# Solusi Optimum Minmax 0/1 Knapsack Menggunakan Algoritma Greedy

**Raja Sabaruddin**
Manajemen Informatika, AMIK BSI Pontianak
raja.rjd@bsi.ac.id

***Abstract -*** *Knapsack is a container used for storing objects of the same size or less in some capacity . The problem that often arises when searching for the optimal choice of the object to be inserted into a container with limited capacity . At the loading of goods using container for example , loading the object or goods to be delivered must minimize the total weight or volume capacity without exceeding the maximum limit . This analysis mengggunakan 0-1 knapsack , which is the object taken entirely or not taken . This research aims to develop a greedy algorithm to solve knapsack minmax 0/1 . Pointing to research results that the solution of 0/1 knapsack minmax using greedy algorithm can be used to produce the optimal solution of the problem of loading the container so that the minimum and maximum capacity constraints are met* .
***Keywords: Knapsack 0/1, Greedy Algorithms, Containers***

**Abstrak** - Knapsack adalah wadah yang digunakan untuk menyimpan benda-benda dengan ukuran yang sama atau kurang dalam beberapa kapasitas. Masalah yang sering timbul ketika mencari pilihan yang optimal dari objek yang akan dimasukkan ke dalam wadah dengan kapasitas terbatas. Pada pemuatan barang menggunakan kontainer misalnya, memuat objek atau barang yang akan dikirim harus meminimalkan total berat atau volume kapasitas tanpa melebihi batas maksimum. Analisis ini mengggunakan 0-1 ransel, yang merupakan objek diambil seluruhnya atau tidak diambil. Penelitian ini bertujuan untuk mengembangkan algoritma serakah untuk memecahkan ransel minmax 0/1. Menunjuk ke hasil penelitian bahwa solusi dari 0/1 ransel minmax menggunakan algoritma greedy dapat digunakan untuk menghasilkan solusi yang optimal dari masalah loading wadah sehingga minimum dan kapasitas maksimum kendala terpenuhi.
***Kata Kunci: Knapsack 0/1, Algoritma Greedy, Kontainer.***

## A. PENDAHULUAN

Knapsack merupakan suatu kantong atau tempat yang digunakan untuk memuat sesuatu objek. Kantong atau tempat tersebut bisa menampung beberapa objek saja dengan ketentuan total ukuran objek tersebut sama atau lebih kecil dengan ukuran kapasitasnya. Setiap objek tidak harus dimasukkan secara keseluruhan namun bisa sebagian objek saja. Untuk penilaian cara ini bukan hanya dari hasil nilai optimalnya. Banyak tahap-tahap yang diperlukan untuk mendapatkan penyelesaian masalah tersebut.

Sehingga untuk menyelesaikan permasalahan knapsack 0-1 diperlukan suatu cara yang dapat menghasilkan solusi yang optimal, efektif dan efesien yaitu dengan menggunakan strategi algoritma greedy. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang dapat memberikan perolehan yang terbaik yaitu dengan membuat pilihan minimum pada setiap langkah dengan tujuan sisanya mengarah ke solusi maksimum yang optimal.

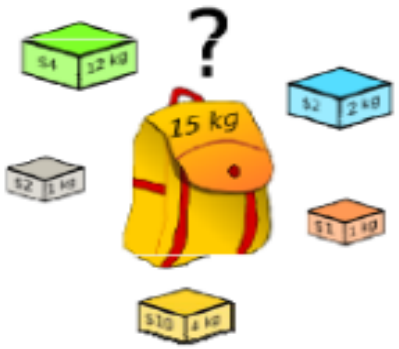
Untuk mendapatkan hasil yang optimal dalam menyelesaikan permasalahan knapsack 0-1 adalah dengan strategi algoritma greedy. Konsep kerja algoritma greedy dalam meyelesaikan masalah knapsack 0/1 adalah dengan cara strategi greedy by profit, atau greedy by weight, atau dapat juga diselesaikan dengan greedy by density. Setelah dilakukan analisis ternyata dapat dibuktikan bahwa algoritma greedy dapat mengurangi jumlah langkah komlpleksitas waktu asimpotiknya adalah 0(n).

Implementasi dari minmax 0/1 knapsack adalah pemuatan barang ke dalam tempat yang bertujuan untuk meminimalkan penggunaan ruang kontainer yang tersedia pada pengiriman barang antar pulau atau negara. Dalam masalah ini item yang dimasukkan ke dalam wadah harus memenuhi batas minimum dan tidak melebihi kapasitas maksimum.

## B. TINJAUAN PUSTAKA

1. Permasalahan Knapsack 0/1

Knapsack adalah tas atau karung yang digunakan untuk memuat sesuatu yang tentunya tidak semua objek dapat ditampung di dalam karung tersebut. Karung tersebut hanya dapat menyimpan beberapa objek dengan total ukuran (weight) lebih kecil atau sama dengan ukuran kapasitas karung. Ilustrasi dapat dilihat pada gambar 1. Pada gambar 1 terlihat sebuah tas berkapasitas 15 kg dan terdapat 5 barang dengan berat dan keuntungan masing-masing yang menjadi persoalan adalah barang mana saja yang harus dimasukkan ke dalam tas.




Gambar 1. Ilustrasi Knapsack

Knapsack problem secara secara matematis dapat ditulis sebagai berikut:
Diberikan bobot knapsack adalah M. diketahui n buah objek yang masing-masing bobotnya adalah:

$$M=b_1w_1 + b_2w_2 + \ldots + b_nw_n$$

Dalam hal ini, bi bernilai 0 atau 1. Jika bi = 1, berarti objek I dimasukkan ke dalam knapsack, sebaliknya jika bi = 0, objek I tidak dimasukkan. Berhubungan ini bi 0 dan 1 maka masalah ini sering juga disebut sebagai permasalahan knapsack 0/1.

Persoalan knapsack termasuk ke dalam NPComplete. Persoalan NPComplete tidak dapat dipecahkan dalam ode waktu polynomial(Paryati, 2009:2).

2. Algoritma

Algoritma merupakan suatu teknik penyusunan langkah dalam penyelesaian masalah dan berbetuk kalimat dengan jumlah kata terbatas tetapi tersusun secara logis dan sistematis atau bisa juga disebut suatu prosedur yang jelas untuk menyelesaikan persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya (Suarga, 2012:1).

Agar algoritma dapat ditulis lebih teratur maka sebaiknya dibagi ke dalam beberapa bagian (Suarga, 2012:6) yaitu sebagai berikut:

a) Bagian Kepala (*Header*) yaitu memuat nama algoritma serta informasi atau keterangan tentang algoritma yang ditulis.
b) Bagian Deklarasi (Defenisi Variabel) yaitu nama variable yang nama tetapan, nama prosedur, nama fungsi, tipe data yang akan digunakan dalam algoritma.
c) Bagian Deskripsi (Rincian langkah) yaitu langkah-langkah penyelesaian masalah, termasuk beberapa perintah seperti baca data, tampilkan, ulangi, yang mengubah data input menjadi output.

3. Algoritma Greedy

Algoritma greedy merupakan salah satu metode yang popular untuk memecahkan persoalan optimasi. Maksud dari pemecahan persoalan optiasi sendiri adalah mencari solusi paling optimum dari segala kemungkinan yang ada. Persoalan *maximization* atau memaksimalkan segala kemungkinan yang terjadi kedepannya. Kemudian permasalah *minimization* atau minimalisasi dari segala kemungkinan yang akan terjadi kedepannya.

Sesuai dengan namanya, greedy dalam bahasa Indonesia artinya rakus, tamak, dan loba, maka algoritma ini memiliki prinsip "*Take what you can get now*!". Algoritma greedy ini membentuk solusi langkah, pada setiap langkahnya tentu path tersebut akan memiliki banyak pilihan dan kemungkinan yang dapat di eksplorasi, dengan algoritma ini keputusan langkah yang di ambil berikutnya adalah yang paling menguntungkan pada keadaan sekarang (Kesuma,2014).

Sedangkan menurut Rachmawati 2013:187) mengungkapkan algoritma greedy adalah algoritma yang lazim untuk memecahkan persoalan optimasi maeskipun hasilnya tidak selalu memberikan solusi yang optimum, dan untuk memecahkan masalah dengan algoritma greedy harus memerlukan elemen-elemen sebagai berikut, himpunan kandat yaitu himpunan berisi elemen-elemen pembentuk solusi, himpunan solusi yaitu himpunan berisi kandidat terpilih sebagai solusi persoalan, fungsi seleksi yaitu fungsi yang ada pada setiap langkah memilih kandidat yang paling memungkinkan guna mencapai solusi optimal, fungsi kelayakan yaitu fungsi yang memeriksa apakan suatu kandidat yang telah dipilih dapat memberikan solusi yang layak dan tidak melanggar batasan atau constraints yang ada, dan yang terakhir fungsi objektif yaitu fungsi yang memaksimumkan atau meminimumkan niai solusi.

Berikut skema umum algoritma greedy :

```
Function greedy (input C:himp_kandidat) ->himp_kandidat
        {
        Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy. Masukkan himpunan kandidat C keluaran: himp. Solusi bertipe himp_kandidat
        }
        Deklarasi
        X= kandidat S=himp_kandidat
        Algoritma
```

```
S<- {  } { inisialisasi S dengan kosong }
While
(not SOLUSI (S)) and (C!={ }) do
X<- seleksi (C) {pilih kandidat C}
C<- -{x} {elemen C berkurang satu}
If LAYAK (S U (x)) then
S<- U {x}
Endif
Endwhile
{
solusi (S) atau C kosong
}
If SOLUSI (S) then
Return
Else
{tidak ditemukan solusi global}
Endif
```

Untuk memilih objek yang akan dimasukkan ke dalam knapsack terdapat beberapa strategi greedy yang heuristic (Paryati, 2009:188) yaitu:

a) *Greedy by Profit*
   Pada setiap langkah, pilih objek yang mempunyai keuntungan dengan memaksimumkan keuntungan dengan memilih objek yngpaling menguntukan terlebih dahulu. Pertama kali yang dilakukan adalah program mengurutkan secara menurun objek-objek berdasarkan profitnya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.
b) *Greedy by weight*
   Pada setiap langkah pilih objek yang mempunyai berat teringan. Mencoba memaksimumkan keuntungan dengan memasukkan sebanyak mungkin objek ke dalam knapsack.

Pertama kali yang dilakukan adalah program mengurutkan secara menaik objek-objek berdasarkan weightnya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan. Pada setiap langkah knapsack di isi dengan objek yang mempunyai pi/wi terbesar, dimana p adalah keuntungan dan w adalah berat barang. Mencoba memaksimumkan keuntungan dengan memilih objek yang mempunyai density per unit berat terbesar. Pertama kali yang dilakukan adalah program mencari nilai profit per berat tiap-tia unit (density) dari semua objek. Kemudian objek-objek tersebut diurutkan berdasarkan density-nya. Kemudian baru diambil satu-persatu objek yag dapat ditampung oleh knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.

4. Kompleksitas

Misalkan $g(n)=n^2-n,g(1)=n^2+100n$ contoh tersebut mempunyai komplesitas yang sama karena nilai fungsinya sama yaitu fungsi kuadrat. Jadi jika rumus tersebut diimplementasikan ke dala bentuk program akan menghasilkan perbandingan waktu yang sama. Algoritma dikatakan mempunyai kompleksitas Ø=(f(n)) jika terdapat konstanta C sehingga @g(n)@≤C@f(n) (Ding-Zhu Du,2000).

5. Bahasa Pemrograman

Menurut Joni(2011:3) mengungkapkan bahwa “pemrograman adalah suatu kumpulan kata (perintah) yang siap digunakan untuk menulis suatu kode program sehingga kode-kode program yang ditulis tersebut akan dapat dikenali oleh kompilator yang sesuai”.

Sedangkan menurut Munir (2007:13) bahasa pemrograman merupakan bahasa komputer yang dibuat untuk menulis program.

6. Pemrograman C

Bahasa C merupakan bahasa yang *powerfull* dan fleksibel yang telah terbukti dapat menyelesaikan program-program besar seperti pembuatan system operasi, pengolahan kata, pengolahan gambar (seperti pembuatan game) dan juga pembuatan kompilator (Joni, 2011:3).

Adapun kerangka pemrograman bahasa C (joni, 2011:7) yaitu sebagai berikut:

```
#include < nama_header_file >

/*fungsi-fungsi yang dibutuhkan ditulis
sebelum fungsi main() sehingga
membutuhkan prototype fungsi */

Tipe_data nama_fungsi1(parameter1,
parameter2, … )
{

Statemen_yang_akan_dieksekusi;
…
}
Tipe_data_nama_fungsi1(parameter1,
parameter2, … )
{
Statemen_yang_akan_dieksekusi;
…
}
/* fungsi utama */
Int main(void)
{
Statemen_yang_akan_dieksekusi;
…
```

```
Return 0;
}
```

7. Kontainer

Kontainer merupakan gudang kecil yang berjalan untuk mengangkut barang dari satu tempat ke tempat lain harus bersama-sama alat pengangkutnya yakni, kapal atau kereta api sampau ke tempat yang dituju, biasanya ke gudang pemilik barang(*exporter* dan *importer*).

Penggerakan kontainer dari satu tempat lain adanya pembatasan territorial/wilayah pembawa muatan di dalamnya (kargo) secara aman, efesien serta dapat dipindah-pindahkan dari jenis angkutan yang satu ke angkutan yang lain, tidek perlu membogkar lagi isi muatannya, oleh karena itu kontainer harus dalam kondisi laik laut(*sea worthy*) mampu mehan getaran pada waktu dalam pengangkutan di jalan raya, rel kereta api ataupun di depot dengan iklim dan suhu yang berbeda-beda, betuk-bentuk kontainer sudah ada ukurannya yang diakui oleh dunia internasional, namun ada kalanya untuk keperluan khusus, bentuknya dapat disesuaikan dengan kebutuhan pemakai(samidjan, 1991:1)

## C. METODE PENELITIAN

Dalam pembahasan ini, metodologi pengembangan sistem dilakukan hanya sampai tahap analisa dan implementasi suatu minmax 0/1 knapsack.

1. Analisa Kasus

Contoh analisa kasus:

Data awal
W1 = 6, P1 = 12
W2 = 5, P2 = 15
W3 = 10, P3 = 50
W4 = 5, P4 = 10
Knapsack = 16

*Greedy by profit*

Pertama kali dilakukan adalah program secara menurun objek-objek berdasarkan profitya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bias dimasukkan.

Tabel 1. *Greedy by profit*

| Property objek | | | | |
|---|---|---|---|---|
| I | wi | Pi | Pi/wi | status |
| 3 | 10 | 50 | 5 | Diambil |
| 2 | 5 | 15 | 3 | Diambil |
| 1 | 6 | 12 | 2 | Tidak |
| 4 | 5 | 10 | 2 | tidak |

*Greedy by weight*

Pertama kali dilakukan adalah program mengurutkan secara menaik objek-objek berdaarkan weightnya. Kemudian diambil satu-persatu objek yang di tamping oleh knapsack penuh atau sudah tidak ada objek lagi yang bias di masukkan

Tabel 2. *Greedy by weight*

| Property objek | | | | |
|---|---|---|---|---|
| I | wi | Pi | Pi/wi | status |
| 2 | 5 | 15 | 3 | Diambil |
| 4 | 5 | 10 | 2 | Diambil |
| 1 | 6 | 12 | 2 | Diambil |
| 3 | 10 | 50 | 5 | tidak |

*Greedy by Density*

Langkah awal yang dilakukan adalah program mencari nilai profit perunit(density) dari tiap-tiap objek. Kemudian objek-objek tersebut diurutkan berdasarkan densitynya. Kemudian baru diambbil satu-persatu objek yang dapat ditampung oleh knapsack hingga knapsack penuh atau sudah tidak adaobjek lagi yang bias dimasukkan.

Tabel 3. *Greedy by density*

| Property objek | | | | |
|---|---|---|---|---|
| I | Wi | Pi | Pi/wi | status |
| 3 | 10 | 50 | 5 | Diambil |
| 2 | 5 | 15 | 3 | Diambil |
| 4 | 5 | 10 | 2 | Tidak |
| 1 | 6 | 12 | 2 | tidak |

2. Algoritma Penyelesaian dengan C++

Efesiensi *Greedy by Profit*

```
procedure GbyProfit(var dt3:
arrayData;
W: real);
Var
i,j : integer
temp: data
cW: real
tukar:boolean program
UrutkanDataBerdasarkanProfit(dt3)
while ((cW<W)and(i<=length(dt3))
do
if(dt3[i].w<=W-cW) then cW
cW+dt3[i].w
dt3[i].status True
inc(i)
```

Berdasarkan algortima diatas maka dapat dihitung kompleksitas waktu asimppotiknya adalah O(n).

**Efesiensi *Greedy by Weight***

```
procedure GbyWeight(var dt3:
arrayData;W: real);
Var
i,j: integer
temp: data
cW: real
tukar: boolean
```

```
Program
UrutkanDataBerdasarkanWeight(dt3)
while ((cW<W)and(i<=length(dt3))
do
if(dt3[i].w<=W-cW)then
cW cW+dt3[i].w
dt3[i].status True
inc(i)
```

Berdasarkan algoritma diatas maka dapat dihitung kompleksitas waktu asimpotiknya adalah O(n).

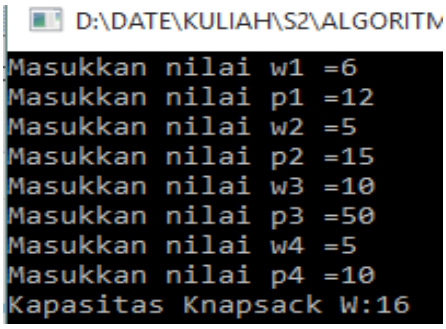
**Efesiensi *Greedy by Density***

```
procedure TForm1.GbyPW(var dt3:
arrayData;W: real);
Var
i,j: integer
temp: data
cW: real
tukar: boolean
Program
UrutkanDataBerdasarkanDensity(d
t3)
//Mengurutkan data bedasarkan
while
((cW<W)and(i<=length(dt3))
do //nilai density yang
terbesar
if(dt3[i].w<=W-cW)then
//ke nilai yang terkecil
cW cW+dt3[i].w
dt3[i].status True
inc(i)
```

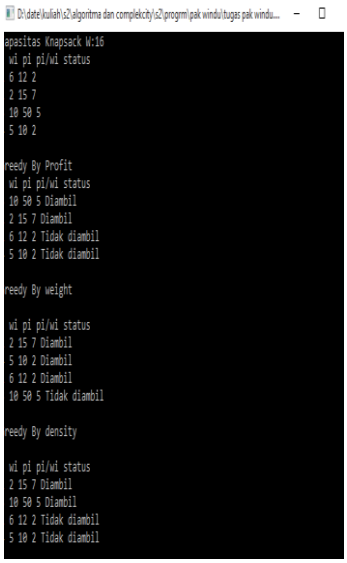
Berdasarkan algoritmaa diatas maka dapat dihitung kompleksitas waktu asimpotiknya adalah O(n).

## D. HASIL DAN PEMBAHASAN

1. Hasil Analisa dengan C++




Gambar 2. *Input*




Gambar 3. *Output*

2. Implementasi MinMax 0/1 Knapsack

Pada bagian ini menerapkan solusi dari masalah minmax 0/1 knapsack menggunakan algoritma greedy untuk masalah pemuatan barang ke dalam wadah, untuk mengoptimalkan ruang penyimpanan. Kapasitas maksimum dan minimum atau volume ruang kontainer yang tersedia dapat dilihat pada table 4.

Data yang digunakan dalam penelitian uni diterima dari PT DFI yang bisnis adalah pengiriman container ke Singapura. Setiap kontainer yang dikirim memiliki batas volume minimum. Jika total volume kontainer kurang dari batas minimum, perusahan akan didenda per meter kubik. Item data di PT DFI adalah item data yang dikumpulkan data pada 5 maret 2010 dan 15 maret 2010, item data yang di sampaikan dalam data stages. Kemudian dimasukkan ke kontainer prusahaan berukuran 20’ dan 40’ masing-masing dan sebanyak dua container yang ditunjukkan di tabel 5.

Penerapan knapsack dengan kendala minmax dilakukan dengan mengisi container pertama. 20’ di mana solusi optimal dihitung dengan pembatasan minimal menghasilkan 30m$^3$ kapasitas 30 m$^3$ 589. Solusi optimal dengan total berat minimal 4.905,30 kg dan ditunjukkan di tabel 6 dan hasilnya tidak melebihi kapasitas maksimu dari 31 152 m$^3$ dan berat maksimum 20 ton kontainer.

Wadah kedua menggunakan ukuran 20’, ini karena yang tersisa cukup untuk wadah ukuran 20’ dan item data yang digunakan adalah item yang tersisa dari data container pertama. Item yang dipilih adalah semua sisa barang-barang dari container pertama. Hasil kedua pemuatan container untuk sisa ada banyak ruang sehingga jika da pengiriman tambahan barang dapat dilakukan untuk

memaksimalkan wadah ruang dan bisa dilihat di tabel 7.

Jika data pada tanggal 5 maret 2010 menggunakan ukuran container 40' maka akan mengisi ruang 49,931 $m^3$ kontainer dan berat total minimum item dipilih dari 14.838,30 kg dan semua item yang dipilih adalah ditunjukkan pada tabel 8. Wadah kedua untuk barang tanggal 15 maret 2010 dilakukan dengan menggunakan container ukuran 40' akan tetapi ada satu item yang masih tetap. Jika semua data pada 5 maret dan 15 maret 2010 dikumpulkan kemudian menghasilkan solusi optimal yang hanya menggunakan 3 kontainer dan ditunjukkan pada tabel 9 sehingga lebih efesien daripada pengiriman oleh PT DFI yang menggunakan 4 kontainer.

Tabel 4. Kontainer data

| No | Container length | Volume | Maximal weight | Minimal volume |
|---|---|---|---|---|
| 1 | 20 feet | 31.152 $m^3$ | 20 ton | 20 $m^3$ |
| 2 | 40 feet | 62.683 $m^3$ | 30 ton | 40 $m^3$ |

Sumber: Rahajo, 2013

Tabel 5. Data dari PT DFI

| No | Container | Date | Capacity | Total weight |
|---|---|---|---|---|
| 1 | 20 | March, 5 2010 | 23.661 $m^3$ | 8,222.50 kg |
| 2 | 20 | March, 5 2010 | 26.270 $m^3$ | 6.615,80 kg |
| 3 | 40 | March, 15 2010 | 45.999 $m^3$ | 13,987,40 kg |
| 4 | 40 | March, 15 2010 | 47.469 $m^3$ | 7,832.20 kg |

Sumber: Rahajo, 2013

Tabel 6. Data 15 maret 2010 dengan minmax 0/1 knapsack

| No | Container | Date | Capacity | Total Weight |
|---|---|---|---|---|
| 1 | 20 | March 5, 2010 | 30.589 $m^3$ | 490.80 kg |
| 2 | 20 | March 5, 2010 | 19.342 $m^3$ | 9932.50 kg |

Sumber: Rahajo, 2013

Tabel 7. tanggal 5 maret 2010 dengan minmax 0/1 knapsack

| No | Container | Date | Capacity | Total Weight |
|---|---|---|---|---|
| 1 | 40 | March 5, 2010 | 49.931$m^3$ | 14838.30 kg |

Sumber : Rahajo, 2013

Tabel 8. Data tanggal 15 maret 2010 dengan minmax 0/1 knapsack

| No | Container | Date | Capacity | Total Weight |
|---|---|---|---|---|
| 1 | 20 | March 15, 2010 | 30.535 $m^3$ | 4744 kg |
| 2 | 40 | March 15, 2010 | 61.933 $m^3$ | 16875.60 kg |

Sumber: Rahajo, 2013

Tabel 9. data 5 maret 2010 dengan minmax 0/1 knapsack

| No | Container | Date | Capacity | Total Weight |
|---|---|---|---|---|
| 1 | 40 | March 5 & 15, 2010 | 62.117 $m^3$ | 19244.30 kg |
| 2 | 40 | March 5 & 15, 2010 | 60.691 $m^3$ | 9858 kg |
| 3 | 20 | March 5 & 15, 2010 | 21.215 $m^3$ | 7555.60 kg |

Sumber: Rahajo, 2013

## E. KESIMPULAN DAN SARAN

1. Kesimpulan

Berdasarkan hasil analisa yang telah dilakukan dalam penelitian ini, maka dapat diambil beberapa kesimpulan sebagai berikut:

a) MinMax 0/1 kanapsak dapat diselesaikan dengan menggunakan pemrograman greedy sehingga nilai total barang terpenuhi secara optimal tanpa melebihi batas kapasitas maksimum.
b) Minmax 0/1 knapsack dapat diterapkan untuk masalah pemuatan barang pada kontainer sehingga berat total minimum terpenuhi tanpa melebihi kapasitas maksimum kontainer.

2. Saran

a) Permasalahan MinMax 0/1 Knapsack pada jurnal ini penulis buat dengan

metode greedy dan bisa di kembangkan lagi dengan metode lain juga.

b) Perlu di implementasinnya terhadap aplikasi yang telah dirancang sehingga dimanfaatkan sebagaimana dengan maksud dan tujuan aplikasi ini.
c) Pembuatan aplikasi MinMax 0/1 Knapsack masih sebatas menggunakan pemrograman C++ dan masih bisa dikembaangkan lagi dengan pemrograman yang lain, seperti pemrograman visual basic, web, maupun java.

**DAFTAR PUSTAKA**


[1] Ding-Zhu du. 2000. *Theory Of Comulation Complexity*. John Wiley & Son.
[2] Joni, made, dan Budi Raharjo. 2011. *Pemrograman C dan Implementasinya*. Bandung: Informatika.
[3] Kesuma, Dharma, Albhikautsar. 2014. *Penerapan Algoritma Greedy untuk Menentukan Penjadwalan Kelas Gedung Labtek V*. Bandung.
[4] Munir, Rinaldi. 2007. *Algoritma dan Pemrograman dalam bahasa Pascal dan C*. Bandung: Informatika bandung.
[5] Paryati. 2009. *Optimasi Strategi Algoritma Greedy untuk Menyelesaikan Permasalahan Knapsack 0-1*. Seminar Nasional Informatika (semnasIF 2009) ISSN:1979-2328.
[6] Samidjan. 1991. *Pengertian dan Penggunaan Peti Kemas (Kontainer)*. Diambil dari https://lib.atmajaya.ac.id/default.aspx?tabID=470&id=107350&lok=1 (pada tangggal 22 januari 2016).
[7] Suarga. 2012. *Algoritma dan pemrograman*. Yogyakarta: Andi Offset.
[8] Wahyu Eko Susanto, *Pendekatan Keamanan Serta Kecepatan Akses Data Pada Cloud Dengan Algoritma Huffman Dan Aes*, Vol 2, No 2 (2014): Jurnal Bianglala Informatika 2014
[9] Saifudin, *Penerapan Algoritma C4.5 Dalam Prediksi Penyewa Sepeda*, Vol 3, No 2 (2015): Jurnal Evolusi 2015
[10] Pudji Widodo, *Rule-Based Classifier Untuk Mendeteksi Penyakit Liver*, Vol 2, No 1 (2014): Jurnal Bianglala Informatika 2014
[11] Sardiarinto, *Aplikasi Sistem Pendukung Keputusan Kelayakan Peminjaman Kredit Nasabah Koperasi Berbasis Android*, Vol 1, No 1 (2013): Bianglala Informatika 2013
[12] Muhammad Multazam, Bambang Eka Purnama, *Influence Of Classified Ad On Google Page Rank And Number Of Visitors*, Journal of Theoretical and Applied Information Technology, Vol. 81. No. 2 – 2015
[13] Karya Gunawan, Bambang Eka Purnama (2015), *Implementation of Location Base Service on Tourism Places in West Nusa Tenggara by using Smartphone*, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 6, No. 8, 2015