahap tersebut yang dapat dilihat dari berbagai kemungkinan yang ada pada status *input* yang diberikan.

6. Fungsi Rekursif

   Nilai sebuah variabel dari fungsi rekursif merupakan nilai akumulatif dari nilai variabel tersebut pada tahap sebelumnya.

Program *dynamic* tidak seperti program linier, karena pada program *dynamic* tidak ada bentuk matematis standar untuk perumusannya. Namun, pro-

gram *dynamic* merupakan pendekatan umum untuk memecahkan masalah dan persamaan tertentu yang digunakannya harus sesuai dengan permasalahan yang ada. Oleh karena itu, tingkat pemahaman dan kreatifitas sangat diperlukan dalam memahami suatu permasalahan dengan menggunakan program *dynamic* karena setiap permasalahan memiliki cara yang berbeda dalam mengatasinya.

Algoritma *Dynamic Programming* memiliki beberapa kriteria, di antaranya adalah sebagai berikut:

1. Persoalan dalam *dynamic programming* dapat dibagi menjadi beberapa tahap (*stage*), di mana setiap tahap menghasilkan suatu keputusan yang optimal dan saling berhubungan antar tahap yang satu dengan tahap lainnya.

2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan kondisi awal tahap tersebut. Secara umum, *state* adalah berbagai kondisi sistem yang mungkin pada suatu tahap masalah. Banyaknya *state* bisa terbatas ataupun tidak terbatas, sesuai dengan masalah yang ada.

3. Efek keputusan kebijakan pada setiap tahap adalah mengubah *state* saat ini menjadi *state* lain pada awal tahap berikutnya.

4. Ongkos (*cost*) pada suatu tahap bergantung pada ongkos tahap-tahap sebelumnya dan meningkat secara teratur dengan bertambahnya jumlah tahapan.

5. Keputusan terbaik pada suatu tahap bersifat *independen* terhadap keputusan yang dilakukan tahap sebelumnya. Oleh karena itu, keputusan yang optimal selanjutnya hanya tergantung pada *state* saat ini dan bu-

kan cara mencapai *state* saat ini. Hal ini merupakan prinsip optimalitas untuk program *dynamic*.

6. Adanya hubungan rekursif, yaitu hubungan yang mengidentifikasi keputusan pada tahap k mempengaruhi keputusan pada tahap $k+1$.

7. Menggunakan prinsip optimalitas dalam penyelesaiannya.

2.4.2 Analisis Algoritma *Dynamic Programming*

Pada *Dynamic Programming* rangkaian keputusan dibuat dengan menggunakan prinsip optimalitas. Prinsip optimalitas yaitu jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Hal tersebut dapat diartikan bahwa jika suatu persoalan dikerjakan dari tahap ke-k sampai dengan tahap ke-$k+1$, maka hasil optimal yang digunakan bisa dari tahap ke-k tanpa harus mencari kembali hasil optimal dari tahap awal. Secara konsep, Algoritma *Dynamic Programming* menggunakan konsep sub-optimasi. Maka, dengan demikian sebelum melaksanakan proses optimasi suatu persoalan perlu mengetahui konsep sub-optimasi berikut ini:

1. Tahap pertama tidak dipengaruhi tahap yang lainnya. Jadi, pada tahap ke-1 dapat dioptimumkan tersendiri yang merupakan sub-optimasi yang pertama.

2. Mendapatkan sub-optimasi yang ke-2 yaitu dengan menggunakan penyelesaian pada tahap pertama digabungkan dengan tahap yang kedua.

3. Mendapatkan sub-optimasi pada tahap ke-3 yaitu dengan menggunakan penyelesaian tahap kedua digabungkan dengan tahap ketiga. Demikian seterusnya sampai pada tahap ke-n.

Dengan demikian, prinsip optimalitas menjamin bahwa keputusan pada suatu tahap merupakan keputusan yang benar pada tahap-tahap selanjutnya.

2.4.3 Pendekatan Algoritma *Dynamic Programming* Secara Rekursif

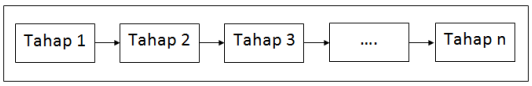
Algoritma *Dynamic Programming* menggunakan teknik perhitungan secara rekursif (bersifat pengulangan) yang diketahui sebagai prinsip optimalisasi. Jika pada suatu proses menghitung perolehan optimal sampai pada tahap ke-n, maka selesailah prosedur perhitungan berdasarkan program dinamik.

Pada Algoritma *Dynamic Programming* terdapat dua pendekatan rekursif, yaitu pendekatan maju (*forward* atau *up-down*) dan pendekatan mundur (*backward* atau *bottom-up*). Jika dimisalkan $x_1, x_2, x_3, ..., x_n$ menyatakan variabel keputusan yang harus dibuat masing-masing untuk tahap $1, 2, 3, ..., n$. Maka:

1. Pendekatan maju (*forward*) atau *up-down*

   Prosedur pendekatan maju ini merupakan suatu prosedur di mana perhitungannya dimulai dari tahap awal sampai tahap akhir. Bergerak dari tahap 1, dilanjutkan ke tahap $2, 3, ..., n$. Di mana n merupakan banyaknya jumlah barang. Urutan *variable* keputusannya yaitu $x_1, x_2, x_3, ..., x_n$. Pada pendekatan maju (*forward*) atau *up-down* prinsip optimalitasnya adalah sebagai berikut :

   ongkos pada tahap $k + 1$ = (ongkos yang dihasilkan pada tahap k) + (ongkos dari tahap k ke tahap $k+1$) ; $k = 1, 2, 3, ..., n-1$. Dengan demikian, perhitungan yang dilakukan oleh prosedur *forward* pada *Dynamic Programming* dapat digambarkan sebagai berikut:

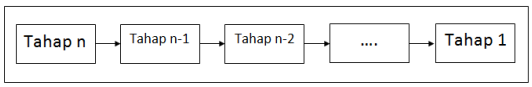


Gambar 2.1: Perhitungan Prosedur *Forward* pada *Dynamic Programming*

2. Pendekatan mundur (*backward*) atau *bottom-up*

   Prosedur pendekatan mundur ini merupakan suatu kebalikan dari prosedur maju, di mana perhitungan dimulai dari tahap terakhir ke tahap awal. Bergerak mulai dari tahap n, mundur ke tahap $n-1, n-2, ..., 1$. Urutan *variable* keputusannya yaitu $x_n, x_{n-1}, x_{n-2}, ..., x_1$. Pada pendekatan mundur (*backward*) atau *bottom-up* prinsip optimalitasnya adalah sebagai berikut :

   ongkos pada tahap k = (ongkos yang dihasilkan pada tahap k+1) + (ongkos dari tahap k+1 ke tahap k) ; k = n, n-1, n-2, ..., 1. Dengan demikian, perhitungan yang dilakukan oleh prosedur *backward* pada *Dynamic Programming* dapat digambarkan sebagai berikut:

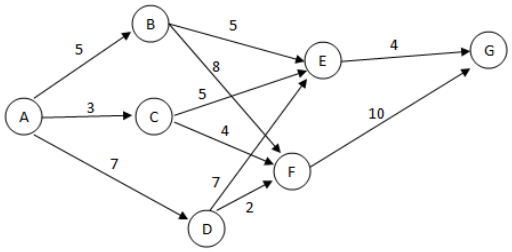


Gambar 2.2: Perhitungan Prosedur *Backward* pada *Dynamic Programming*

Dari Gambar 2.1 dan Gambar 2.2, perbedaan dari kedua tahap tersebut adalah bagaimana cara untuk mendefinisikan tahap yang akan dipakai. Apakah menggunakan prosedur maju atau prosedur mundur dalam memulai perhitungannya.

Contoh 2.4.1. Terdapat sebuah lintasan yang terdiri dari 7 simpul, yaitu simpul A, B, C, D, E, F, dan G. Dari lintasan tersebut tentukan jarak terpen-

dek (*Shortest Path*) dengan menggunakan pendekatan maju (*forward*) dan pendekatan mundur (*backward*) pada *Dynamic Programming*.



Gambar 2.3: Lintasan

Penyelesaian:

1. Mencari lintasan terpendek dengan menggunakan pendekatan maju (*forward*) pada *Dynamic Programming*.

   Relasi rekurens berikut menyatakan lintasan terpendek dari status s ke x_3 pada tahap k:

   $f_1(s) = c_{x_1 s}$

   $f_k(s) = min_{x_k}\{c_{x_k s} + f_{k-1}(x_k)\}, ; k = 2, 3.$

   Keterangan:

   x_k : Peubah keputusan pada tahap $k(k = 2, 3)$.

   $c_{x_k s}$: Bobot sisi dari x_k ke s.

   $f_k(s)$: Nilai minimum dari $f_k(x_k, s)$.

   $f_k(x_k, s)$: Total bobot lintasan dari x_k ke s.

   Misalkan A, B, C, D, E, F dan G dinotasikan dengan (1, 2, 3, 4, 5, 6 dan 7).

   Tahap 1:

   $f_1(s) = c_{x_1 s}$

s	Solusi optimum	
	$f_1(s)$	x_1^*
2	5	1
3	3	1
4	7	1

Gambar 2.4: Tabel Perhitungan *forward* tahap 1 Lintasan Terpendek

Tahap 2:

$f_2(s) = min_{x_2}\{c_{x_2 s} + f_1(x_2)\}$

x_2 / s	$f_2(x_2,s) = c_{x_2,s} + f_1(x_2)$			Solusi Optimum	
	2	3	4	$f_2(s)$	x_2^*
5	10	8	14	8	3
6	13	7	9	7	3

Gambar 2.5: Tabel Perhitungan *forward* tahap 2 Lintasan Terpendek

Tahap 3:

$f_3(s) = min_{x_3}\{c_{x_3 s} + f_2(x_3)\}$

x_3 / s	$f_3(x_3,s) = c_{x_3,s} + f_2(x_3)$		Solusi Optimum	
	2	3	$f_3(s)$	x_3^*
7	10	8	8	3

Gambar 2.6: Tabel Perhitungan *forward* tahap 3 Lintasan Terpendek

Dengan demikian, ada satu lintsan terpendek dari A-G (1-7) yaitu 1 →3 →5 →7 atau A →C →E →G dengan panjang lintasan sebesar 12.

2. Mencari lintasan terpendek dengan menggunakan pendekatan mundur (*backward*) pada *Dynamic Programming*.

   Relasi rekurens berikut menyatakan lintasan terpendek dari status s ke x_3 pada tahap k:

$f_3(s) = c_{x_3 s}$

$f_k(s) = min_{x_k}\{c_{sx_k} + f_{k+1}(x_k)\}, ; k = 1, 2.$

Keterangan:

x_k : Peubah keputusan pada tahap $k(k = 1, 2)$.

c_{sx_k} : Bobot sisi dari s ke x_k.

$f_k(s)$: Nilai minimum dari $f_k(s, x_k)$.

$f_k(s, x_k)$: Total bobot lintasan dari s ke x_k.

Misalkan A, B, C, D, E, F dan G dinotasikan dengan (1, 2, 3, 4, 5, 6 dan 7).

Tahap 1:

$f_3(s) = c_{x_3 s}$

s	Solusi optimum	
	$f_3(s)$	x_3^*
5	4	7
6	10	7

Gambar 2.7: Tabel Perhitungan *backward* tahap 1 Lintasan Terpendek

Tahap 2:

$f_2(s) = min_{x_2}\{c_{sx_2} + f_3(x_2)\}$

s \ x_2	$f_2(s, x_2) = c_{s,x_2} + f_3(x_2)$		Solusi Optimum	
	5	6	$f_2(s)$	$x_2{}^*$
2	9	18	9	5
3	9	14	9	5
4	11	12	11	5

Gambar 2.8: Tabel Perhitungan *backward* tahap 2 Lintasan Terpendek

Tahap 3:

$f_1(s) = min_{x_1}\{c_{sx_1} + f_2(x_1)\}$

x_1 / s	$f_1(s,x_1) = c_{s,x_1} + f_2(x_1)$			Solusi Optimum	
	2	3	4	$f_1(s)$	x_1^*
1	14	12	18	12	3

Gambar 2.9: Tabel Perhitungan *backward* tahap 3 Lintasan Terpendek

Dengan demikian, ada satu lintasan terpendek dari G-A (7-1) yaitu 7 →5 →3 →1 atau G →E →C →A dengan panjang lintasan sebesar 12.

BAB III

PEMBAHASAN

3.1 0-1 *Knapsack Problem* dengan menggunakan *Dynamic Programming*

Sebuah pemecahan pada permasalahan *knapsack* 0-1 dianggap sebagai suatu rangkaian keputusan $x_1, x_2, x_3, ..., x_n$, di mana nilai x merupakan nilai yang menunjukan 0 atau 1. Menurut keputusannya, x_n berada dalam satu dari dua penyataan yang mungkin.

Terdapat beberapa tahap yang dimiliki program *dynamic* dalam menyelesaikan masalah *knapsack* 0-1. Tahap tersebut di antaranya adalah sebagai berikut (Hiller, 1994):

1. Tahap ke-k

   Tahap di mana proses memasukan barang ke dalam suatu tempat (*knapsack* 0-1).

2. Status ke-y

   Status yang menyatakan kapasitas muat *knapsack* 0-1 yang dimulai dari kapasitas sebesar 0 sampai kapasitas maksimal *knapsack* 0-1 sebesar M.

Pada pencarian keuntungan maksimal, pendekatan Algoritma *Dynamic Programming* yang digunakan adalah pendekatan maju. Nilai pada saat tahap $k = 0$ atau $(f_0(y))$ selalu nol, hal ini dikarenakan belum adanya beban (*weight*) yang ditampung oleh *knapsack* 0-1 atau belum adanya barang yang

dimasukkan ke dalam *knapsack* 0-1 tersebut (*knapsack* kosong). Jika kapasitas *knapsack* 0-1 lebih kecil dari pada kapasitas barang yang akan dimasukkan atau *knapsack* 0-1 memiliki kapasitas negatif, maka keuntungan dari *knapsack* 0-1 tersebut bernilai *minus* tak hingga. Cara memasukan barang ke dalam *knapsack* 0-1 melalui beberapa tahap atau disebut dengan tahap ke-k, di mana pada tahap ini barang di masukan satu persatu dari barang pertama ke dalam satuan kapasitas *knapsack* 0-1 sampai kapasitas yang mampu ditampung oleh *knapsack* 0-1 tersebut maksimal. Ketika memasukan barang pada tahap ke-k maka kapasitas awal *knapsack* 0-1 dikurangi oleh berat benda ke-k. Dengan demikian, kapasitas *knapsack* 0-1 yang tersedia menjadi:

$$y - w_k \tag{3.1}$$

Sedangkan untuk mengisi sisanya, diterapkan prinsip optimalitas dengan mengacu pada nilai optimum dari tahap sebelumnya untuk kapasitas sisa $y - w_k$ dinotasikan dengan,

$$f_{k-1}(y - w_k) \tag{3.2}$$

Tahap selanjutnya yaitu dengan membandingkan nilai keuntungan dari barang pada tahap ke-k (r_k) ditambah dengan nilai $f_{k-1}(y - w_k)$ dengan keuntungan pengisian hanya $k-1$ macam barang ($f_{k-1}(y)$). Jika $r_k + f_{k-1}(y - w_k) < f_{k-1}(y)$, maka barang ke-$k$ tidak dimasukkan ke dalam *knapsack* 0-1. Jika sebaliknya, maka dimasukkan ke dalam *knapsack* 0-1.

Dari penjabaran di atas, untuk mendapatkan keuntungan yang maksimal dari permasalahan *knapsack* 0-1 maka didapat relasi rekursif maju adalah sebagai berikut (Hiller, 1994):

$$f_0(y) = 0; \qquad k = 0, \;\; y = 0, 1, 2, 3, ..., M \tag{3.3}$$

$$f_k(y) = -\infty; \qquad k = 1, 2, 3, ..., n, \;\; y < 0 \tag{3.4}$$

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}; k = 1, 2, 3, ..., n. \tag{3.5}$$

dengan,

$f_0(y) = 0$ adalah nilai dari persoalan *knapsack* kosong dengan kapasitas yang dimiliki y,

$f_k(y) = -\infty$ adalah nilai dari *knapsack* 0-1 yang memiliki kapasitas negatif,

$f_k(y)$ adalah keuntungan yang optimum dari persoalan *knapsack* 0-1 pada tahap ke-k untuk kapasitas *knapsack* 0-1 sebesar y.

M = Kapasitas muat *knapsack* 0-1.

n = Banyaknya jenis barang.

Sedangkan, pada pencarian jenis barang dan total berat barang yang akan diangkut oleh *knapsack* 0-1, sebelumnya sudah ditentukan terlebih dahulu keuntungan maksimal yang diperoleh dari permasalahan tersebut, dikarenakan dalam pencariannya pendekatan Algoritma *Dynamic Programming* yang digunakan yaitu pendekatan secara mundur. Di mana nilai k dimulai dari $n, n-1, n-2, ..., 1$.

Ketika keuntungan maksimal sudah diperoleh, maka langkah pertama yaitu membandingkan keuntungan saat ini ($f_k(y)$) dengan keuntungan sebelumnya ($f_{k-1}(y)$) dari muatan kapasitas yang tersedia. Jika $f_k(y) \neq f_{k-1}(y)$ maka, $x_k = 1$. Sedangkan jika $f_k(y) = f_{k-1}(y)$ maka, $x_k = 0$.

Jika $x_k = 1$ maka keuntungan awal dikurangi dengan keuntungan barang ke-k, sisa keuntungan ini nantinya akan dijadikan keuntungan awal pada tahap selanjutnya. Hal ini dilakukan sampai keuntungan barang menjadi nol atau sampai tidak adanya barang yang dimasukkan ke dalam *knapsack* 0-1. Selain itu, nilai awal kapasitas muat *knapsack* 0-1 (y) juga dikurangi dengan berat benda ke-k, yang nantinya sisa muat *knapsack* 0-1 tersebut dijadikan sebagai muat awal kapasitas *knapsack* 0-1 pada tahap selanjutnya, tahap ini dilakukan sampai $k = 1$. Sedangkan jika nilai $x_k = 0$, maka banyaknya keuntungan dan muat kapasitas sama dengan keuntungan dan kapasitas sebelumnya. Selan-

jutnya mencari total berat maksimal yang dapat diangkut oleh *knapsack* 0-1, didapatkan dari jumlah berat barang yang dipilih atau bernilai 1 yang akan dimasukkan ke dalam *knapsack* 0-1, sedangkan barang yang bernilai 0, maka berat barang tersebut tidak dihitung atau dianggap 0.

Dari penjabaran di atas, untuk mendapatkan jenis barang yang dipilih secara rekursif mundur adalah sebagai berikut:

$$Untuk \quad y = M, jika \quad f_k(y) \neq f_{k-1}(y), maka \quad x_k = 1, \tag{3.6}$$

di mana $k = n, n-1, n-2, ..., 1 \quad dan \quad y = M, M-1, ..., 0$

sehingga,

$$u_k = f_k(y) \tag{3.7}$$

$$s_k = u_k - r_k, dan \tag{3.8}$$

$$q_k = y - w_k \tag{3.9}$$

$$M = q_k \tag{3.10}$$

$$Untuk \quad y = M, jika \quad f_k(y) = f_{k-1}(y), maka \quad x_k = 0, \tag{3.11}$$

di mana $k = n, n-1, n-2, ..., 1 \quad dan \quad y = M, M-1, ..., 0$

sehingga,

$$u_k = f_k(y) \tag{3.12}$$

$$s_k = u_k, dan \tag{3.13}$$

$$q_k = y \tag{3.14}$$

$$M = q_k \tag{3.15}$$

Keterangan:

$f_k(y)$ = Keuntungan yang optimum pada tahap ke-k untuk kapasitas muat

knapsack 0-1 sebesar y.

$f_{k-1}(y)$ = Keuntungan yang optimum pada tahap ke-$k-1$ untuk kapasitas muat *knapsack* 0-1 sebesar y.

s_k = Sisa keuntungan barang yang berada di dalam *knapsack* 0-1 setelah barang ke-k dikeluarkan .

u_k = Keuntungan maksimal pada saat tahap ke-k.

w_k = Berat barang ke-k.

r_k = keuntungan barang ke-k.

M = Kapasitas muat *knapsack* 0-1.

n = Banyaknya jenis barang.

q_k = Sisa muat *knapsack* 0-1 ketika barang ke-k dimasukkan.

x_k = Status barang apakah dipilih atau tidak. Jika barang tersebut dipilih maka diberi nilai 1, namun jika barang tersebut tidak dipilih maka diberi nilai 0.

Selanjutnya, dari hasil akhir sisa *knapsack* 0-1 dapat dilakukan pencarian total berat barang maksimal yang akan diangkut *knapsack* 0-1 sehingga keuntungan menjadi maksimal. Maka, total berat barang secara rekursif mundur dapat dinyatakan sebagai berikut :

$$W = \Sigma_{k=1}^{n} w_k \tag{3.16}$$

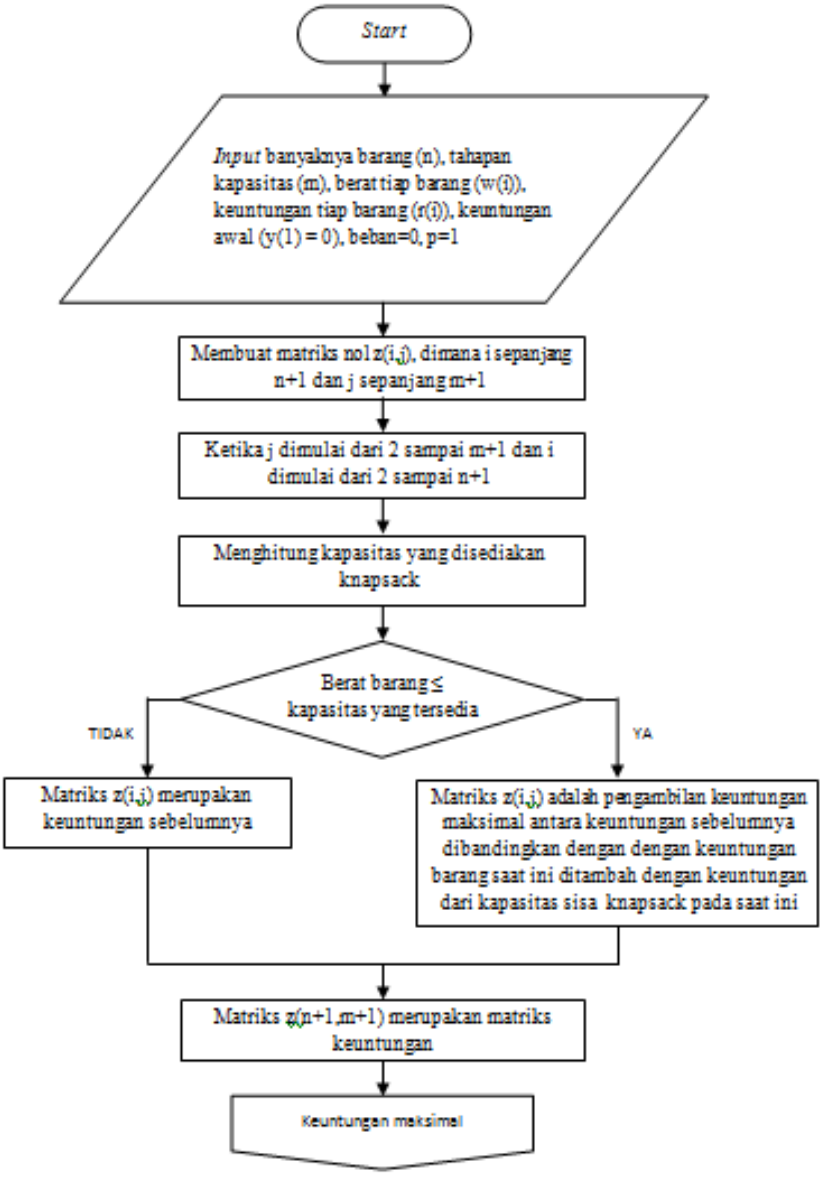
Keterangan:

W = Total berat maksimal yang dapat diangkut *knapsack* 0-1 tersebut.

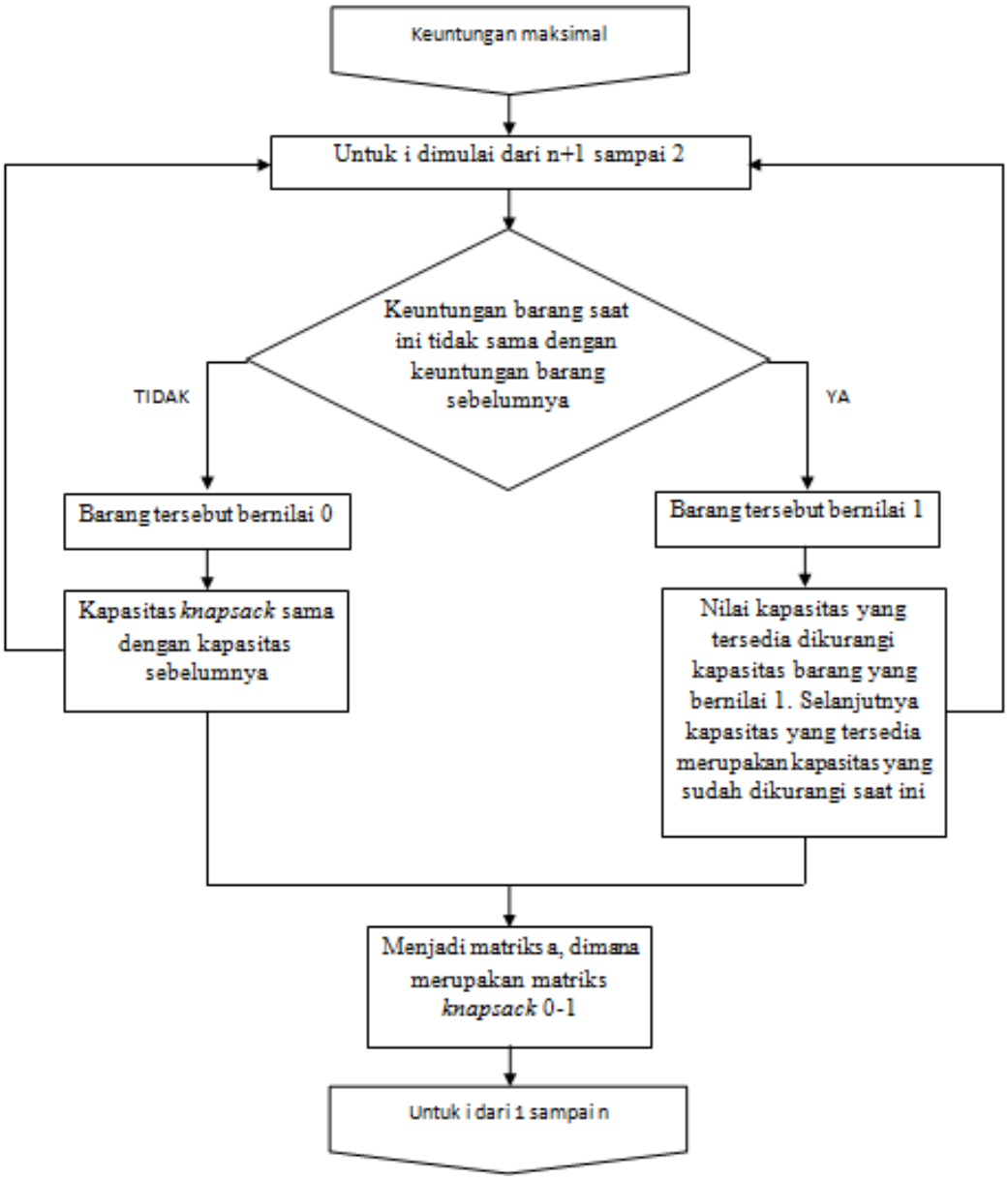
w_k = Berat barang ke-k.

n = Banyaknya jenis barang.

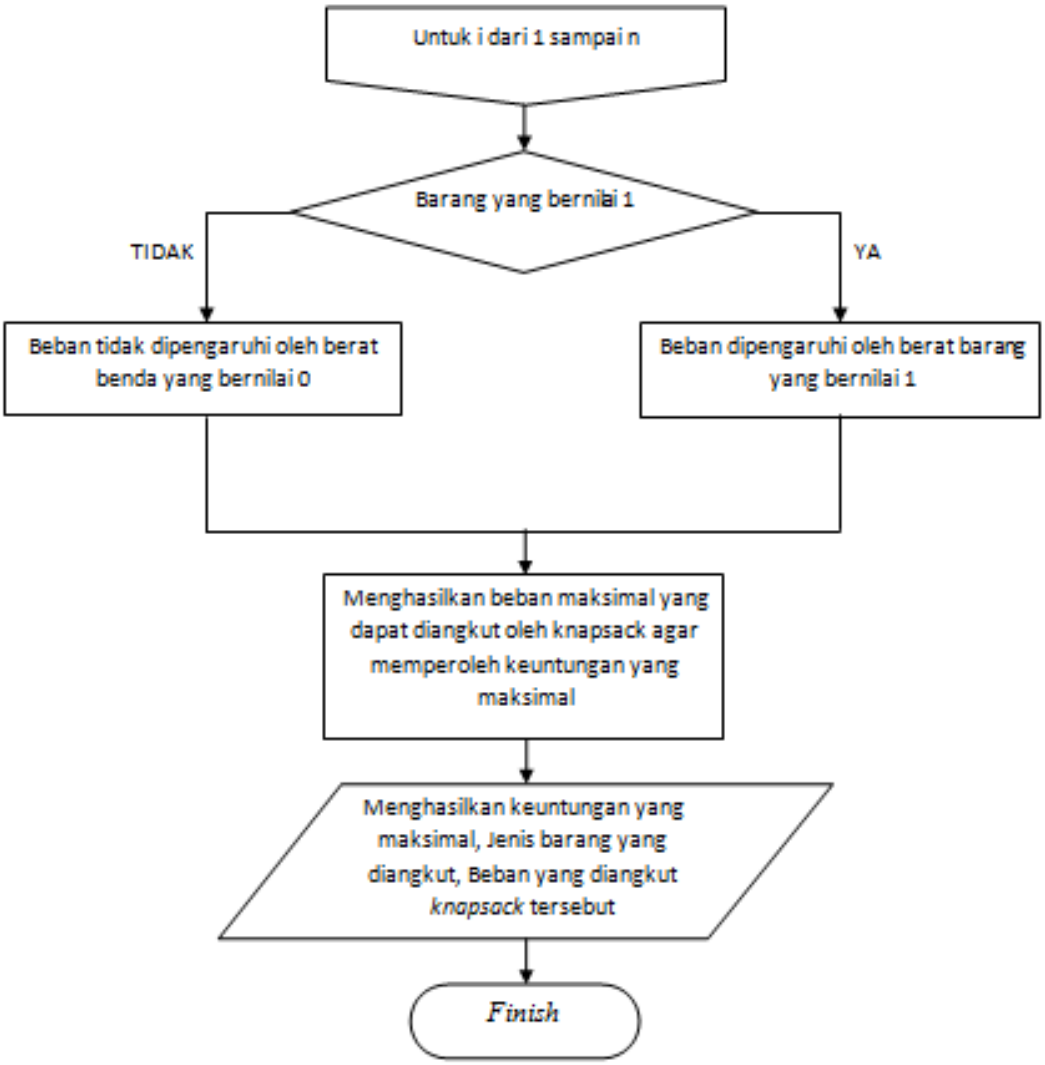
3.2 Langkah-Langkah Penyelesaian



Gambar 3.1: *Flowchart* pada Algoritma *Dynamic Programming*



Gambar 3.2: *Flowchart* pada Algoritma *Dynamic Programming*



Gambar 3.3: *Flowchart* pada Algoritma *Dynamic Programming*

Flowchart tersebut menunjukan langkah-langkah penyelesaian Algoritma *Dynamic Programming* adalah sebagai berikut:

1. Memasukan banyaknya jenis barang yang akan dimasukkan (n), memasukan nilai kapasitas *knapsack* 0-1 (M), memasukan berat tiap barang (w), memasukan keuntungan tiap barang (r), memasukan keuntungan awal (y(1) = 0), memasukan nilai p = 1, dan memasukan beban awal = 0.

2. Apabila berat pada barang tersebut lebih kecil dari kapasitas yang tersedia, maka keuntungan barang tersebut dijumlah dengan keuntungan barang-barang dari sisa *knapsack* 0-1 saat ini. Namun, apabila berat barang lebih besar dari kapasitas yang tersedia, maka keuntungan saat ini menggunakan keuntungan sebelumnya. Pada saat mencari keuntungan maksimal menggunakan pendekatan maju (*forward*).

3. Setelah mendapatkan keuntungan maksimal maka barang yang dipilih untuk diangkut diberi nilai 1 dan yang tidak dipilih diberi nilai 0. Hal ini dilakukan dengan menerapkan pendekatan algoritma secara mundur (*backward*).

4. Barang yang bernilai 1 nantinya akan dijumlah beratnya. Berat itulah yang akan diangkut oleh *knapsack* 0-1 tersebut untuk mendapatkan keuntungan yang maksimal.

Contoh 3.2.1. Hitunglah keuntungan maksimal yang diperoleh dari keenam jenis rasa permen tersebut yang berkapasitas sebesar 7 kardus dengan menggunakan algoritma *Dynamic Programming* dan jenis permen apa sajakah yang dipilih agar keuntungan menjadi maksimal ?

Tabel 3.1: Kualifikasi Keuntungan

Jenis Rasa	w_k (Lusin Kardus)	r_k (Ribu)
Apel	1	200
Mangga	2	600
Durian	3	450
Jeruk	2	100
Alpukat	1	150
Anggur	4	1.600

Penyelesaian:

Diketahui:

Banyaknya jenis permen $(n) = 6$

Kapasitas muat *knapsack* 0-1 $(M) = 7$

Berat barang ke-k $(w_k) = \{w_1, w_2, w_3, w_4, w_5, w_6\} = \{1, 2, 3, 2, 1, 4\}$

Keuntungan barang ke-k $(r_k) = \{r_1, r_2, r_3, r_4, r_5, r_6\} = \{200, 600, 450, 100, 150, 1.600\}$

Keuntungan maksimal saat tidak ada barang yang dimasukkan $(f_0(y))= 0$

Ditanya:

1. Berapa keuntungan yang maksimal yang diperoleh perusahaan tersebut?
2. Rasa permen mana sajakah yang mampu membuat perusahaan mendapatkan keuntungan yang maksimal?

Jawab:

Proses pencarian keuntungan maksimal melalui rekursif maju dapat dilakukan sebagai berikut:

Tahap 1 :

$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$

$f_1(y) = max\{f_{1-1}(y), r_1 + f_{1-1}(y - w_1)\}$

$f_1(y) = max\{f_0(y), r_1 + f_0(y - w_1)\}$

Tabel 3.2: Perhitungan *forward* tahap 1

y	$f_0(y)$	$200 + f_0(y-1)$	$f_1(y)$
0	0	$-\infty$	0
1	0	$200 + 0 = 200$	200
2	0	$200 + 0 = 200$	200
3	0	$200 + 0 = 200$	200
4	0	$200 + 0 = 200$	200
5	0	$200 + 0 = 200$	200
6	0	$200 + 0 = 200$	200
7	0	$200 + 0 = 200$	200

Tahap 2 :

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$$

$$f_2(y) = max\{f_{2-1}(y), r_2 + f_{2-1}(y - w_2)\}$$

$$f_2(y) = max\{f_1(y), r_2 + f_1(y - w_2)\}$$

Tabel 3.3: Perhitungan *forward* tahap 2

y	$f_1(y)$	$600 + f_1(y-2)$	$f_2(y)$
0	0	$600 + (-\infty) = -\infty$	0
1	200	$600 + (-\infty) = -\infty$	200
2	200	$600 + 0 = 600$	600
3	200	$600 + 200 = 800$	800
4	200	$600 + 200 = 800$	800
5	200	$600 + 200 = 800$	800
6	200	$600 + 200 = 800$	800
7	200	$600 + 200 = 800$	800

Tahap 3 :

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$$

$$f_3(y) = max\{f_{3-1}(y), r_3 + f_{3-1}(y - w_3)\}$$

$$f_3(y) = max\{f_2(y), r_3 + f_2(y - w_3)\}$$

Tabel 3.4: Perhitungan *forward* tahap 3

y	$f_2(y)$	$450 + f_2(y-3)$	$f_3(y)$
0	0	$450 + (-\infty) = -\infty$	0
1	200	$450 + (-\infty) = -\infty$	200
2	600	$450 + (-\infty) = -\infty$	600
3	800	$450 + 0 = 450$	800
4	800	$450 + 200 = 650$	800
5	800	$450 + 600 = 1.050$	1.050
6	800	$450 + 800 = 1.250$	1.250
7	800	$450 + 800 = 1.250$	1.250

Tahap 4 :

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$$

$$f_4(y) = max\{f_{4-1}(y), r_4 + f_{4-1}(y - w_4)\}$$

$$f_4(y) = max\{f_3(y), r_4 + f_3(y - w_4)\}$$

Tabel 3.5: Perhitungan *forward* tahap 4

y	$f_3(y)$	$100 + f_3(y-2)$	$f_4(y)$
0	0	$100 + (-\infty) = -\infty$	0
1	200	$100 + (-\infty) = -\infty$	200
2	600	$100 + 0 = 100$	600
3	800	$100 + 200 = 300$	800
4	800	$100 + 600 = 700$	800
5	1.050	$100 + 800 = 900$	1.050
6	1.250	$100 + 800 = 900$	1.250
7	1.250	$100 + 1.050 = 1.150$	1.250

Tahap 5 :

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$$

$$f_5(y) = max\{f_{5-1}(y), r_5 + f_{5-1}(y - w_5)\}$$

$$f_5(y) = max\{f_4(y), r_5 + f_4(y - w_5)\}$$

Tabel 3.6: Perhitungan *forward* tahap 5

y	$f_4(y)$	$150 + f_4(y-1)$	$f_5(y)$
0	0	$150 + (-\infty) = -\infty$	0
1	200	$150 + 0 = 150$	200
2	600	$150 + 200 = 350$	600
3	800	$150 + 600 = 750$	800
4	800	$150 + 800 = 950$	950
5	1.050	$150 + 800 = 950$	1.050
6	1.250	$150 + 1.050 = 1.200$	1.250
7	1.250	$150 + 1.250 = 1.400$	1.400

Tahap 6 :

$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}$

$f_6(y) = max\{f_{6-1}(y), r_6 + f_{6-1}(y - w_6)\}$

$f_6(y) = max\{f_5(y), r_6 + f_5(y - w_6)\}$

Tabel 3.7: Perhitungan *forward* tahap 6

y	$f_5(y)$	$1.600 + f_5(y-4)$	$f_6(y)$
0	0	$1.600 + (-\infty) = -\infty$	0
1	200	$1.600 + (-\infty) = -\infty$	200
2	600	$1.600 + (-\infty) = -\infty$	600
3	800	$1.600 + (-\infty) = -\infty$	800
4	950	$1.600 + 0 = 1.600$	1.600
5	1.050	$1.600 + 200 = 1.800$	1.800
6	1.250	$1.600 + 600 = 2.200$	2.200
7	1.400	$1.600 + 800 = 2.400$	2.400

Dilihat dari hasil akhir perhitungan tersebut diperoleh keuntungan sebesar 2400 atau Rp2.400.000,00. Selanjutnya, keuntungan maksimal tersebut digunakan untuk pencarian jenis barang yang dipilih agar keuntungan tersebut menjadi maksimal. Maka, penulis menggunakan persamaan rekursif mundur sebagai berikut:

Banyaknya jenis permen (n)=6, $k = 6, 5, 4, ..., 1$,

Kapasitas muat *knapsack* 0-1 $(M) = 7$.

Tahap 1:

$k = 6$

$y = 7$

Tabel 3.8: Perhitungan *backward* tahap 1

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3

Pada tahap 1, diperoleh nilai $x_6 = 1$, artinya barang ke-6 dipilih pada *knapsack* 0-1 dan diperoleh juga nilai baru $M = 3$, di mana nilai M tersebut akan digunakan pada tahap 2 sebagai nilai y.

Tahap 2:

$k = 5$

$y = 3$

Tabel 3.9: Perhitungan *backward* tahap 2

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3
5	3	0	800	800	3	3

Pada tahap 2, diperoleh nilai $x_5 = 0$, artinya barang ke-5 tidak dipilih pada *knapsack* 0-1 dan nilai M tetap sama untuk tahap ke-3 yaitu $M = 3$.

Tahap 3:

$k = 4$

$y = 3$

Tabel 3.10: Perhitungan *backward* tahap 3

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3
5	3	0	800	800	3	3
4	3	0	800	800	3	3

Pada tahap 3, diperoleh nilai $x_4 = 0$, artinya barang ke-4 tidak dipilih pada *knapsack* 0-1 dan nilai M tetap sama untuk tahap ke-4 yaitu $M = 3$.

Tahap 4:

$k = 3$

$y = 3$

Tabel 3.11: Perhitungan *backward* tahap 4

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3
5	3	0	800	800	3	3
4	3	0	800	800	3	3
3	3	0	800	800	3	3

Pada tahap 4, diperoleh nilai $x_3 = 0$, artinya barang ke-3 tidak dipilih pada *knapsack* 0-1 dan nilai M tetap sama untuk tahap ke-5 yaitu $M = 3$.

Tahap 5:

$k = 2$

$y = 3$

Tabel 3.12: Perhitungan *backward* tahap 5

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3
5	3	0	800	800	3	3
4	3	0	800	800	3	3
3	3	0	800	800	3	3
2	3	1	800	$800 - 600 = 200$	$3 - 2 = 1$	1

Pada tahap 5, diperoleh nilai $x_2 = 1$, artinya barang ke-2 dipilih pada *knapsack* 0-1 dan diperoleh juga nilai baru $M = 1$, di mana nilai M tersebut akan digunakan pada tahap 6 sebagai nilai y.

Tahap 6:

$k = 1$

$y = 1$

Tabel 3.13: Perhitungan *backward* tahap 6

k	y	x_k	u_k	s_k	q_k	M
6	7	1	2.400	$2.400 - 1.600 = 800$	$7 - 4 = 3$	3
5	3	0	800	800	3	3
4	3	0	800	800	3	3
3	3	0	800	800	3	3
2	3	1	800	$800 - 600 = 200$	$3 - 2 = 1$	1
1	1	1	200	$200 - 200 = 0$	$1 - 1 = 0$	0

Pada tahap 6, diperoleh nilai $x_1 = 1$, artinya barang ke-1 dipilih pada *knapsack* 0-1 dan diperoleh hasil akhir $M = 0$, artinya tidak ada kapasitas muat *knapsack* 0-1 yang tersedia atau *knapsack* 0-1 terisi penuh tanpa ada sisa muat *knapsack* 0-1.

Dari pendekatan rekursif mundur tersebut dapat ditentukan macam barang yang dipilih untuk dimasukkan ke dalam *knapsack* 0-1 sehingga menghasilkan keuntungan yang optimal. Berdasarkan contoh soal, maka jenis barang yang

dipilih yaitu jenis barang ke-1, ke-2, dan ke-6. Dengan demikian, total berat barang yang diangkut oleh *knapsack* 0-1 tersebut dapat dihitung dengan persamaan sebagai berikut:

$W = \Sigma_{k=1}^{n} w_k$

$W = \Sigma_{k=1}^{6} w_k$

$W = w_1 + w_2 + w_3 + w_4 + w_5 + w_6$

$W = 1 + 2 + 0 + 0 + 0 + 4$

$W = 7$

Dari perhitungan di atas dengan $n = 6$ di mana n merupakan jenis permen dapat disimpulkan bahwa pengangkutan yang dilakukan *knapsack* 0-1 tersebut memperoleh keuntungan maksimal sebesar Rp2.400.000,00. Keuntungan tersebut diperoleh dari pengangkutan tiga macam jenis barang yaitu jenis barang ke-1, ke-2, dan ke-6, barang-barang tersebut berupa jenis permen rasa apel, mangga, dan anggur dengan total berat sebesar 7 kardus. Adapun hasil perhitungan matlabnya adalah sebagai berikut (data terlampir):

3.3 Studi Kasus

Pada studi kasus ini, data yang digunakan adalah data primer sejumlah barang yang akan dibeli oleh Usaha Dagang milik Ikhsan di mana Usaha Dagang tersebut bergerak pada bidang perdagangan jenis kebutuhan makanan dan minuman ringan yang terletak di Jalan Pekapuran, Kecamatan Tapos, Kota Depok. Data yang diperoleh merupakan data mentah berat barang, harga beli tiap karton barang, dan harga jual tiap item barang. Pada kasus ini, data yang diambil hanya berupa data minuman ringan sebanyak 72 jenis minuman yang akan dibeli oleh Usaha Dagang tersebut. Untuk menerapkan data ini ke dalam permasalahan *knapsack* 0-1, penulis perlu melakukan pengidenti-

fikasian dari data mentah tersebut yaitu dengan menggunakan data harga beli tiap karton, harga jual tiap item barang, dan berat barang serta dengan mencari keuntungan pada masing-masing barang. Dari data yang telah diambil oleh penulis, berikut ini merupakan data yang akan diolah untuk diterapkan ke dalam permasalahan *knapsack* 0-1 dengan menggunakan Algoritma *Dynamic Programming* yang memiliki kapasitas daya angkut sebesar 1.800 kg (data terlampir):

3.4 Algoritma *Dynamic Programming* dalam Permasalahan *Knapsack* 0-1 untuk Memperoleh Keuntungan Maksimal

Pada permasalahan daya angkut tersebut agar memperoleh keuntungan yang maksimal, penulis menggunakan *knapsack* 0-1 (Integer *Knapsack*). Pada penggunaan *Knapsack* 0-1 ini terdapat dua keputusan pemilihan barang, yaitu barang diangkut semua (bernilai 1) atau tidak diangkut semua (bernilai 0). Sedangkan, Algoritma *Dynamic Programming* digunakan pada permasalahan tersebut karena dalam mencari keuntungan yang maksimal dengan daya tampung yang terbatas dengan cara menguraikan sekumpulan langkah atau tahapan. Hal ini dilakukan dengan mencari terlebih dahulu keuntungan yang diperoleh dari sekelompok barang dengan kemungkinan kapasitasnya dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada permasalahan ini, pendekatan Algoritma *Dynamic Programming* yang digunakan yaitu pendekatan maju (*forward*) dan pendekatan mundur (*backward*). Di mana pendekatan maju *forward* digunakan untuk perhitungan men-

cari keuntungan maksimal pada pengangkutan barang tersebut, di mana dalam perhitungannya dimulai dari tahap awal sampai tahap akhir atau dimulai saat *knapsack* 0-1 dalam keadaan kosong hingga *knapsack* 0-1 terisi maksimal dan tidak boleh melebihi batas *knapsack* 0-1 yang telah tersedia. Persamaan yang digunakan pada pendekatan maju dapat dilihat pada persamaan (3.5), di mana nilai $f_k(y)$ tergantung dengan nilai k pada saat kapasitas sebesar y.

Sedangkan pendekatan mundur *backward* digunakan ketika telah memperoleh keuntungan yang maksimal, yaitu dimulai dari akhir hingga awal yang bertujuan untuk mencari jenis barang apa saja yang dipilih untuk diangkut dan menentukan juga berat optimal barang tersebut. Persamaan pendekatan mundur dapat dilihat pada persamaan (3.6) dan (3.11). Di mana, nilai M akan merubah nilai y pada tahap selanjutnya ketika nilai $x_k = 1$ atau nilai M tidak berubah untuk tahap selanjutnya sehingga tidak merubah nilai y, hal tersebut terjadi jika nilai $x_k = 0$. Dengan demikian, pada persamaan (3.16) dapat ditentukan total berat maksimal yang akan diangkut oleh *knapsack* 0-1 tersebut.

3.5 Hasil Perhitungan

Pada kasus di atas, dengan menggunakan bantuan matlab maka dapat dilihat bahwa dari 72 jenis minuman yang akan dibeli untuk dijual kembali dengan kapasitas sebesar 1.800 kg diperoleh keuntungan maksimal sebesar Rp9.243.700,00. Keuntungan tersebut berasal dari tiap item barang yang diambil. Pada kasus ini, barang-barang yang diambil adalah Buavita 250 ml, Coca-Cola Mini 250 ml, Fanta Mini 250 ml, Sprite Mini 250 ml, Floridina Orange 360 ml, Love Juice 300 ml, Mytea 500 ml, Nu Green Tea 500 ml, Milo Actigen 18 gr, Teh Pucuk Harum/Less Sugar 350 ml, Susu Bendera UHT 70

ml, Susu Bendera UHT 115 ml, Susu Ultra 250 ml, Teh Botol Sosro 350 ml, You Cee 1000 Orange 140 ml, You Cee 1000 Lemon 140 ml, Fruit Tea Blackcurrant 500 ml, Fruit Tea Apple 500 ml, Mizone Lychee Lemon 500 ml, Mizon Apple Guava 500 ml, Good Day Tiramisu 250 ml, Good Day Mocacino 250 ml, Larutan Penyegar Cap Badak Strawberry 320 ml, Larutan Penyegar Cap Badak Jambu Can 320 ml, Larutan Penyegar Cap Badak Lychee Can 320 ml, Coolant 350 ml, Adem Sari Ching Ku 350 ml, Vit Levite 350 ml, Kratingdaeng 150 ml, Kopiko Botol 78 C 240 ml, Pulpy Orange 350 ml, dan Q Guava 350 ml. Dari barang-barang tersebut, maka *knapsack* 0-1 akan mengangkut beban sebesar 1.797 kg. Berat tersebut merupakan berat maksimal yang akan diangkut yang memperoleh keutungan yang maksimal tanpa melewati kapasitas yang tersedia.

BAB IV

PENUTUP

4.1 Kesimpulan

Pada bab sebelumnya telah dijelaskan oleh penulis mengenai setiap rumusan masalah dalam skripsi ini. Berdasarkan pembahasan tersebut maka dapat disimpulkan bahwa:

1. Algoritma *Dynamic Programming* menyelesaikan suatu permasalahan dengan cara membagi permasalahan tersebut ke dalam subpermasalahan atau tahapan-tahapan yang lebih kecil dan berurutan. Berdasarkan tahapan-tahapan tersebut nantinya akan diperoleh suatu status, di mana status terakhir setiap tahap akan menjadi *input* untuk tahap selanjutnya atau dapat dikatakan sebagai kapasitas muat *knapsack* 0-1 yang tersisa setelah memasukan barang pada tahap sebelumnya. Pada penyelesaian kasus ini, Algoritma *Dynamic Programming* menggunakan dua pendekatan rekursif yaitu pendekatan maju untuk mencari keuntungan maksimal dan pendekatan mundur untuk mencari macam barang yang dipilih dan total berat maksimal yang dapat diangkut oleh *knapsack* 0-1 tersebut. *Knapsack* 0-1 mempunyai dua keputusan pada saat pemilihan barang, yaitu jika barang tersebut dipilih maka bernilai 1 dan jika barang tersebut tidak dipilih maka bernilai 0. Barang yang bernilai 1 akan ditotal beratnya, sehingga menghasilkan berat maksimal yang dapat diangkut oleh *knapsack* 0-1 tersebut.

2. Model optimasi yang diperoleh permasalahan *knapsack* 0-1 ini adalah

sebagai berikut:

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}; k = 1, 2, 3, ..., n$$

Dari studi kasus yang diberikan dengan menggunakan 72 jenis minuman dan mempunyai kapasitas sebesar 1.800 kg, maka model optimasi pada kasus tersebut adalah:

$$f_k(y) = max\{f_{k-1}(y), r_k + f_{k-1}(y - w_k)\}; \text{ dengan,}$$
$$M = 1.800, k = 1, 2, 3, ..., 72, dany = 0, 1, 2, 3, ..., 1.800$$

sehingga total berat maksimal dapat dihitung dengan,

$$W = \Sigma_{k=1}^{72} w_k$$

3. Pada data studi kasus yang digunakan penulis, yaitu jenis minuman yang akan dibeli oleh Usaha Dagang milik Ikhsan yang terdiri dari 72 jenis minuman dengan daya tampung *knapsack* 0-1 sebesar 1.800 kg diperoleh keuntungan sebesar Rp9.243.700,00 dengan mengangkut beban sebesar 1.797 Kg. Berat tersebut merupakan berat maksimal yang dapat diangkut oleh *Knapsack* 0-1 tersebut agar mendapatkan keuntungan yang maksimal. Pada kasus seperti ini dengan data yang tidak terlalu banyak, penggunaan Algoritma *Dynamic Programming* sangat cocok untuk digunakan, karena tidak membutuhkan waktu yang lama untuk memperoleh hasilnya. Dengan menggunakan bantuan matlab, pada kasus ini Algoritma *Dynamic Programming* membutuhkan waktu selama 1,3 detik untuk memperoleh hasilnya.

4.2 Saran

Adapun saran dari penulis untuk penelitian selanjutnya adalah sebagai berikut:

1. Agar penelitian selanjutnya dapat menggunakan data yang lebih banyak dibandingkan data yang sudah diteliti sebelumnya.

2. Agar penelitian selanjutnya dapat menggunakan metode *knapsack* yang berbeda seperti *knapsack* terbatas (*bounded*) dan *knapsack* tak terbatas (*unbounded*).

3. Agar penelitian selanjutnya dapat menggunakan Algoritma lain selain dari Algoritma *Dynamic Programming* dalam penggunaan *knapsack* 0-1 dan dapat membandingkan hasilnya.

DAFTAR PUSTAKA

A. Taha, Handy. 1996. *Riset Operasi*. Jakarta : Binarupa Aksara.

Ahmed, Zaheed, Irfan Younas. February 2011. *A Dynamic Programming based GA for 0-1 Modified Knapsack Problem*. International Journal of Coumputer Applications (0975-8887). Volume 16, No. 7, http://www.ijcaonline.org/volume16/number7/pxc3872668.pdf, 2 Febuari 2016.

Fitrianto, Iwan, Djoko Soetarno. 2011. *Menentukan Lintasan Terpendek (Shortest Path) dengan 0/1 Knapsack Problem dan Pendekatan Algoritma Dynamic Programming*. Vol. 4, No. 3, *http* : $//raharja.ac.id/raharja_file/file_jurnal/file//4030611.pdf$, 3 Febuari 2016.

Hadi, Ivan Syakhul. 2015. *Penerapan Algoritma Genetika Hybrid pada Permasalahan Bounded Knapsack* . http://repository.unej.ac.id/handle/123456789/72723. [17 Maret 2016, 15:06]

Kosasi, Sandi. Juli 2013. *Penyelesaian Bounded Knapsack Problem Menggunakan Dynamic Programming (Studi Kasus: CV. Mulia Abadi)*. Jurnal Informatika Mulawarman. Vol. 8, No. 2, http://e-journals.unmul.ac.id/index.php/JIM/article/view/107/pdf, 2 Febuari 2016.

Martello, S. 2006. *New Trends in Exact Algorithms for the 0-1 Knapsack Problem.*[ONLINE] tersedia : http://www.cise.ufl.edu/ sahni/papers/knao.pdf. [20 Maret 2016, 17:33]

MT, Suryadi. 1996. *Pengantar Analisis Algoritma*. Jakarta : Gunadarma.

Munir M.T. Rinaldi. 2005. *Diktat Kuliah IF225I Strategi Algoritmik*. Bandung: Institusi Teknologi Bandung.

Pisinger, David. 1995. *Algorithms for Knapsack Problems*. Denmark: Dept. of Computer Science University of Copenhagen.

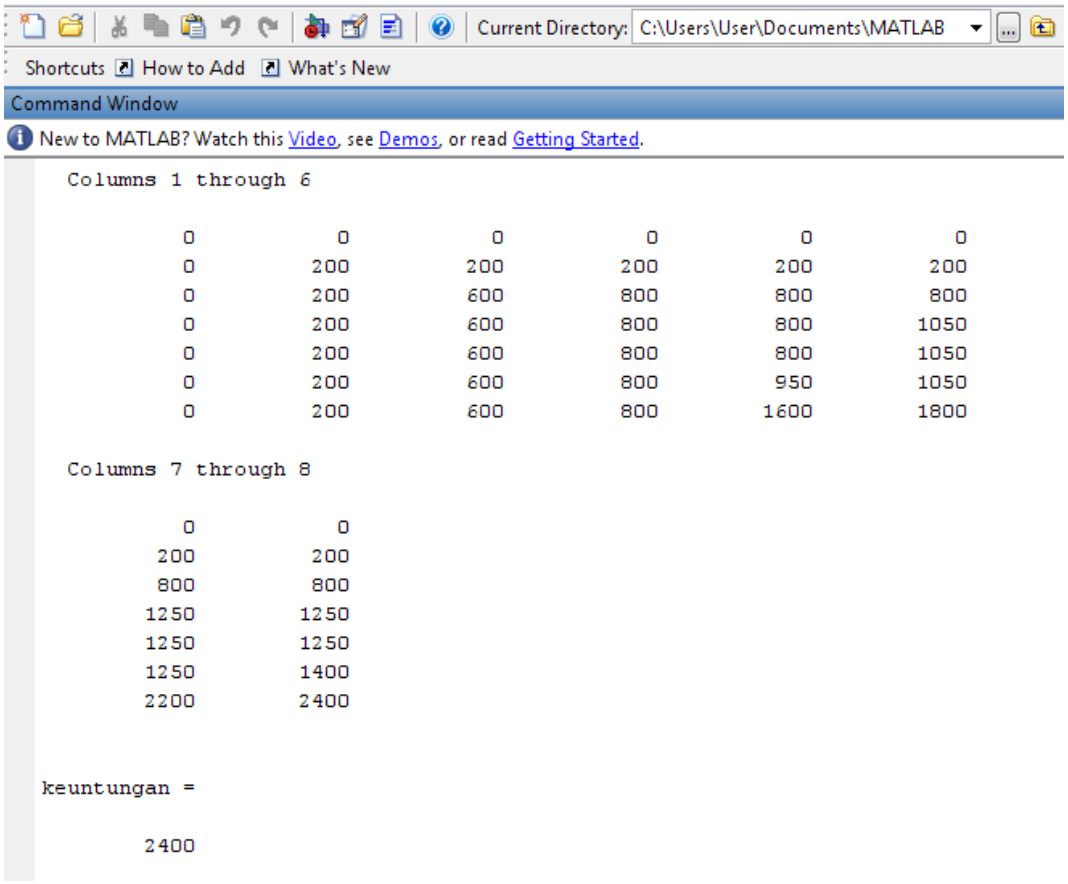
S. Hiller, Frederick,. Gerald J. Lieberman. 1994. *Pengantar Riset Operasi*. Jakarta: PT. Gelora Aksara Pratama.

LAMPIRAN-LAMPIRAN

Script* Contoh 3.2.1 Algoritma *Dynamic Programming

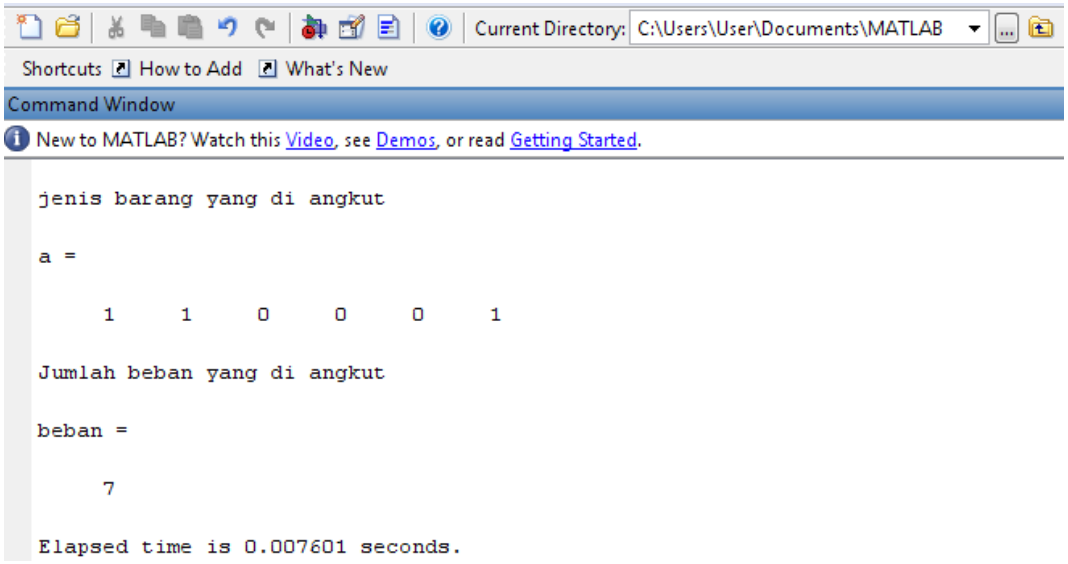
```
tic;
n=6;
m=7;
M=7;
r=[200 600 450 100 150 1600]
w=[1 2 3 2 1 4]
p=1
z=zeros(n+1,m+1);
beban=0;
%tahap 1 sampai n+1
for i=2:n+1
    y(1)=0;
    z(i,1)=0;
    for j=2:m+1
        y(j)=y(j-1)+p;
        if w(i-1)<=y(j)
            z(i,j)=max([z(i-1,j)  r(i-1)+z(i-1,j-w(i-1))]);
        else
            z(i,j)=z(i-1,j);
        end
    end
end
disp(z);
%mencari solusi optimum
keuntungan = (z(i,m+1))
%Mencari Barang yang dibeli
for i=n+1:-1:2
    if z(i,m+1)~=z(i-1,m+1)
        a(i)=1;
        M=m;
        q=M-w(i-1);
        m=round(q);
    else
        a(i)=0;
    end
end
disp('jenis barang yang di angkut')
a=a(2:n+1)
for i=1:n
    if a(i)==1
        beban=beban+w(i);
    end
end
disp('Jumlah beban yang di angkut')
beban
toc;
```

Tampilan *Output* Keuntungan Maksimal yang Diperoleh dari Contoh 3.2.1



Gambar 4.1: Keuntungan Maksimal dari Contoh 3.2.1

Tampilan *Output* Jenis Barang yang Diangkut dan Beban Maksimal yang Diperoleh dari Contoh 3.2.1



Gambar 4.2: Jenis Barang yang Diangkut dan Beban Maksimal dari Contoh 3.2.1

DAFTAR HARGA MINUMAN RINGAN

No.	Nama Barang	Berat/Barang (kg)	Banyak Barang	Total Berat Barang (kg)	Harga Beli	Harga Jual/barang	Total Harga Jual/Karton	Keuntungan
1	Kopi Kapal Api Spesial 165 gr	0.165	1 karton = 20 bungkus	3.3	Rp201,500.00	Rp10,200.00	Rp204,000.00	Rp2,500.00
2	Kopi Kapal Api Spesial 380 gr	0.38	1 karton = 9 bungkus	3.42	Rp160,000.00	Rp18,000.00	Rp162,000.00	Rp2,000.00
3	Kopi Kapal Api Spesial Mix 25 gr	0.025	1 karton = 12 pak	3	Rp100,000.00	Rp8,500.00	Rp102,000.00	Rp2,000.00
4	Aqua 240 ml/gelas	0.24	1 karton = 48 gelas	11.52	Rp23,000.00	Rp500.00	Rp24,000.00	Rp1,000.00
5	Aqua 330 ml/mini	0.33	1 karton = 24 botol	7.92	Rp32,000.00	Rp1,500.00	Rp36,000.00	Rp4,000.00
6	Aqua 600 ml/tanggung	0.6	1 karton = 24 botol	14.4	Rp40,000.00	Rp3,000.00	Rp72,000.00	Rp32,000.00
7	Aqua 1500 ml/besar	1.5	1 karton = 12 botol	18	Rp40,000.00	Rp5,600.00	Rp67,200.00	Rp27,200.00
8	Big Cola 535 ml	0.535	1 karton = 12 botol	6.42	Rp29,500.00	Rp3,500.00	Rp42,000.00	Rp12,500.00
9	Big Cola 3,1 L	3.1	1 botol	3.1	Rp13,000.00	Rp15,000.00	Rp15,000.00	Rp2,000.00
10	Buavita 250 ml	0.25	1 karton = 24 dus	6	Rp128,500.00	Rp6,900.00	Rp165,600.00	Rp37,100.00
11	coca-cola mini 250 ml	0.25	1 karton = 12 botol	3	Rp31,000.00	Rp3,800.00	Rp45,600.00	Rp14,600.00
12	fanta mini 250 ml	0.25	1 karton = 12 botol	3	Rp31,000.00	Rp3,800.00	Rp45,600.00	Rp14,600.00
13	sprite mini 250 ml	0.25	1 karton = 12 botol	3	Rp31,000.00	Rp3,800.00	Rp45,600.00	Rp14,600.00
14	fanta seru 425 ml	0.425	1 karton = 12 botol	5.1	Rp45,000.00	Rp4,500.00	Rp54,000.00	Rp9,000.00
15	coca-cola seru 425 ml	0.425	1 karton = 12 botol	5.1	Rp45,000.00	Rp4,500.00	Rp54,000.00	Rp9,000.00

16	Sprite seru 425 ml	0.425	1 karton = 12 botol	5.1	Rp45,000.00	Rp4,500.00	Rp54,000.00	Rp9,000.00
17	Floridina Orange 360 ml	0.36	1 karton = 12 botol	4.32	Rp29,500.00	Rp7,890.00	Rp94,680.00	Rp65,180.00
18	Love Juice 300 ml	0.3	1 karton = 24 botol	7.2	Rp85,000.00	Rp6,500.00	Rp156,000.00	Rp71,000.00
19	Mytea 500 ml	0.5	1 karton = 24 botol	12	Rp101,700.00	Rp5,500.00	Rp132,000.00	Rp30,300.00
20	Nescafe UHT 200 ml	0.2	1 karton = 36 dus	7.2	Rp119,500.00	Rp3,500.00	Rp126,000.00	Rp6,500.00
21	Nu Green Tea 500 ml	0.5	1 karton = 24 botol	12	Rp98,000.00	Rp6,500.00	Rp156,000.00	Rp58,000.00
22	Pocari Sweat Botol 350 ml	0.35	1 karton = 24 botol	8.4	Rp144,000.00	Rp6,250.00	Rp150,000.00	Rp6,000.00
23	Pocari Sweat Botol 500 ml	0.5	1 karton = 24 botol	12	Rp168,000.00	Rp7,500.00	Rp180,000.00	Rp12,000.00
24	Pocari Sweat Botol 900 ml	0.9	1 karton = 15 botol	13.5	Rp180,000.00	Rp12,500.00	Rp187,500.00	Rp7,500.00
25	Pocari Sweat Botol 2000 ml	2	1 karton = 6 botol	12	Rp120,000.00	Rp20,500.00	Rp123,000.00	Rp3,000.00
26	Pocari Sweat Kaleng 330 ml	0.33	1 karton = 24 kaleng	7.92	Rp144,000.00	Rp6,500.00	Rp156,000.00	Rp12,000.00
27	Milo Actigen 18 gr	0.018	1 karton = 20 renteng	0.36	Rp248,000.00	Rp12,600.00	Rp252,000.00	Rp4,000.00
28	Teh Kotak Ultrajaya 300 ml	0.3	1 karton = 24 dus	7.2	Rp60,000.00	Rp3,000.00	Rp72,000.00	Rp12,000.00
29	Teh Pucuk Harum/Less Sugar 350 ml	0.35	1 karton = 24 botol	8.4	Rp49,000.00	Rp3,000.00	Rp72,000.00	Rp23,000.00
30	Vit 600 ml	0.6	1 karton = 24 botol	14.4	Rp27,000.00	Rp1,500.00	Rp36,000.00	Rp9,000.00
31	Milo UHT 115 ml	0.115	1 karton = 36 biji	4.14	Rp78,000.00	Rp2,300.00	Rp82,800.00	Rp4,800.00
32	Milo UHT 200 ml	0.2	1 karton = 36 biji	7.2	Rp123,500.00	Rp3,600.00	Rp129,600.00	Rp6,100.00
33	Real Good Bantal 90 ml	0.09	1 karton = 48 biji	4.32	Rp41,000.00	Rp1,000.00	Rp48,000.00	Rp7,000.00

34	Susu Bendera Botol 180 ml	0.18	1 karton = 24 botol	4.32	Rp76,700.00	Rp3,300.00	Rp79,200.00	Rp2,500.00
35	Susu Bendera UHT 70 ml	0.07	1 karton = 36 dus	2.52	Rp48,000.00	Rp1,500.00	Rp54,000.00	Rp6,000.00
36	Susu Bendera UHT 115 ml	0.115	1 karton = 36 kotak	4.14	Rp74,000.00	Rp2,300.00	Rp82,800.00	Rp8,800.00
37	Susu Bendera UHT 180 ml	0.18	1 karton = 36 dus	6.48	Rp115,000.00	Rp3,300.00	Rp118,800.00	Rp3,800.00
38	Susu Bendera UHT 250 ml	0.25	1 karton = 36 dus	9	Rp152,000.00	Rp4,500.00	Rp162,000.00	Rp10,000.00
39	Susu Ultra 125 ml	0.125	1 karton = 40 dus	5	Rp85,600.00	Rp2,300.00	Rp92,000.00	Rp6,400.00
40	Susu Ultra 200 ml	0.2	1 karton = 24 dus	4.8	Rp81,200.00	Rp3,500.00	Rp84,000.00	Rp2,800.00
41	Susu Ultra 250 ml	0.25	1 karton = 24 dus	6	Rp101,000.00	Rp5,500.00	Rp132,000.00	Rp31,000.00
42	Susu Ultra 1000 ml Full Cream	1	1 karton = 12 dus	12	Rp172,000.00	Rp14,600.00	Rp175,200.00	Rp3,200.00
43	Larutan Badak Botol 330 ml	0.33	1 karton = 8 banded	2.64	Rp103,000.00	Rp13,000.00	Rp104,000.00	Rp1,000.00
44	Teh Zegar 2 Tang 185 ml	0.185	1 karton = 24 gelas	4.44	Rp19,000.00	Rp1,000.00	Rp24,000.00	Rp5,000.00
45	Teh Botol Sosro 350 ml	0.35	1 karton = 12 botol	4.2	Rp39,000.00	Rp4,900.00	Rp58,800.00	Rp19,800.00
46	You Cee 1000 Orange 140 ml	0.14	1 karton = 6 botol	0.84	Rp27,000.00	Rp6,500.00	Rp39,000.00	Rp12,000.00
47	You Cee 1000 Lemon 140 ml	0.14	1 karton = 6 botol	0.84	Rp27,000.00	Rp6,500.00	Rp39,000.00	Rp12,000.00
48	Fruit Tea Blackcurrant 500 ml	0.5	1 karton = 24 botol	12	Rp57,000.00	Rp7,500.00	Rp180,000.00	Rp123,000.00
49	Fruit Tea Apple 500 ml	0.5	1 karton = 24 botol	12	Rp57,000.00	Rp7,000.00	Rp168,000.00	Rp111,000.00
50	Mizone Lychee Lemon 500 ml	0.5	1 karton = 12 botol	6	Rp34,000.00	Rp4,400.00	Rp52,800.00	Rp18,800.00
51	Mizone Apple Guava 500 ml	0.5	1 karton = 12 botol	6	Rp34,000.00	Rp4,400.00	Rp52,800.00	Rp18,800.00

52	Good Day Tiramisu 250 ml	0.25	1 karton = 24 botol	6	Rp142,000.00	Rp7,000.00	Rp168,000.00	Rp26,000.00
53	Good Day Mocacino 250 ml	0.25	1 karton = 24 botol	6	Rp142,000.00	Rp7,000.00	Rp168,000.00	Rp26,000.00
54	Teh Rio 300 ml	0.3	1 karton = 24 gelas	7.2	Rp18,500.00	Rp1,000.00	Rp24,000.00	Rp5,500.00
55	Larutan Penyegar Cap Badak Strawberry 320 ml	0.32	1 karton = 24 kaleng	7.68	Rp105,000.00	Rp6,000.00	Rp144,000.00	Rp39,000.00
56	Larutan Penyegar Cap Badak Jambu Can 320 ml	0.32	1 karton = 24 kaleng	7.68	Rp105,000.00	Rp6,000.00	Rp144,000.00	Rp39,000.00
57	Larutan Penyegar Cap Badak Lychee Can 320 ml	0.32	1 karton = 24 kaleng	7.68	Rp105,000.00	Rp6,000.00	Rp144,000.00	Rp39,000.00
58	Ale-Ale Sirsak 200 ml	0.2	1 karton = 24 gelas	4.8	Rp18,500.00	Rp1,000.00	Rp24,000.00	Rp5,500.00
59	Ultra Sari Kacang Ijo 200 ml	0.2	1 karton = 24 kerdus	4.8	Rp80,000.00	Rp3,700.00	Rp88,800.00	Rp8,800.00
60	Mirai Ocha Pet Bottle 450 ml	0.45	1 karton = 24 botol	10.8	Rp96,000.00	Rp4,200.00	Rp100,800.00	Rp4,800.00
61	Mytea Original Bottle 450 ml	0.45	1 karton = 6 botol	2.7	Rp35,000.00	Rp6,500.00	Rp39,000.00	Rp4,000.00
62	Coolant 350 ml	0.35	1 karton = 24 botol	8.4	Rp90,500.00	Rp6,500.00	Rp156,000.00	Rp65,500.00
63	Adem Sari Ching Ku 350 ml	0.35	1 karton = 24 kaleng	8.4	Rp120,000.00	Rp6,400.00	Rp153,600.00	Rp33,600.00
64	Alto Mineral Water 600 ml	0.6	1 karton = 24 botol	14.4	Rp42,000.00	Rp2,000.00	Rp48,000.00	Rp6,000.00
65	Ades Mineral Water 600 ml	0.6	1 karton = 24 botol	14.4	Rp50,000.00	Rp3,400.00	Rp81,600.00	Rp31,600.00
66	Vit Levite 350 ml	0.35	1 karton = 12 botol	4.2	Rp38,000.00	Rp4,300.00	Rp51,600.00	Rp13,600.00
67	Kratingdaeng 150 ml	0.15	1 karton = 6 botol	0.9	Rp28,000.00	Rp6,500.00	Rp39,000.00	Rp11,000.00
68	Kopiko Botol 78 C 240 ml	0.24	1 Karton = 24 Botol	5.76	Rp117,500.00	Rp5,900.00	Rp141,600.00	Rp24,100.00
69	Pulpy Orange 350 ml	0.35	1 karton = 12 botol	4.2	Rp63,500.00	Rp6,500.00	Rp78,000.00	Rp14,500.00

70	Frestea Pet 350 ml	0.35	1 Karton = 8 Botol	2.8	Rp34,500.00	Rp4,700.00	Rp37,600.00	Rp3,100.00
71	Fatigon Hydro 250 ml	0.25	1 Karton = 24 buah	6	Rp85,000.00	Rp3,700.00	Rp88,800.00	Rp3,800.00
72	Q Guava 350 ml	0.35	1 karton = 12 botol	4.2	Rp41,000.00	Rp5,000.00	Rp60,000.00	Rp19,000.00

DATA PENELITIAN

No.	Nama Barang	Banyaknya Pembelian/Karton	Total Berat Pembelian	Harga Pembelian	Harga Penjualan	Keuntungan
1	Kopi Kapal Api Spesial 165 gr	30	99	Rp6,045,000.00	Rp6,120,000.00	Rp 75,000.00
2	Kopi Kapal Api Spesial 380 gr	25	86	Rp4,000,000.00	Rp4,050,000.00	Rp50,000.00
3	Kopi Kapal Api Spesial Mix 25 gr	40	120	Rp4,000,000.00	Rp4,080,000.00	Rp80,000.00
4	Aqua 240 ml/gelas	20	230	Rp460,000.00	Rp480,000.00	Rp20,000.00
5	Aqua 330 ml/mini	15	119	Rp480,000.00	Rp540,000.00	Rp60,000.00
6	Aqua 600 ml/tanggung	25	360	Rp1,000,000.00	Rp1,800,000.00	Rp800,000.00
7	Aqua 1500 ml/besar	10	180	Rp 400,000.00	Rp672,000.00	Rp272,000.00
8	Big Cola 535 ml	15	96	Rp442,500.00	Rp630,000.00	Rp187,500.00
9	Big Cola 3,1 L	10	31	Rp130,000.00	Rp150,000.00	Rp20,000.00
10	Buavita 250 ml	10	60	Rp1,285,000.00	Rp 1,656,000.00	Rp 371,000.00
11	coca-cola mini 250 ml	7	21	Rp217,000.00	Rp319,200.00	Rp102,200.00
12	fanta mini 250 ml	10	30	Rp310,000.00	Rp456,000.00	Rp146,000.00
13	sprite mini 250 ml	10	30	Rp310,000.00	Rp456,000.00	Rp146,000.00
14	fanta seru 425 ml	12	61	Rp540,000.00	Rp648,000.00	Rp108,000.00
15	coca-cola seru 425 ml	15	77	Rp675,000.00	Rp810,000.00	Rp135,000.00
16	Sprite seru 425 ml	10	51	Rp450,000.00	Rp540,000.00	Rp90,000.00
17	Floridina Orange 360 ml	10	43	Rp295,000.00	Rp946,800.00	Rp651,800.00
18	Love Juice 300 ml	10	72	Rp850,000.00	Rp1,560,000.00	Rp710,000.00
19	Mytea 500 ml	15	180	Rp1,525,500.00	Rp1,980,000.00	Rp454,500.00
20	Nescafe UHT 200 ml	10	72	Rp1,195,000.00	Rp1,260,000.00	Rp65,000.00
21	Nu Green Tea 500 ml	10	120	Rp980,000.00	Rp1,560,000.00	Rp580,000.00
22	Pocari Sweat Botol 350 ml	15	126	Rp2,160,000.00	Rp2,250,000.00	Rp90,000.00
23	Pocari Sweat Botol 500 ml	12	144	Rp2,016,000.00	Rp2,160,000.00	Rp144,000.00
24	Pocari Sweat Botol 900 ml	10	135	Rp1,800,000.00	Rp1,875,000.00	Rp75,000.00
25	Pocari Sweat Botol 2000 ml	8	96	Rp960,000.00	Rp984,000.00	Rp24,000.00
26	Pocari Sweat Kaleng 330 ml	7	55	Rp1,008,000.00	Rp1,092,000.00	Rp84,000.00
27	Milo Actigen 18 gr	10	4	Rp2,480,000.00	Rp2,520,000.00	Rp40,000.00
28	Teh Kotak Ultrajaya 300 ml	10	72	Rp600,000.00	Rp720,000.00	Rp120,000.00

29	Teh Pucuk Harum/Less Sugar 350 ml	8	67	Rp392,000.00	Rp576,000.00	Rp184,000.00
30	Vit 600 ml	10	144	Rp270,000.00	Rp360,000.00	Rp90,000.00
31	Milo UHT 115 ml	10	41	Rp780,000.00	Rp828,000.00	Rp48,000.00
32	Milo UHT 200 ml	10	72	Rp1,235,000.00	Rp1,296,000.00	Rp61,000.00
33	Real Good Bantal 90 ml	5	22	Rp205,000.00	Rp240,000.00	Rp35,000.00
34	Susu Bendera Botol 180 ml	10	43	Rp767,000.00	Rp792,000.00	Rp25,000.00
35	Susu Bendera UHT 70 ml	20	50	Rp960,000.00	Rp1,080,000.00	Rp120,000.00
36	Susu Bendera UHT 115 ml	25	104	Rp1,850,000.00	Rp2,070,000.00	Rp220,000.00
37	Susu Bendera UHT 180 ml	5	32	Rp575,000.00	Rp594,000.00	Rp19,000.00
38	Susu Bendera UHT 250 ml	8	72	Rp1,216,000.00	Rp1,296,000.00	Rp80,000.00
39	Susu Ultra 125 ml	5	25	Rp428,000.00	Rp460,000.00	Rp32,000.00
40	Susu Ultra 200 ml	15	72	Rp1,218,000.00	Rp1,260,000.00	Rp42,000.00
41	Susu Ultra 250 ml	20	120	Rp2,020,000.00	Rp2,640,000.00	Rp620,000.00
42	Susu Ultra 1000 ml Full Cream	5	60	Rp860,000.00	Rp876,000.00	Rp16,000.00
43	Larutan Badak Botol 330 ml	25	66	Rp2,575,000.00	Rp2,600,000.00	Rp25,000.00
44	Teh Zegar 2 Tang 185 ml	25	111	Rp475,000.00	Rp600,000.00	Rp125,000.00
45	Teh Botol Sosro 350 ml	5	21	Rp195,000.00	Rp294,000.00	Rp99,000.00
46	You Cee 1000 Orange 140 ml	10	8	Rp270,000.00	Rp390,000.00	Rp120,000.00
47	You Cee 1000 Lemon 140 ml	10	8	Rp270,000.00	Rp390,000.00	Rp120,000.00
48	Fruit Tea Blackcurrant 500 ml	5	60	Rp285,000.00	Rp900,000.00	Rp615,000.00
49	Fruit Tea Apple 500 ml	5	60	Rp285,000.00	Rp840,000.00	Rp555,000.00
50	Mizone Lychee Lemon 500 ml	6	36	Rp204,000.00	Rp316,800.00	Rp112,800.00
51	Mizone Apple Guava 500 ml	5	30	Rp170,000.00	Rp264,000.00	Rp94,000.00
52	Good Day Tiramisu 250 ml	25	150	Rp3,550,000.00	Rp4,200,000.00	Rp650,000.00
53	Good Day Mocacino 250 ml	22	132	Rp3,124,000.00	Rp3,696,000.00	Rp572,000.00
54	Teh Rio 300 ml	20	144	Rp370,000.00	Rp480,000.00	Rp110,000.00
55	Larutan Penyegar Cap Badak Strawberry 320 ml	8	61	Rp840,000.00	Rp1,152,000.00	Rp312,000.00
56	Larutan Penyegar Cap Badak Jambu Can 320 ml	5	38	Rp525,000.00	Rp720,000.00	Rp195,000.00
57	Larutan Penyegar Cap Badak Lychee Can 320 ml	4	31	Rp420,000.00	Rp576,000.00	Rp156,000.00
58	Ale-Ale Sirsak 200 ml	10	48	Rp185,000.00	Rp240,000.00	Rp55,000.00
59	Ultra Sari Kacang Ijo 200 ml	7	34	Rp560,000.00	Rp621,600.00	Rp61,600.00

60	Mirai Ocha Pet Bottle 450 ml	6	65	Rp576,000.00	Rp604,800.00	Rp28,800.00
61	Mytea Original Bottle 450 ml	6	16	Rp210,000.00	Rp234,000.00	Rp24,000.00
62	Coolant 350 ml	5	42	Rp452,500.00	Rp780,000.00	Rp327,500.00
63	Adem Sari Ching Ku 350 ml	6	50	Rp720,000.00	Rp921,600.00	Rp201,600.00
64	Alto Mineral Water 600 ml	10	144	Rp420,000.00	Rp480,000.00	Rp60,000.00
65	Ades Mineral Water 600 ml	15	216	Rp750,000.00	Rp1,224,000.00	Rp474,000.00
66	Vit Levite 350 ml	8	34	Rp304,000.00	Rp412,800.00	Rp108,800.00
67	Kratingdaeng 150 ml	15	14	Rp420,000.00	Rp585,000.00	Rp165,000.00
68	Kopiko Botol 78 C 240 ml	10	58	Rp1,175,000.00	Rp1,416,000.00	Rp241,000.00
69	Pulpy Orange 350 ml	7	29	Rp444,500.00	Rp546,000.00	Rp101,500.00
70	Frestea Pet 350 ml	8	22	Rp276,000.00	Rp300,800.00	Rp24,800.00
71	Fatigon Hydro 250 ml	5	30	Rp425,000.00	Rp444,000.00	Rp19,000.00
72	Q Guava 350 ml	8	34	Rp328,000.00	Rp480,000.00	Rp152,000.00

Script Algoritma *Dynamic Programming* pada Kasus

```
tic;
n=72;
m=1800;
M=1800;
r=[75000 50000 80000 20000 60000 800000 272000 187500 20000 371000 102200
146000 146000 108000 135000 90000 651800 710000 454500 65000 580000 90000
144000 75000 24000 84000 40000 120000 184000 90000 48000 61000 35000 25000
120000 220000 19000 80000 32000 42000 620000 16000 25000 125000 99000 120000
120000 615000 555000 112800 94000 650000 572000 110000 312000 195000 156000
55000 61600 28800 24000 327500 201600 60000 474000 108800 165000 241000
101500 24800 19000 152000];
w=[99 86 120 230 119 360 180 96 31 60 21 30 30 61 77 51 43 72 180 72 120 126
144 135 96 55 4 72 67 144 41 72 22 43 50 104 32 72 25 72 120 60 66 111 21 8 8
60 60 36 30 150 132 144 61 38 31 48 34 65 16 42 50 144 216 34 14 58 29 22 30
34];
p=1
z=zeros(n+1,m+1);
beban=0;
%tahap 1 sampai n+1
for i=2:n+1
    y(1)=0;
    z(i,1)=0;
    for j=2:m+1
        y(j)=y(j-1)+p;
        if w(i-1)<=y(j)
            z(i,j)=max([z(i-1,j)  r(i-1)+z(i-1,j-w(i-1))]);
        else
            z(i,j)=z(i-1,j);
        end
    end
end
disp(z);
%mencari solusi optimum
keuntungan = (z(i,m+1))
%Mencari Barang yang dibeli
for i=n+1:-1:2
    if z(i,m+1)~=z(i-1,m+1)
        a(i)=1;
        M=m;
        q=M-w(i-1);
        m=round(q);
    else
        a(i)=0;
    end
end
disp('jenis barang yang di angkut')
a=a(2:n+1)
for i=1:n
    if a(i)==1
        beban=beban+w(i);
    end
end
disp('Jumlah beban yang di angkut')
beban
toc;
```

Tampilan *Output* dari Permasalahan *Knapsack* 0-1 dengan Menggunakan Algoritma *Dynamic Programming*

Columns 1747 through 1752

0	0	0	0	0	0
75000	75000	75000	75000	75000	75000
125000	125000	125000	125000	125000	125000
205000	205000	205000	205000	205000	205000
225000	225000	225000	225000	225000	225000
285000	285000	285000	285000	285000	285000
1085000	1085000	1085000	1085000	1085000	1085000
1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4004500	4004500	4004500	4004500	4004500	4004500
4459000	4459000	4459000	4459000	4459000	4459000
4474000	4474000	4474000	4474000	4474000	4474000
4994000	4994000	4994000	4994000	4994000	4994000
5009000	5009000	5009000	5009000	5009000	5009000
5058000	5058000	5063000	5063000	5063000	5063000
5058000	5058000	5063000	5063000	5063000	5063000
5058000	5058000	5063000	5063000	5063000	5063000
5097000	5097000	5097000	5097000	5097000	5122000
5137000	5137000	5137000	5137000	5137000	5137000
5207000	5207000	5207000	5207000	5207000	5207000
5326000	5326000	5326000	5326000	5326000	5326000
5326000	5326000	5326000	5326000	5326000	5326000
5326000	5326000	5326000	5326000	5326000	5326000
5326000	5326000	5326000	5326000	5326000	5326000
5361000	5361000	5361000	5361000	5361000	5361000
5361000	5361000	5361000	5361000	5361000	5361000
5415000	5415000	5415000	5415000	5415000	5415000

```
   5522000   5522000   5522000   5522000   5522000   5522000
   5522000   5522000   5522000   5522000   5522000   5522000
   5522000   5522000   5522000   5522000   5522000   5522000
   5522000   5522000   5522000   5522000   5522000   5522000
   5522000   5522000   5522000   5522000   5522000   5522000
   5958000   5968000   5973000   5973000   5973000   5973000
   5958000   5968000   5973000   5973000   5973000   5973000
   5958000   5968000   5973000   5973000   5973000   5973000
   5958000   5968000   5973000   5973000   5973000   5973000
   6036000   6037000   6037000   6037000   6037000   6037000
   6137000   6137000   6137000   6156000   6156000   6156000
   6244000   6244000   6244000   6244000   6244000   6257000
   6775000   6775000   6775000   6775000   6775000   6775000
   7240000   7240000   7240000   7240000   7242000   7242000
   7287800   7287800   7287800   7287800   7299800   7299800
   7326800   7330800   7330800   7330800   7330800   7338800
   7709300   7709300   7709300   7709300   7709300   7709300
   8006300   8006300   8006300   8006300   8006300   8006300
   8006300   8006300   8006300   8006300   8006300   8006300
   8210300   8210300   8210300   8210300   8210300   8210300
   8285300   8292500   8292500   8292500   8292500   8292500
   8406300   8406300   8406300   8406300   8406300   8406300
   8406300   8406300   8406300   8406300   8406300   8406300
   8406300   8406300   8406300   8406300   8406300   8406300
   8406300   8406300   8406300   8406300   8406300   8406300
   8406300   8406300   8406300   8406300   8406300   8406300
   8584800   8586300   8621000   8621000   8621000   8621000
   8697900   8697900   8697900   8697900   8697900   8697900
   8697900   8697900   8697900   8697900   8697900   8697900
   8729400   8729400   8729400   8729400   8729400   8729400
   8753200   8753200   8753200   8753200   8753200   8753200
   8883200   8883200   8883200   8883200   8883200   8883200
   8990200   8990200   8992700   8992700   9014200   9014200
   9023300   9033300   9033300   9033300   9033300   9033300
   9023300   9033300   9033300   9033300   9033300   9033300
   9023300   9033300   9033300   9033300   9033300   9033300
   9113700   9123700   9123700   9123700   9123700   9123700

Columns 1753 through 1758

         0         0         0         0         0         0
     75000     75000     75000     75000     75000     75000
    125000    125000    125000    125000    125000    125000
    205000    205000    205000    205000    205000    205000
    225000    225000    225000    225000    225000    225000
    285000    285000    285000    285000    285000    285000
   1085000   1085000   1085000   1085000   1085000   1085000
```

1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4004500	4004500	4004500	4004500	4004500	4004500
4459000	4459000	4459000	4459000	4459000	4459000
4474000	4474000	4474000	4474000	4474000	4504000
4994000	4994000	4994000	4994000	4994000	5004000
5009000	5009000	5009000	5009000	5009000	5019000
5063000	5063000	5073000	5073000	5073000	5073000
5063000	5063000	5073000	5073000	5073000	5073000
5063000	5063000	5073000	5073000	5073000	5073000
5122000	5122000	5122000	5122000	5122000	5122000
5137000	5137000	5137000	5162000	5162000	5162000
5207000	5207000	5207000	5217000	5217000	5217000
5326000	5326000	5346000	5346000	5346000	5346000
5326000	5326000	5346000	5346000	5346000	5346000
5326000	5326000	5346000	5346000	5346000	5346000
5326000	5326000	5346000	5346000	5346000	5346000
5361000	5361000	5361000	5361000	5361000	5361000
5361000	5361000	5361000	5361000	5361000	5361000
5415000	5415000	5422000	5422000	5422000	5422000
5522000	5522000	5522000	5557000	5557000	5557000
5522000	5522000	5522000	5557000	5557000	5557000
5522000	5522000	5522000	5557000	5557000	5557000
5522000	5522000	5522000	5557000	5557000	5557000
5522000	5522000	5522000	5557000	5557000	5557000
5522000	5522000	5522000	5557000	5557000	5557000
5973000	5973000	5973000	5973000	5973000	5973000
5973000	5973000	5973000	5973000	5973000	5973000
5973000	5973000	5973000	5973000	5973000	5973000
5973000	5973000	5973000	5973000	5973000	5973000
6037000	6037000	6037000	6037000	6037000	6052000
6156000	6156000	6156000	6157000	6157000	6157000
6257000	6257000	6257000	6257000	6257000	6276000
6775000	6775000	6775000	6775000	6788000	6788000
7242000	7244000	7244000	7244000	7244000	7244000
7299800	7299800	7299800	7299800	7307800	7307800
7338800	7338800	7338800	7340800	7340800	7340800
7733800	7733800	7733800	7733800	7733800	7733800
8006300	8006300	8010800	8010800	8018300	8018300
8006300	8006300	8010800	8010800	8018300	8018300
8210300	8210300	8210300	8210300	8210300	8210300
8298300	8311300	8311300	8311300	8311300	8330300

```
   8406300   8406300   8406300   8406300   8413500   8413500
   8406300   8406300   8406300   8406300   8413500   8413500
   8406300   8406300   8406300   8406300   8413500   8413500
   8406300   8406300   8406300   8406300   8413500   8413500
   8406300   8406300   8406300   8406300   8413500   8413500
   8621000   8621000   8639800   8639800   8639800   8639800
   8697900   8697900   8697900   8697900   8700400   8700400
   8697900   8697900   8697900   8697900   8700400   8700400
   8729400   8729400   8729400   8729400   8753400   8753400
   8753200   8753200   8753200   8753200   8753400   8753800
   8890400   8890400   8907200   8907200   8907200   8907200
   9014200   9014200   9014200   9014200   9025200   9025200
   9033300   9033300   9041300   9059200   9059200   9059200
   9033300   9033300   9041300   9059200   9059200   9059200
   9033300   9033300   9041300   9059200   9059200   9059200
   9123700   9123700   9131700   9131700   9131700   9131700

Columns 1759 through 1764

         0         0         0         0         0         0
     75000     75000     75000     75000     75000     75000
    125000    125000    125000    125000    125000    125000
    205000    205000    205000    205000    205000    205000
    225000    225000    225000    225000    225000    225000
    285000    285000    285000    285000    285000    285000
   1085000   1085000   1085000   1085000   1085000   1085000
   1357000   1357000   1357000   1357000   1357000   1357000
   1544500   1544500   1544500   1544500   1544500   1544500
   1564500   1564500   1564500   1564500   1564500   1564500
   1935500   1935500   1935500   1935500   1935500   1935500
   2037700   2037700   2037700   2037700   2037700   2037700
   2183700   2183700   2183700   2183700   2183700   2183700
   2329700   2329700   2329700   2329700   2329700   2329700
   2437700   2437700   2437700   2437700   2437700   2437700
   2572700   2572700   2572700   2572700   2572700   2572700
   2662700   2662700   2662700   2662700   2662700   2662700
   3314500   3314500   3314500   3314500   3314500   3314500
   4004500   4004500   4004500   4004500   4004500   4004500
   4459000   4459000   4459000   4459000   4459000   4459000
   4504000   4504000   4504000   4504000   4504000   4504000
   5024000   5024000   5024000   5024000   5024000   5024000
   5024000   5024000   5024000   5024000   5024000   5024000
   5073000   5073000   5073000   5073000   5073000   5073000
   5073000   5073000   5073000   5073000   5073000   5073000
   5073000   5073000   5073000   5073000   5073000   5073000
   5122000   5122000   5122000   5122000   5122000   5122000
   5162000   5162000   5162000   5162000   5162000   5162000
   5217000   5227000   5227000   5227000   5227000   5227000
   5346000   5346000   5346000   5346000   5346000   5346000
   5346000   5346000   5346000   5346000   5346000   5346000
```

5346000	5346000	5346000	5346000	5346000	5346000
5346000	5346000	5346000	5346000	5346000	5346000
5361000	5361000	5361000	5361000	5361000	5361000
5361000	5361000	5361000	5361000	5361000	5361000
5422000	5422000	5422000	5422000	5422000	5422000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5973000	5985000	5985000	5985000	5985000	5985000
5973000	5985000	5985000	5985000	5985000	5985000
5973000	5985000	5985000	5985000	5985000	5985000
5973000	5985000	5985000	5985000	5985000	5985000
6052000	6052000	6052000	6052000	6052000	6052000
6157000	6157000	6157000	6157000	6157000	6157000
6276000	6276000	6276000	6276000	6276000	6277000
6788000	6788000	6788000	6788000	6807000	6807000
7259000	7259000	7259000	7259000	7259000	7259000
7307800	7307800	7317800	7322800	7322800	7322800
7342800	7342800	7343800	7343800	7343800	7357800
7733800	7733800	7733800	7733800	7736300	7736300
8050800	8050800	8050800	8050800	8050800	8050800
8050800	8050800	8050800	8050800	8050800	8050800
8210300	8210300	8211300	8211300	8211300	8211300
8330300	8330300	8330300	8370300	8370300	8370300
8413500	8413500	8413500	8413500	8432300	8432300
8413500	8413500	8413500	8413500	8432300	8432300
8413500	8413500	8413500	8413500	8432300	8432300
8413500	8413500	8413500	8413500	8432300	8432300
8430300	8430300	8430300	8430300	8432300	8432300
8639800	8639800	8639800	8639800	8639800	8639800
8724500	8724500	8724500	8724500	8724500	8724500
8724500	8724500	8724500	8724500	8724500	8724500
8753400	8753400	8753400	8753400	8764400	8764400
8779800	8779800	8779800	8779800	8779800	8779800
8909200	8909200	8918200	8918200	8918200	8918200
9025200	9025200	9025200	9025200	9025200	9025200
9059200	9059200	9059200	9059200	9059200	9091700
9059200	9059200	9059200	9059200	9059200	9091700
9059200	9059200	9059200	9059200	9059200	9091700
9131700	9131700	9137700	9137700	9137700	9147700

Columns 1765 through 1770

0	0	0	0	0	0
75000	75000	75000	75000	75000	75000
125000	125000	125000	125000	125000	125000
205000	205000	205000	205000	205000	205000
225000	225000	225000	225000	225000	225000
285000	285000	285000	285000	285000	285000
1085000	1085000	1085000	1085000	1085000	1085000

1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4004500	4004500	4024500	4024500	4024500	4024500
4459000	4459000	4459000	4459000	4459000	4459000
4504000	4504000	4504000	4504000	4504000	4504000
5024000	5024000	5024000	5024000	5024000	5024000
5034000	5034000	5034000	5034000	5034000	5034000
5073000	5073000	5073000	5073000	5073000	5073000
5073000	5073000	5073000	5073000	5073000	5073000
5073000	5073000	5073000	5073000	5073000	5073000
5122000	5122000	5122000	5122000	5122000	5122000
5162000	5162000	5162000	5162000	5162000	5162000
5227000	5227000	5227000	5227000	5227000	5227000
5346000	5346000	5346000	5346000	5346000	5346000
5346000	5346000	5346000	5346000	5346000	5346000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5450000	5450000	5450000	5450000	5450000	5450000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5557000	5557000	5557000	5557000	5557000	5557000
5985000	5985000	5985000	5986000	5990000	6003000
5985000	5985000	5985000	5986000	5990000	6003000
5985000	5985000	5985000	5986000	5990000	6003000
5985000	5985000	5985000	5986000	5990000	6003000
6057000	6057000	6057000	6057000	6067000	6072000
6157000	6172000	6172000	6172000	6172000	6172000
6277000	6277000	6277000	6277000	6277000	6277000
6807000	6807000	6807000	6807000	6807000	6807000
7259000	7259000	7259000	7259000	7259000	7271000
7322800	7322800	7322800	7322800	7322800	7322800
7366800	7366800	7366800	7366800	7366800	7381800
7736300	7736300	7736300	7739300	7744300	7744300
8050800	8050800	8083300	8083300	8083300	8083300
8050800	8050800	8083300	8083300	8083300	8083300
8223300	8223300	8223300	8223300	8223300	8242800
8370300	8370300	8370300	8370300	8370300	8370300

8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8645000	8645000	8645000	8669800	8669800	8669800
8724500	8724500	8724500	8751400	8751400	8751400
8724500	8724500	8724500	8751400	8751400	8751400
8764400	8764400	8764400	8764400	8764400	8764400
8779800	8779800	8779800	8779800	8779800	8779800
8918200	8918200	8918200	8918200	8918200	8918200
9025200	9025200	9030200	9030200	9051800	9051800
9091700	9091700	9091700	9091700	9091700	9091700
9091700	9091700	9091700	9091700	9091700	9091700
9091700	9091700	9091700	9091700	9091700	9091700
9147700	9147700	9148700	9149700	9149700	9158700

Columns 1771 through 1776

0	0	0	0	0	0
75000	75000	75000	75000	75000	75000
125000	125000	125000	125000	125000	125000
205000	205000	205000	205000	205000	205000
225000	225000	225000	225000	225000	225000
285000	285000	285000	285000	285000	285000
1085000	1085000	1085000	1085000	1085000	1085000
1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4024500	4024500	4024500	4024500	4024500	4024500
4459000	4459000	4459000	4459000	4459000	4459000
4504000	4504000	4504000	4504000	4504000	4504000
5024000	5024000	5024000	5024000	5024000	5024000
5034000	5034000	5034000	5034000	5034000	5034000
5073000	5073000	5073000	5073000	5073000	5073000
5073000	5073000	5073000	5073000	5073000	5073000
5073000	5073000	5073000	5073000	5073000	5073000
5122000	5122000	5127000	5127000	5127000	5127000
5162000	5162000	5162000	5162000	5162000	5162000
5227000	5227000	5227000	5227000	5227000	5227000
5346000	5346000	5346000	5346000	5346000	5346000
5346000	5346000	5346000	5346000	5346000	5346000

5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5450000	5450000	5450000	5450000	5450000	5450000
5557000	5557000	5557000	5557000	5570000	5570000
5557000	5557000	5557000	5557000	5570000	5570000
5557000	5557000	5557000	5557000	5570000	5570000
5557000	5557000	5557000	5557000	5570000	5570000
5557000	5557000	5557000	5557000	5570000	5570000
6003000	6003000	6003000	6005000	6005000	6005000
6003000	6003000	6003000	6005000	6005000	6005000
6003000	6003000	6003000	6005000	6005000	6005000
6003000	6003000	6003000	6005000	6005000	6005000
6072000	6072000	6072000	6072000	6072000	6072000
6172000	6172000	6177000	6177000	6177000	6177000
6277000	6277000	6277000	6292000	6292000	6292000
6824000	6824000	6824000	6824000	6824000	6824000
7271000	7272000	7272000	7272000	7272000	7295000
7322800	7322800	7334800	7334800	7334800	7334800
7381800	7381800	7381800	7381800	7381800	7381800
7744300	7746800	7754300	7754300	7754300	7754300
8083300	8083300	8083300	8083300	8083300	8083300
8083300	8083300	8083300	8083300	8083300	8083300
8242800	8242800	8242800	8242800	8265300	8265300
8370300	8370300	8370300	8370300	8370300	8370300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8441300	8441300	8441300	8441300	8441300	8441300
8669800	8669800	8669800	8669800	8669800	8669800
8751400	8751400	8751400	8751400	8751400	8752900
8751400	8751400	8751400	8751400	8751400	8752900
8764400	8764400	8764400	8764400	8791000	8791000
8798200	8798200	8798200	8798200	8838200	8838200
8918400	8918800	8944800	8944800	8944800	8944800
9051800	9051800	9051800	9051800	9051800	9051800
9091700	9091700	9091700	9091700	9091700	9091700
9091700	9091700	9091700	9091700	9091700	9091700
9091700	9091700	9091700	9091700	9091700	9091700
9158700	9158700	9158700	9158700	9158700	9158700

Columns 1777 through 1782

0	0	0	0	0	0
75000	75000	75000	75000	75000	75000
125000	125000	125000	125000	125000	125000
205000	205000	205000	205000	205000	205000
225000	225000	225000	225000	225000	225000
285000	285000	285000	285000	285000	285000
1085000	1085000	1085000	1085000	1085000	1085000

1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4024500	4024500	4024500	4024500	4024500	4024500
4459000	4459000	4459000	4459000	4459000	4459000
4504000	4504000	4504000	4504000	4504000	4504000
5024000	5024000	5024000	5024000	5024000	5024000
5034000	5034000	5034000	5034000	5034000	5034000
5073000	5073000	5073000	5073000	5073000	5083000
5073000	5073000	5073000	5073000	5073000	5083000
5073000	5073000	5073000	5073000	5073000	5083000
5127000	5127000	5137000	5137000	5137000	5137000
5167000	5167000	5167000	5167000	5167000	5167000
5227000	5227000	5227000	5227000	5227000	5227000
5346000	5346000	5346000	5346000	5346000	5346000
5346000	5346000	5346000	5346000	5346000	5346000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5381000	5381000	5381000	5381000	5381000	5381000
5381000	5381000	5381000	5381000	5381000	5381000
5450000	5450000	5450000	5450000	5450000	5450000
5570000	5570000	5570000	5570000	5570000	5570000
5570000	5570000	5570000	5570000	5570000	5570000
5570000	5570000	5570000	5570000	5570000	5570000
5570000	5570000	5570000	5570000	5589000	5589000
5570000	5570000	5570000	5570000	5589000	5589000
6007000	6007000	6007000	6007000	6007000	6022000
6007000	6007000	6007000	6007000	6007000	6022000
6007000	6007000	6007000	6007000	6007000	6022000
6007000	6007000	6007000	6007000	6007000	6022000
6072000	6072000	6072000	6072000	6084000	6084000
6187000	6192000	6192000	6192000	6192000	6192000
6292000	6292000	6292000	6292000	6297000	6297000
6824000	6824000	6824000	6824000	6824000	6824000
7295000	7295000	7295000	7295000	7295000	7295000
7334800	7334800	7334800	7334800	7335800	7339800
7381800	7381800	7381800	7381800	7393800	7393800
7754300	7765800	7765800	7765800	7765800	7788800
8083300	8083300	8083300	8083300	8085800	8085800
8083300	8083300	8083300	8083300	8085800	8085800
8265300	8275300	8275300	8275300	8275300	8275300
8370300	8370300	8370300	8370300	8370300	8370300

```
   8441300   8441300   8448500   8448500   8448500   8448500
   8441300   8441300   8448500   8448500   8448500   8448500
   8467900   8467900   8467900   8467900   8467900   8467900
   8467900   8467900   8467900   8467900   8467900   8467900
   8467900   8467900   8467900   8467900   8467900   8467900
   8674800   8674800   8674800   8674800   8693800   8693800
   8752900   8752900   8752900   8752900   8759500   8759500
   8752900   8752900   8752900   8752900   8759500   8759500
   8791000   8791000   8791000   8791000   8791000   8791000
   8838200   8838200   8838200   8838200   8838200   8838200
   8944800   8944800   8944800   8944800   8944800   8944800
   9051800   9051800   9051800   9060200   9060200   9060200
   9091700   9094200   9094200   9115700   9115700   9115700
   9091700   9094200   9094200   9115700   9115700   9115700
   9091700   9094200   9094200   9115700   9115700   9115700
   9166700   9166700   9175300   9175300   9175300   9185300

Columns 1783 through 1788

         0         0         0         0         0         0
     75000     75000     75000     75000     75000     75000
    125000    125000    125000    125000    125000    125000
    205000    205000    205000    205000    205000    205000
    225000    225000    225000    225000    225000    225000
    285000    285000    285000    285000    285000    285000
   1085000   1085000   1085000   1085000   1085000   1085000
   1357000   1357000   1357000   1357000   1357000   1357000
   1544500   1544500   1544500   1544500   1544500   1544500
   1564500   1564500   1564500   1564500   1564500   1564500
   1935500   1935500   1935500   1935500   1935500   1935500
   2037700   2037700   2037700   2037700   2037700   2037700
   2183700   2183700   2183700   2183700   2183700   2183700
   2329700   2329700   2329700   2329700   2329700   2329700
   2437700   2437700   2437700   2437700   2437700   2437700
   2572700   2572700   2572700   2572700   2572700   2572700
   2662700   2662700   2662700   2662700   2662700   2662700
   3314500   3314500   3314500   3314500   3314500   3314500
   4024500   4024500   4024500   4024500   4024500   4024500
   4459000   4459000   4459000   4459000   4459000   4459000
   4504000   4504000   4504000   4504000   4504000   4504000
   5024000   5024000   5024000   5024000   5024000   5024000
   5034000   5034000   5034000   5039000   5039000   5039000
   5088000   5088000   5088000   5088000   5088000   5088000
   5088000   5088000   5088000   5088000   5088000   5088000
   5088000   5088000   5088000   5088000   5088000   5088000
   5142000   5142000   5142000   5142000   5142000   5142000
   5177000   5177000   5177000   5177000   5182000   5182000
   5232000   5232000   5232000   5232000   5237000   5237000
   5346000   5346000   5346000   5346000   5346000   5346000
   5346000   5346000   5346000   5346000   5346000   5346000
```

5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5381000	5381000	5381000	5381000	5409000	5409000
5381000	5381000	5381000	5381000	5409000	5409000
5450000	5450000	5450000	5450000	5450000	5450000
5570000	5570000	5570000	5570000	5577000	5577000
5570000	5570000	5570000	5570000	5577000	5577000
5570000	5570000	5570000	5570000	5577000	5577000
5589000	5589000	5589000	5589000	5589000	5589000
5589000	5589000	5589000	5589000	5589000	5589000
6022000	6022000	6022000	6022000	6022000	6022000
6022000	6022000	6022000	6022000	6022000	6022000
6022000	6022000	6022000	6022000	6022000	6022000
6022000	6022000	6022000	6022000	6022000	6022000
6084000	6084000	6084000	6084000	6084000	6084000
6192000	6192000	6192000	6192000	6192000	6192000
6297000	6297000	6307000	6312000	6312000	6312000
6824000	6824000	6824000	6824000	6824000	6824000
7295000	7295000	7295000	7295000	7295000	7295000
7352800	7352800	7352800	7352800	7354800	7354800
7393800	7393800	7393800	7393800	7401800	7401800
7788800	7788800	7788800	7788800	7788800	7788800
8085800	8085800	8085800	8085800	8085800	8085800
8085800	8085800	8085800	8085800	8085800	8085800
8275300	8275300	8283300	8283300	8283300	8283300
8370300	8405300	8405300	8405300	8405300	8405300
8448500	8454300	8467300	8467300	8467300	8467300
8448500	8454300	8467300	8467300	8467300	8467300
8467900	8467900	8467900	8467900	8467900	8467900
8467900	8467900	8467900	8467900	8467900	8467900
8467900	8467900	8467900	8467900	8467900	8467900
8693800	8693800	8733800	8733800	8733800	8733800
8759500	8775400	8775400	8775400	8775400	8777400
8759500	8775400	8775400	8775400	8775400	8777400
8791000	8791000	8791000	8791000	8796900	8796900
8838200	8838200	8838200	8838200	8838200	8838200
8944800	8944800	8963200	8963200	8963200	8963200
9060200	9060200	9075800	9080200	9080200	9080200
9115700	9115700	9115700	9126700	9126700	9126700
9115700	9115700	9115700	9126700	9126700	9126700
9115700	9115700	9115700	9126700	9126700	9126700
9185300	9185300	9185300	9185300	9185300	9185300

Columns 1789 through 1794

0	0	0	0	0	0
75000	75000	75000	75000	75000	75000
125000	125000	125000	125000	125000	125000
205000	205000	205000	205000	205000	205000
225000	225000	225000	225000	225000	225000
285000	285000	285000	285000	285000	285000
1085000	1085000	1085000	1085000	1085000	1085000

1357000	1357000	1357000	1357000	1357000	1357000
1544500	1544500	1544500	1544500	1544500	1544500
1564500	1564500	1564500	1564500	1564500	1564500
1935500	1935500	1935500	1935500	1935500	1935500
2037700	2037700	2037700	2037700	2037700	2037700
2183700	2183700	2183700	2183700	2183700	2183700
2329700	2329700	2329700	2329700	2329700	2329700
2437700	2437700	2437700	2437700	2437700	2437700
2572700	2572700	2572700	2572700	2572700	2572700
2662700	2662700	2662700	2662700	2662700	2662700
3314500	3314500	3314500	3314500	3314500	3314500
4024500	4024500	4024500	4024500	4024500	4024500
4459000	4459000	4459000	4459000	4459000	4459000
4524000	4524000	4524000	4524000	4524000	4524000
5024000	5044000	5044000	5044000	5044000	5044000
5039000	5044000	5044000	5044000	5044000	5044000
5088000	5088000	5088000	5088000	5088000	5088000
5088000	5088000	5088000	5088000	5088000	5088000
5088000	5088000	5088000	5088000	5088000	5088000
5142000	5142000	5142000	5142000	5142000	5142000
5182000	5182000	5182000	5182000	5182000	5182000
5237000	5237000	5237000	5237000	5237000	5237000
5346000	5346000	5346000	5351000	5351000	5351000
5346000	5346000	5346000	5351000	5351000	5351000
5374000	5374000	5374000	5374000	5374000	5374000
5374000	5374000	5374000	5374000	5374000	5374000
5409000	5409000	5409000	5409000	5409000	5409000
5409000	5409000	5409000	5409000	5409000	5409000
5450000	5450000	5450000	5460000	5460000	5460000
5577000	5577000	5577000	5577000	5577000	5577000
5577000	5577000	5577000	5577000	5577000	5577000
5577000	5577000	5577000	5577000	5577000	5577000
5589000	5589000	5589000	5589000	5589000	5589000
5589000	5589000	5589000	5589000	5589000	5589000
6022000	6022000	6022000	6022000	6034000	6034000
6022000	6022000	6022000	6022000	6034000	6034000
6022000	6022000	6022000	6022000	6034000	6034000
6022000	6022000	6022000	6022000	6034000	6034000
6085000	6089000	6102000	6102000	6102000	6102000
6204000	6204000	6204000	6204000	6204000	6204000
6312000	6312000	6312000	6312000	6312000	6312000
6824000	6824000	6824000	6824000	6859000	6859000
7295000	7295000	7295000	7306000	7306000	7306000
7354800	7356800	7356800	7356800	7356800	7356800
7401800	7401800	7411800	7416800	7416800	7416800
7788800	7788800	7788800	7806800	7806800	7806800
8118300	8118300	8118300	8118300	8118300	8118300
8118300	8118300	8118300	8118300	8118300	8118300
8283300	8283300	8283300	8283300	8283300	8283300
8405300	8405300	8405300	8405300	8405300	8405300

```
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8733800   8733800   8733800   8733800   8733800   8733800
   8777400   8786400   8786400   8786400   8786400   8786400
   8777400   8786400   8786400   8786400   8786400   8786400
   8796900   8796900   8815000   8819400   8819400   8819400
   8838200   8838200   8862200   8862200   8862200   8862200
   9003200   9003200   9003200   9003200   9003200   9003200
   9080200   9080200   9086800   9086800   9086800   9086800
   9126700   9126700   9126700   9126700   9126700   9126700
   9126700   9126700   9126700   9126700   9126700   9126700
   9126700   9126700   9126700   9126700   9126700   9126700
   9193300   9211200   9211200   9211200   9211200   9211200

Columns 1795 through 1800

         0         0         0         0         0         0
     75000     75000     75000     75000     75000     75000
    125000    125000    125000    125000    125000    125000
    205000    205000    205000    205000    205000    205000
    225000    225000    225000    225000    225000    225000
    285000    285000    285000    285000    285000    285000
   1085000   1085000   1085000   1085000   1085000   1085000
   1357000   1357000   1357000   1357000   1357000   1357000
   1544500   1544500   1544500   1544500   1544500   1544500
   1564500   1564500   1564500   1564500   1564500   1564500
   1935500   1935500   1935500   1935500   1935500   1935500
   2037700   2037700   2037700   2037700   2037700   2037700
   5577000   5577000   5577000   5577000   5577000   5577000
   5589000   5589000   5589000   5589000   5589000   5589000
   5589000   5589000   5589000   5589000   5589000   5589000
   6022000   6022000   6022000   6022000   6034000   6034000
   6022000   6022000   6022000   6022000   6034000   6034000
   6022000   6022000   6022000   6022000   6034000   6034000
   6022000   6022000   6022000   6022000   6034000   6034000
   6085000   6089000   6102000   6102000   6102000   6102000
   6204000   6204000   6204000   6204000   6204000   6204000
   6312000   6312000   6312000   6312000   6312000   6312000
   6824000   6824000   6824000   6824000   6859000   6859000
   7295000   7295000   7295000   7306000   7306000   7306000
   7354800   7356800   7356800   7356800   7356800   7356800
   7401800   7401800   7411800   7416800   7416800   7416800
   7788800   7788800   7788800   7806800   7806800   7806800
   8118300   8118300   8118300   8118300   8118300   8118300
   8118300   8118300   8118300   8118300   8118300   8118300
   8283300   8283300   8283300   8283300   8283300   8283300
   8405300   8405300   8405300   8405300   8405300   8405300
```

```
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8486300   8486300   8486300   8486300   8526300   8526300
   8733800   8733800   8733800   8733800   8733800   8733800
   8777400   8786400   8786400   8786400   8786400   8786400
   8777400   8786400   8786400   8786400   8786400   8786400
   8796900   8796900   8815000   8819400   8819400   8819400
   8838200   8838200   8862200   8862200   8862200   8862200
   9003200   9003200   9003200   9003200   9003200   9003200
   9080200   9080200   9086800   9086800   9086800   9086800
   9126700   9126700   9126700   9126700   9126700   9126700
   9126700   9126700   9126700   9126700   9126700   9126700
   9126700   9126700   9126700   9126700   9126700   9126700
   9193300   9211200   9211200   9211200   9211200   9211200

Columns 1795 through 1800

         0         0         0         0         0         0
     75000     75000     75000     75000     75000     75000
    125000    125000    125000    125000    125000    125000
    205000    205000    205000    205000    205000    205000
    225000    225000    225000    225000    225000    225000
    285000    285000    285000    285000    285000    285000
   1085000   1085000   1085000   1085000   1085000   1085000
   1357000   1357000   1357000   1357000   1357000   1357000
   1544500   1544500   1544500   1544500   1544500   1544500
   1564500   1564500   1564500   1564500   1564500   1564500
   1935500   1935500   1935500   1935500   1935500   1935500
   2037700   2037700   2037700   2037700   2037700   2037700
   2183700   2183700   2183700   2183700   2183700   2183700
   2329700   2329700   2329700   2329700   2329700   2329700
   2437700   2437700   2437700   2437700   2437700   2437700
   2572700   2572700   2572700   2572700   2572700   2572700
   2662700   2662700   2662700   2662700   2662700   2662700
   3314500   3314500   3314500   3314500   3314500   3314500
   4024500   4024500   4024500   4024500   4024500   4024500
   4459000   4459000   4459000   4459000   4459000   4459000
   4524000   4524000   4524000   4524000   4524000   4524000
   5044000   5044000   5044000   5044000   5044000   5044000
   5044000   5054000   5054000   5054000   5064000   5064000
   5088000   5088000   5088000   5088000   5088000   5088000
   5088000   5088000   5088000   5088000   5088000   5088000
   5088000   5088000   5088000   5088000   5088000   5088000
   5142000   5142000   5142000   5142000   5142000   5142000
   5182000   5182000   5182000   5182000   5182000   5182000
   5237000   5237000   5237000   5237000   5237000   5237000
   5351000   5391000   5391000   5391000   5391000   5391000
   5351000   5391000   5391000   5391000   5391000   5391000
```

5374000	5394000	5394000	5394000	5394000	5394000
5374000	5394000	5394000	5394000	5394000	5394000
5409000	5409000	5409000	5409000	5409000	5409000
5409000	5409000	5409000	5409000	5409000	5409000
5460000	5481000	5481000	5481000	5481000	5481000
5577000	5577000	5605000	5605000	5605000	5605000
5577000	5577000	5605000	5605000	5605000	5605000
5577000	5577000	5605000	5605000	5605000	5605000
5589000	5589000	5605000	5605000	5605000	5605000
5589000	5589000	5605000	5605000	5605000	5605000
6035000	6035000	6035000	6035000	6058000	6058000
6035000	6035000	6035000	6035000	6058000	6058000
6035000	6035000	6035000	6035000	6058000	6058000
6035000	6035000	6035000	6035000	6058000	6058000
6104000	6104000	6104000	6106000	6106000	6106000
6204000	6204000	6205000	6209000	6222000	6222000
6312000	6312000	6324000	6324000	6324000	6324000
6859000	6859000	6859000	6859000	6859000	6859000
7306000	7306000	7306000	7330000	7330000	7330000
7371800	7371800	7371800	7371800	7371800	7371800
7416800	7416800	7416800	7416800	7416800	7416800
7806800	7806800	7806800	7806800	7806800	7806800
8118300	8118300	8118300	8118300	8118300	8118300
8118300	8118300	8118300	8118300	8118300	8118300
8283300	8295300	8300300	8301300	8301300	8310300
8405300	8405300	8405300	8405300	8406300	8406300
8526300	8526300	8526300	8526300	8526300	8526300
8526300	8526300	8526300	8526300	8526300	8526300
8526300	8526300	8526300	8526300	8526300	8526300
8526300	8526300	8526300	8526300	8526300	8526300
8526300	8526300	8526300	8526300	8526300	8526300
8733800	8733800	8733800	8733800	8741000	8741000
8786400	8786400	8786400	8787900	8822600	8822600
8786400	8786400	8786400	8787900	8822600	8822600
8829400	8829400	8829400	8829400	8829400	8829400
8862200	8862200	8873200	8873200	8873200	8873200
9003200	9003200	9003200	9003200	9003200	9003200
9086800	9098200	9124200	9124200	9124200	9124200
9126700	9131700	9131700	9153300	9153300	9153300
9126700	9131700	9131700	9153300	9153300	9153300
9126700	9131700	9131700	9153300	9153300	9153300
9211200	9211200	9211200	9243700	9243700	9243700

Column 1801

0
75000
125000
205000
225000
285000
1085000

1357000
1544500
1564500
1935500
2037700
2183700
2329700
2437700
2572700
2662700
3314500
4024500
4459000
4524000
5044000
5064000
5088000
5088000
5088000
5142000
5182000
5237000
5391000
5391000
5394000
5394000
5409000
5409000
5481000
5605000
5605000
5605000
5605000
5605000
6058000
6058000
6058000
6058000
6106000
6222000
6324000
6859000
7330000
7371800
7416800
7806800
8118300
8118300
8310300
8406300

```
     8526300
     8526300
     8526300
     8526300
     8526300
     8757800
     8822600
     8822600
     8829400
     8873200
     9003200
     9124200
     9153300
     9153300
     9153300
     9243700


keuntungan =

     9243700

jenis barang yang di angkut

a =

  Columns 1 through 12

     0     0     0     0     0     0     0     0     0     1     1     1

  Columns 13 through 24

     1     0     0     0     1     1     1     0     1     0     0     0

  Columns 25 through 36

     0     0     1     0     1     0     0     0     0     0     1     1

  Columns 37 through 48

     0     0     0     0     1     0     0     0     1     1     1     1

  Columns 49 through 60

     1     1     1     1     1     0     1     1     1     0     0     0
```

```
  Columns 61 through 72

     0     1     1     0     0     1     1     1     1     0     0     1

Jumlah beban yang di angkut

beban =

        1797

Elapsed time is 1.341294 seconds.
```

SURAT PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya yang bertanda tangan di bawah ini, mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta:

Nama	:	Anggita Dyah Ayu Pitaloka
No. Registrasi	:	3125121978
Jurusan	:	Matematika
Program Studi	:	Matematika

Menyatakan bahwa skripsi ini yang saya buat dengan judul "**Permasalahan Optimasi *Knapsack* 0-1 dengan Menggunakan Algoritma *Dynamic Programming***" adalah :

1. Dibuat dan diselesaikan oleh saya sendiri.
2. Bukan merupakan duplikat skripsi yang pernah dibuat oleh orang lain atau jiplakan karya tulis orang lain.

Pernyataan ini dibuat dengan sesungguhnya dan saya bersedia menanggung segala akibat yang timbul jika pernyataan saya tidak benar.

Depok, Januari 2017

Yang membuat pernyataan

Anggita Dyah Ayu Pitaloka

DAFTAR RIWAYAT HIDUP

ANGGITA DYAH AYU PITALOKA. Lahir di Jakarta, 31 Agustus 1994. Anak pertama dari pasangan Bapak Teguh Harjono dan Ibu Siti Nurjanah. Saat ini bertempat tinggal di Kp. Sindangkarsa RT 006/005 No. 76 Kel. Sukamaju Baru Kec. Tapos-Depok 16462.

No. Ponsel : 089667794274

Email : anggitadyahayu@gmail.com

Riwayat Pendidikan : Penulis mengawali pendidikan di TK Kartika X-VII selama 2 tahun, dan kemudian melanjutkan pendidikan di SDN Pekayon 05 Pagi selama dua tahun dan berpindah ke SDN Sukatani VII hingga lulus SD tahun 2006. Setelah itu, penulis melanjutkan ke SMPN 11 Depok tahun 2006-2009. Kemudian kembali melanjutkan ke SMAN 99 Jakarta tahun 2009-2012. Di Tahun 2012 penulis melanjutkan ke Universitas Negeri Jakarta (UNJ), jurusan Matematika, melalui jalur SNMPTN. Di awal tahun 2017 penulis telah memperoleh gelar Sarjana Sains untuk Jurusan Matematika, Program Studi Matematika, FMIPA, UNJ.

Riwayat Organisasi : Selama perkuliahan, penulis pernah aktif di UKM (Unit Kesenian MAhasiswa) UNJ di tahun pertama. Selain itu, penulis pernah menjadi panitia pelangi matematika UNJ selama dua tahun berturut-turut sebagai humas. Di luar itu, penulis merupakan anggota Paskibra di SMAN 99 Jakarta yang menjabat sebagai pengawas dan kedisiplinan. Selain itu, penulis pernah menjadi pembawa bendera pada saat 17 Agustus di Kecamatan.

Riwayat Pekerjaan : Penulis mulai mengajar pada tahun 2013 di LP3I selama setahun. Setelah itu, penulis juga mengajar di bimbingan belajar Assa *Course* hingga saat ini. Penulis juga pernah magang atau PKL di Badan Kepegawaian Negara (BKN) yang di tempatkan pada divisi Keuangan.