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 mengunakan 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 komplesitas 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 mengunakan 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 komplesitas 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 memaksimumkan 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 memaksimumkan 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 memaksimumkan keuntungan. Tahap pertama yang dilakukan adalah mencari nilai profit perunit (density) dari tiap-tiap objek. Kemudian objek-objek tersebut diurutkan berdasarkan densitynya. 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 samakarena 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 $Ø=(f(n))$ 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 genetik 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 didapatkan generasi yang terbaik. Generasi yang terbaik tersebut yang dimasukkan ke dalam kantong atau tempat sehingga tidak akan melebihi kapasitas yang ada (Yuliani Indriyaningsih, 2004).

Perbedaan dari penelitian yang pernah dilkakukan 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 mengunakan 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 komplesitas 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 :

w1 = 2; p1 = 12 w2 = 5; p1 = 15 w3 = 10; p1 = 50 w4 = 5; p1 = 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	**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 yang dilakukan adalah program mengurutkan secara menaik objek-objekberdasarkan 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	**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

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 komplesitas 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 komplesitas 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 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 algoritma diatas maka dapat dihitung komplesitas 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],n[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
3        8        3        0        Diambil
4        4        1        0        Tidak diambil
1        5        4        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
2        2          5          2
3        9          1          0
4        4          6          1

Greedy By Profit
i       wi          pi          pi/wi       status
4        4          6          1          Diambil
2        2          5          2          Diambil
1        4          2          0          Diambil
3        9          1          0          Tidak diambil

Greedy By weight

i       wi          pi          pi/wi       status
2        2          5          2          Diambil
1        4          2          0          Diambil
4        4          6          1          Diambil
3        9          1          0          Tidak diambil

Greedy By density

i       wi          pi          pi/wi       status
2        2          5          2          Diambil
4        4          6          1          Diambil
3        9          1          0          Tidak diambil
1        4          2          0          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 mengunakan 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 yang dapat memberikan perolehan terbaik yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan tujuan bahwa sisanya mengarah ke solusi optimum global.

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

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 Enfineering*, 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, K*napsack 0-1 Problem Diselesaikan Menggunakan Algoritma Genetic, Tesis S-2 UGM*