

semnasif
seminar nasional teknik informatika

ISSN : 1979-2328

PROSIDING

BUKU 1

Seminar Nasional
Teknik Informatika
e-Democracy
Jogjakarta 23 Mei 2009

“Dukungan e-Democracy dalam meningkatkan mutu demokrasi di Indonesia”



**jurusan teknik informatika
fakultas teknologi industri
upn "veteran" jogjakarta
2009**

PROSIDING



e-DEMOCRACY :

**Dukungan e-Democracy dalam meningkatkan
mutu demokrasi di Indonesia**

YOGYAKARTA, 23 MEI 2009

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UPN "VETERAN" YOGYAKARTA**

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur dipanjatkan kehadirat Allah SWT. Atas berkat rahmat dan hidayah-Nya maka Seminar Nasional Informatika 2009 yang diselenggarakan pada tanggal 23 Mei 2009 di Jurusan Teknik Informatika UPN "Veteran" Yogyakarta dapat berlangsung.

Seminar Nasional Informatika 2009 merupakan Seminar Nasional Kedua yang dilaksanakan oleh Jurusan Teknik Informatika UPN "Veteran" Yogyakarta dan kegiatan ini menjadi agenda tahunan Jurusan Teknik Informatika UPN "Veteran" Yogyakarta. Seminar ini sebagai wadah bagi para akademisi, peneliti, praktisi, serta pengguna Teknologi Informasi di Indonesia maupun di Dunia.

Sampai dengan batas waktu yang telah ditentukan, panitia menerima 130 (seratus tiga puluh) buah abstrak yang meliputi bidang *Computational*, kelompok *Intelligent System and Application*, kelompok *Information System and Application*, kelompok *Network and Security, Instrumentation and Robotic* dan kelompok *e-Democracy*. Komite Program akhirnya diputuskan untuk menerima sebanyak 120 (seratus dua puluh) buah makalah yang layak. Namun dari jumlah itu hanya 105 (seratus lima) buah makalah yang akhirnya dapat di presentasikan dalam SemnasIF 2009. Peserta SemnasIF 2009 berasal dari berbagai perguruan tinggi dan instansi di Indonesia.

Dalam kesempatan ini, Panitia SemnasIF 2009 ingin mengucapkan terima kasih yang sebesar-besarnya kepada Komite Program, Panitia Pelaksana, Pembicara Utama, Departemen Komunikasi dan Informatika, Komisi Pemilihan Umum, KPU Pusat, Harian Kompas, Para Sponsoring dan peserta yang telah bekerja keras, berpartisipasi serta memberikan dukungan sehingga acara ini dapat terlaksana.

Akhir kata panitia mengucapkan selamat datang bagi para Pemakalah dan Peserta di Kampus Universitas Pembangunan Nasional "Veteran" Condong Catur, Yogyakarta dan semoga Allah SWT selalu memberikan rahmat, hidayah dan perlindungan-Nya kepada kita semua.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 23 Mei 2009

Ketua Pelaksana
Frans Richard Kodong, ST, MKom

SUSUNAN PANITIA

Penanggung Jawab :

Ketua Jurusan Teknik Informatika

Ketua Pelaksana :

Frans Richard Kodong, ST, MKom.

Reviewer :

Ir. Lukito Edi Nugroho, M.Sc., Ph.D.

Dr. Ahmad Ashari

Dr. Azhari S.N., MT.

Surahyo, M.Eng.

Ir. Abdul Kadir, MT., MM.

Budi Santosa, S.Si., MT.

Herry Sofyan, ST., M.Kom.

Hafsah, S.Si., MT.

Komite Pelaksana (Informatika UPN) :

Wilis Kaswidjanti, S.Si., M.Kom.

Bambang Yuwono, ST., MT.

Juwairiah, S.Si., MT.

Hidayatullah Himawan, ST., MM.

Yuli Fauziah, ST., MT.

Novrido Charibaldi, S.Kom., M.Kom.

Dessyanto Boedi P, ST., MT.

Herlina Jayadianti, ST., MT.

Agus Sasmito A, S.Kom.

Awang Hendrianto Pratomo, ST, MT.

Heru Cahya Rustamaji, S.Si., MT.

Paryati, ST., M.Kom.

Nur Heri Cahyana, ST., M.Kom

Rahayu Ari O.

S. Haryanto

Sukardi

Budi Cahyono

Pri Wahyu Eko Setiawan

Partiman

Himpunan Mahasiswa Teknik Informatika

11	DETEKSI CITRA PORNOGRAFI MENGGUNAKAN TSL COLOR SPACE DAN NUDITY DETECTION ALGORITHM	<i>Sani Muhammad Isa, Febri Mariana</i>	A-86
12	MODEL DAN TEKNIK KALIBRASI CITRA UNTUK SISTEM AUTONOMOUS ROBOT	<i>Awang Hendrianto Pratomo, Mohd Shamudin Zakaria, Anton Satria Prabuwono</i>	A-94
13	OPTIMASI STRATEGI ALGORITMA GREEDY UNTUK MENYELESAIKAN PERMASALAHAN KNAPSACK 0-1	<i>Paryati</i>	A-101 ✓
14	IMPLEMENTASI ALGORITMA KOLONI SEMUT PADA PROSES Pencarian JALUR TERPENDEK JALAN PROTOKOL DI KOTA YOGYAKARTA	<i>Bambang Yuwono, Agus Sasmito Aribowo, Siswanto Budi Wardoyo</i>	A-111
15	SISTEM Pencarian CITRA DIGITAL MENGGUNAKAN CONTENT-BASED	<i>Herry Sofyan</i>	A-121
16	MODEL PENELUSURAN CITRA DIGITAL PADA DATABASE CITRA MENGGUNAKAN PENDEKATAN PERHITUNGAN KEDEKATAN POLA WARNA	<i>Agus Sasmito Aribowo</i>	A-132
17	CONTENT BASED IMAGE RETRIEVAL BERDASARKAN FITUR BENTUK MENGGUNAKAN METODE GRADIENT VECTOR FLOW SNAKE	<i>Ida Hastuti, Mochammad Hariadi, I Ketut Eddy Purnama</i>	A-140
18	KONVERSI WARNA RGB KE HLS MENGGUNAKAN C++	<i>Ina Agustina, Fauziah</i>	A-146
19	PEMBENTUKAN TRANSFORMASI OBJEK 2 DIMENSI DENGAN OPENGL	<i>Ina Agustina, Fauziah</i>	A-152

B. INSTRUMENTATION AND ROBOTIC

1	IMPLEMENTASI FPGA (FIELD PROGRAMMABLE GATE ARRAY) PADA SPREADING DAN DESPREADING MENGGUNAKAN KODE PSEUDONOISE KASAMI DAN JPL (JET PROPULSION LABORATORY)	<i>Anang Budikarso, Aries Pratiarso</i>	B-1
2	MANAJEMEN GRID UNTUK RENDER ANIMASI 3 DIMENSI	<i>Arthur Mourits Rumagit, Moch. Hariadi</i>	B-9
3	PERENCANAAN JALUR MOBILE ROBOT PADA LINGKUNGAN DINAMIS BERBASIS COMPACT GENETIC ALGORITHM	<i>Bima Sena Bayu Dewantara, Djoko Purwanto</i>	B-17
4	PENGATURAN KECEPATAN MOTOR INDUKSI TIGA FASA DENGAN METODA DIRECT TORQUE MENGGUNAKAN FUZZY LOGIC CONTROLLER	<i>Endro Wahjono, Soebagio</i>	B-24

OPTIMASI STRATEGI ALGORITMA GREEDY UNTUK MENYELESAIKAN PERMASALAHAN KNAPSACK 0-1

Paryati

Jurusan Teknik Informatika UPN "Veteran" Yogyakarta

Jl. Babarsari no 2 Tambakbayan 55281 Yogyakarta Telp (0274)-485323

e-mail : yaya_upn_cute@yahoo.com.

Abstract

Knapsack is a place used to store an object with same or less size with its capacity. The objects can be put at entirely or partially. This analysis using knapsack 0 or 1, that is an object taken entirely or not at all. As for the profit value is benefit value that possessed by every object. In this analysis, we always expect optimal profit value from an object. But, more object doesn't always mean more profit.

Keywords: algorithm greedy, knapsack problem, object, optimization.

1. PENDAHULUAN

Latar Belakang Masalah

Knapsack merupakan suatu kantong atau tempat yang digunakan untuk memuat sesuatu objek. Kantong atau tempat tersebut hanya dapat menyimpan 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. Salah satu contoh penyelesaiannya menggunakan strategi algoritma yang benar-benar dapat menghasilkan solusi yang optimal adalah Brute Force, tetapi strategi ini sangat tidak efisien.

Sehingga untuk menyelesaikan permasalahan knapsack 0-1 diperlukan suatu algoritma yang dapat menghasilkan solusi yang optimal, efektif dan efisien yaitu dengan menggunakan strategi algoritma greedy. Analisis persoalan knapsack 0-1 ini dikerjakan menggunakan strategi algoritma greedy. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang dapat memberikan perolehan yang terbaik yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan tujuan bahwa sisanya mengarah ke solusi optimum global.

Persoalan knapsack 0-1 supaya mendapatkan hasil yang optimal dapat diselesaikan dengan strategi algoritma greedy. Untuk mencari dan mendapatkan solusi yang optimal yaitu dengan cara menggunakan 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 kompleksitas pencarian. Berdasarkan algoritma diatas maka dapat dihitung kompleksitas waktu asimptotiknya adalah $O(n)$.

Tujuan Penelitian

1. Untuk dapat menghasilkan solusi yang optimal, efektif dan efisien dalam menyelesaikan permasalahan knapsack 0-1 dengan menggunakan strategi algoritma greedy.
2. Untuk menghirung kompleksitas waktu asimtotik algoritma greedy dalam permasalahan knapsack 0-1.

Manfaat Penelitian

Berdasarkan hasil penelitian ini persoalan knapsack 0-1 dapat diselesaikan dengan strategi algoritma greedy. Sedangkan untuk mencari dan mendapatkan solusi yang optimal dengan cara menggunakan strategi greedy by profit, atau greedy by weight, atau dapat juga diselesaikan dengan greedy by density. Analisis membuktikan bahwa algoritma greedy dapat mengurangi jumlah langkah kompleksitas pencarian. Adapun kompleksitas waktu asimptotik-nya adalah $O(n)$.

2. TINJAUAN PUSTAKA

Dasar Teori

Permasalahan Knapsack 0-1

Persoalan knapsack 0 atau 1 ini dikerjakan menggunakan strategi algoritma greedy. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang dapat memberikan perolehan terbaik yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan tujuan bahwa sisanya mengarah ke solusi optimum global (Springer V, 2005).

Algoritma Greedy

Algoritma greedy adalah algoritma untuk menyelesaikan permasalahan secara bertahap (Brassard G, 1996). Tahap penyelesaiannya adalah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan.
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Strategi Algoritma Greedy

Untuk memilih objek yang akan dimasukkan ke dalam knapsack terdapat beberapa strategi greedy yang heuristik (Silvano, 1990) yaitu:

a. Greedy by profit

Knapsack diisi dengan objek yang mempunyai keuntungan terbesar pada setiap tahap. Objek yang paling menguntungkan dipilih terlebih dahulu untuk memaksimalkan keuntungan. Tahap pertama yang dilakukan 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

Knapsack diisi dengan objek yang mempunyai berat paling ringan pada setiap tahap. Sebanyak mungkin objek dimasukkan ke dalam knapsack untuk memaksimalkan keuntungan. Tahap pertama yang dilakukan mengurutkan secara menaik objek-objek berdasarkan weight-nya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.

c. Greedy by density

Knapsack diisi dengan objek yang mempunyai densitas terbesar pada setiap tahap. Memilih objek yang mempunyai keuntungan per unit berat terbesar untuk memaksimalkan keuntungan. Tahap pertama yang dilakukan adalah mencari nilai profit per unit (density) dari tiap-tiap objek. Kemudian objek-objek tersebut diurutkan berdasarkan densitasnya. Kemudian baru diambil satu persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan. Algoritma greedy mengurangi jumlah langkah pencarian.

Kompleksitas

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

Pemrograman Bahasa C

Bahasa pemrograman yang digunakan untuk menyelesaikan riset ini menggunakan bahasa C dengan pertimbangan bahasa tersedia di semua jenis komputer, sifatnya portabel, proses executable program lebih cepat merupakan bahasa yang terstruktur, bahasa tingkat tinggi (Jogiyanto H, 1993).

Studi Pustaka

Pernah dilakukan penelitian dengan judul knapsack 0-1 problem diselesaikan menggunakan algoritma genetik. Dalam penelitian tersebut algoritma digunakan untuk menyeleksi barang-barang yang akan dimasukkan ke dalam kantong atau tempat sehingga didapat generasi terbaik yang optimum dari sejumlah generasi tersebut. Fungsi tujuan memuat barang yang ada tanpa melebihi batasan yang tersedia. Fungsi kendalanya adalah bobot dan volume. Kemudian dibentuk bilangan biner yaitu 1 yang bisa masuk sedangkan 0 tidak bisa masuk. Dengan roda rolet beberapa generasi dapat diseleksi yang terbaik, lalu di crossover dengan persilangan aritmetik, terus dilakukan mutasi sehingga didapat generasi yang terbaik, lalu di crossover dengan tersebut yang dimasukkan ke dalam kantong atau tempat sehingga didapat generasi yang terbaik. Generasi yang terbaik (Yuliani Indriyaningsih, 2004).

Perbedaan dari penelitian yang pernah dilakukan diatas dengan penelitian saya adalah persoalan knapsack 0-1 diselesaikan menggunakan algoritma greedy. Untuk mencari dan mendapatkan solusi yang optimal persoalan knapsack 0-1 dapat dilakukan dengan cara menggunakan strategi greedy by profit, atau greedy by weight, atau dapat juga diselesaikan dengan greedy by density. Hasil analisis terbukti bahwa algoritma greedy dapat mengurangi jumlah langkah kompleksitas pencarian. Berdasarkan algoritma diatas maka dapat dihitung kompleksitas waktu asimptotik-nya adalah $O(n)$.

3. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah metodologi pengembangan sistem dengan metode *water fall* (Pressman R, 2000) yaitu sebagai berikut:

- 1) Analisis dan rekayasa sistem, merupakan tahap menganalisis hal-hal yang diperlukan dalam pelaksanaan proyek pembuatan atau pengembangan *software*.
- 2) Analisis kebutuhan perangkat lunak, merupakan tahap menganalisis perangkat lunak yang dibutuhkan untuk pengembangan dan implementasi.
- 3) Desain atau perancangan, merupakan tahap penerjemah dari keperluan atau data yang telah dianalisis dalam bentuk yang mudah dimengerti oleh pemakai (*user*).
- 4) Kode atau *coding*, merupakan tahap penerjemahan data atau pemecahan masalah yang telah dirancang ke dalam bahasa pemrograman komputer yang telah ditentukan.
- 5) *Testing* atau tahap pengujian, merupakan tahap pengembangan program dan uji coba terhadap program tersebut.
- 6) Pemeliharaan, meliputi kegiatan-kegiatan koreksi kesalahan dan penyesuaian perangkat lunak terhadap perubahan lingkungan.

Dalam penelitian ini, metodologi pengembangan sistem yang dilakukan hanya sampai pada tahap *testing* dan pengujian.

Analisis dan Perancangan

Analisis Kasus

Correctness

Data awal :

$w_1 = 2; p_1 = 12$ $w_2 = 5; p_2 = 15$ $w_3 = 10; p_3 = 50$ $w_4 = 5; p_4 = 10$

Kapasitas knapsack $W = 16$

Greedy by profit

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.

Tabel 1. Greedy by profit

Properti objek				
I	W _i	p _i	p _i /w _i	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 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.

Tabel 2. Greedy by weight

Properti objek				
I	W _i	p _i	p _i /w _i	Status
2	5	15	3	Diambil
4	5	10	2	Diambil
1	6	12	2	Diambil
3	10	50	5	Tidak

Greedy by density

Pertama kali yang dilakukan adalah program mencari nilai profit per unit (density) dari tiap-tiap objek. Kemudian objek-objek tersebut diurutkan berdasarkan density-nya. Kemudian baru diambil satu-persatu objek yang dapat ditampung oleh knapsack sampai knapsack penuh atau sudah tidak ada objek lagi yang bisa dimasukkan.

Tabel 3. Greedy by density

Properti 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

Maka hasil akhir program mendapatkan data seperti di bawah ini:

Tabel 4. Hasil akhir program

Properti objek				Greedy by		
I	wi	pi	pi/wi	profit	Weight	density
1	6	12	2	0	1	0
2	5	15	3	1	1	1
3	10	50	5	1	0	1
4	5	10	2	0	1	0
Total bobot				15	16	15
Total keuntungan				65	37	65

Hasil Analisis

Efficiency 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 algoritma diatas maka dapat dihitung kompleksitas waktu asimptotik-nya adalah $O(n)$.

Efficiency 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 asimptotik-nya adalah $O(n)$

Efficiency 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 berdasarkan
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 algoritma diatas maka dapat dihitung kompleksitas waktu asimptotik-nya adalah $O(n)$.

Perancangan Program

Listing Program Knapsack Problem with greedy strategy

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void urutkanbyprofit(int no[],int weight[],int profit[]);
void urutkanbyweight(int no[],int weight[],int profit[]);
void urutkanbydensity(int no[],int weight[],int profit[],int density[]);
int knapsack;
typedef struct{
int w;
int p;
int d;
char s[20];
int i;
}tabel;
tabel tabel1[4],tabel2[4],tabel3[4],tabel4[4];
void main()
{
int j,k[4],l[4],m[4],o[4];
int acc=0;
for(j=0;j<4;j++)
{
printf("Masukkan nilai w%d -"j+1),scanf("%d",&tabel1[j].w);
printf("Masukkan nilai p%d -"j+1),scanf("%d",&tabel1[j].p);
tabel1[j].d=tabel1[j].p/tabel1[j].w;
tabel1[j].i=j+1;
```

```
}
printf("Kapasitas Knapsack W:");scanf("%d",&knapsack);
printf("i   wi   pi   pi/wi   status\n");
for(j=0;j<4;j++)
{
printf("%d   ",tabel1[j].i);
printf("%d   ",tabel1[j].w);
printf("%d   ",tabel1[j].p);
tabel1[j].d=tabel1[j].p/tabel1[j].w;
printf("%d\n",tabel1[j].d);
}
/*Greedybyprofit*/
acc=0;
printf("\nGreedy By Profit");
for(j=0;j<4;j++)
{
k[j]=tabel1[j].i;
l[j]=tabel1[j].w;
m[j]=tabel1[j].p;
}
urutkanbyprofit(k,l,m);
for(j=0;j<4;j++)
{
tabel2[j].i=k[j];
tabel2[j].w=l[j];
tabel2[j].p=m[j];
}
for(j=0;j<4;j++)
{
acc=acc+tabel2[j].w;
if(acc<=knapsack)
strcpy(tabel2[j].s,"Diambil");
else
strcpy(tabel2[j].s,"Tidak diambil");
}
/*Hasil*/
printf("\ni   wi   pi   pi/wi   status\n");
for(j=0;j<4;j++)
{
printf("%d   ",tabel2[j].i);
printf("%d   ",tabel2[j].w);
printf("%d   ",tabel2[j].p);
tabel2[j].d=tabel2[j].p/tabel2[j].w;
printf("%d   ",tabel2[j].d);
printf("%s\n",tabel2[j].s);
}
/*end Greedybyprofit*/
/*Greedybyweight*/
acc=0;
printf("\nGreedy By weight\n");
for(j=0;j<4;j++)
{
k[j]=tabel1[j].i;
l[j]=tabel1[j].w;
m[j]=tabel1[j].p;
}
```

```
}
urutkanbyweight(k,l,m);
for(j=0;j<4;j++)
{
tabel3[j].i=k[j];
tabel3[j].w=l[j];
tabel3[j].p=m[j];
}
for(j=0;j<4;j++)
{
acc=acc+tabel3[j].w;
if(acc<=knapsack)
strcpy(tabel3[j].s,"Diambil");
else
strcpy(tabel3[j].s,"Tidak diambil");
}
/*Hasil*/
printf("\ni    wi    pi    pi/wi    status\n");
for(j=0;j<4;j++)
{
printf("%d    ",tabel3[j].i);
printf("%d    ",tabel3[j].w);
printf("%d    ",tabel3[j].p);
tabel3[j].d=tabel3[j].p/tabel3[j].w;
printf("%d    ",tabel3[j].d);
printf("%s\n",tabel3[j].s);
}
/*end Greedybyweight*/
```

```
/*Greedybydensity*/
acc=0;
printf("\nGreedy By density\n");
for(j=0;j<4;j++)
{
k[j]=tabel1[j].i;
l[j]=tabel1[j].w;
m[j]=tabel1[j].p;
n[j]=tabel1[j].d;
}
urutkanbydensity(k,l,m,n);
for(j=0;j<4;j++)
{
tabel4[j].i=k[j];
tabel4[j].w=l[j];
tabel4[j].p=m[j];
tabel4[j].d=n[j];
}
for(j=0;j<4;j++)
{
acc=acc+tabel4[j].w;
if(acc<=knapsack)
strcpy(tabel4[j].s,"Diambil");
else
strcpy(tabel4[j].s,"Tidak diambil");
}
}
```

```
/*Hasil*/
printf("ni   wi   pi   pi/wi   status\n");
for(j=0;j<4;j++)
{
printf("%d   ",tabel4[j].i);
printf("%d   ",tabel4[j].w);
printf("%d   ",tabel4[j].p);
printf("%d   ",tabel4[j].d);
printf("%s\n",tabel4[j].s);
}
/*end Greedybydensity*/
getche();
}
void urutkanbyprofit(int no[],int weight[],int profit[])
{
int i,j,temp;
for(i=0;i<4;i++)
for(j=0;j<4;j++)
if(profit[i]>profit[j])
{
temp=no[i];
no[i]=no[j];
no[j]=temp;
temp=weight[i];
weight[i]=weight[j];
weight[j]=temp;
temp=profit[i];
profit[i]=profit[j];
profit[j]=temp;
}
}
void urutkanbyweight(int no[],int weight[],int profit[])
{
int i,j,temp;
for(i=0;i<4;i++)
for(j=0;j<4;j++)
if(weight[i]<weight[j])
{
temp=no[i];
no[i]=no[j];
no[j]=temp;
temp=weight[i];
weight[i]=weight[j];
weight[j]=temp;
temp=profit[i];
profit[i]=profit[j];
profit[j]=temp;
}
}
void urutkanbydensity(int no[],int weight[],int profit[],int density[])
{
int i,j,temp;
for(i=0;i<4;i++)
for(j=0;j<4;j++)
if(density[i]>=density[j])
```

```
{
temp=no[i];
no[i]=no[j];
no[j]=temp;
temp=weight[i];
weight[i]=weight[j];
weight[j]=temp;
temp=profit[i];
profit[i]=profit[j];
profit[j]=temp;
temp=density[i];
density[i]=density[j];
density[j]=temp;
}
```

4. HASIL DAN PEMBAHASAN

Berdasarkan analisis diatas dilakukan ujicoba dan testing terhadap program yang sudah dibuat. Dalam penelitian ini hanya dicantumkan hasil uji coba yang ke-1 dan uji coba yang ke-2 sedangkan hasil uji coba yang lain tidak disertakan. Adapun hasil uji coba dapat dilihat di bawah ini.

Uji coba 1

```
Masukkan nilai w1 -5
Masukkan nilai p1 -4
Masukkan nilai w2 -5
Masukkan nilai p2 -7
Masukkan nilai w3 -8
Masukkan nilai p3 -3
Masukkan nilai w4 -4
Masukkan nilai p4 -1
Kapasitas Knapsack W:15
```

i	wi	pi	pi/wi	status
1	5	4	0	
2	5	7	1	
3	8	3	0	
4	4	1	0	

Greedy By Profit

i	wi	pi	pi/wi	status
2	5	7	1	Diambil
1	5	4	0	Diambil
3	8	3	0	Tidak diambil
4	4	1	0	Tidak diambil

Greedy By weight

i	wi	pi	pi/wi	status
4	4	1	0	Diambil
1	5	4	0	Diambil
2	5	7	1	Diambil
3	8	3	0	Tidak diambil

Greedy By density

i	wi	pi	pi/wi	status
2	5	7	1	Diambil
1	5	4	0	Diambil
3	8	3	0	Tidak diambil
4	4	1	0	Tidak diambil

Uji coba 2

Masukkan nilai w1 =4
 Masukkan nilai p1 =2
 Masukkan nilai w2 =2
 Masukkan nilai p2 =5
 Masukkan nilai w3 =9
 Masukkan nilai p3 =1
 Masukkan nilai w4 =4
 Masukkan nilai p4 =6
 Kapasitas Knapsack W:10

i	wi	pi	pi/wi	status
1	4	2	0.5	
2	2	5	2.5	
3	9	1	0.11	
4	4	6	1.5	

Greedy By Profit

i	wi	pi	pi/wi	status
4	4	6	1.5	Diambil
2	2	5	2.5	Diambil
1	4	2	0.5	Diambil
3	9	1	0.11	Tidak diambil

Greedy By weight

i	wi	pi	pi/wi	status
2	2	5	2.5	Diambil
1	4	2	0.5	Diambil
4	4	6	1.5	Diambil
3	9	1	0.11	Tidak diambil

Greedy By density

i	wi	pi	pi/wi	status
2	2	5	2.5	Diambil
4	4	6	1.5	Diambil
3	9	1	0.11	Tidak diambil
1	4	2	0.5	Tidak diambil

5. KESIMPULAN

Knapsack problem with greedy strategy ini menjelaskan bagaimana persoalan knapsack dapat diselesaikan dengan strategi greedy, untuk mencari dan mendapatkan solusi optimal yaitu dengan menggunakan strategi greedy by profit, atau greedy by weight, atau dapat juga diselesaikan dengan greedy by density. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan langkah dengan tujuan bahwa sisanya mengarah ke solusi optimum lokal pada setiap langkah.

Algoritma greedy dapat mengurangi jumlah langkah global. Algoritma diatas maka dapat dihitung kompleksitas waktu asimptotik-nya adalah $O(n)$. Berdasarkan

6. DAFTAR PUSTAKA

Brassard G, 1996, *Fundamentals of algorithmics*, Prentice-Hall, New Jersey
 Ding-Zhu Du, 2000, *Theory of Computational Complexity*, John Wiley & Son
 Jogiyanto H, 1993, *Konsep Dasar Pemrograman Bahasa C*, Andi Offset, Yogyakarta
 Pressman R, 2002, *Software Engineering*, Prentice-Hall, Inc., New Jersey
 Springer V, 2005, *Knapsack 0-1 Problem*, John Wiley & Sons, ISBN : 3-540-40286-1
 Silvano et al, 1990, *Knapsack problem : Algorithm and Computer Implementation*, John Wiley & Sons, ISBN : 0-471-92420
 Yuliani I, 2004, *Knapsack 0-1 Problem Diselesaikan Menggunakan Algoritma Genetic*, Tesis S-2 UGM