MINGGU KE- 13

METODE GREEDY

METODE GREEDY

- Greedy diambil dari bahasa inggris berarti rakus, tamak, loba, serakah.
- Prinsip greedy: "**Take What You Can Get Now!".**
- Algoritma greedy membentuk solusi langkah perlangkah (step by step).

strategi pencarian untuk masalah optimasi berbasis prinsip: pada setiap tahap, pilih solusi paling baik. Dengan harapan, semua tahapan ini akan menemukan solusi terbaik untuk masalah tersebut. Algoritma greedy termasuk sederhana dan tidak rumit (Santosa and Ai, 2017).

Greedy

METODE GREEDY (Lanjutan)

Untuk mendapatkan solusi optimal dari permasalahan yang mempunyai dua kriteria yaitu:
1. Fungsi Tujuan/Utama
2. Nilai pembatas (constrain)

Proses Kerja Metode Greedy:
Untuk menyelesaikan suatu permasalahan dengan n input data yang terdiri dari beberapa fungsi pembatas & 1 fungsi tujuan yang diselesaikan dengan memilih beberapa solusi yang mungkin (feasible solution/feasible sets), yaitu bila telah memenuhi fungsi tujuan/obyektif.

METODE GREEDY (Lanjutan)

Persoalan Optimasi: **(Masalah Penukaran Uang)**:
Diberikan uang senilai A. Tukar A dengan koin-koin uang yang ada. Berapa jumlah minimum koin yang diperlukan untuk penukaran tersebut?

Contoh 1: tersedia banyak koin 1, 5, 10, 25
Uang senilai A = 32 dapat ditukar dengan banyak cara berikut:

32 = 1 + 1 + … + 1 (32 koin)
32 = 5 + 5 + 5 + 5 + 10 + 1 + 1 (7 koin)
32 = 10 + 10 + 10 + 1 + 1 (5 koin)
… dst

Minimum: 32 = 25 + 5 + 1 + 1 (4 koin)

METODE GREEDY (Lanjutan)

1. Optimal On Tape Storage Problem

Metode GREEDY digunakan dalam penyelesaian masalah-masalah

2. Knapsack Problem

METODE GREEDY (Lanjutan)

1. Optimal On Tape Storage Problem

Permasalahan bagaimana mengoptimalisasi storage/ memory dalam komputer agar data yg disimpan dapat termuat dengan optimal.

Misalkan terdapat n program yang akan disimpan didalam pita (tape). Pita tersebut mempunyai panjang maksimal sebesar L, masing-masing program yang akan disimpan mempunyai panjang L1,L2,L3...,Ln. Cara penyimpanan adalah **terurut** (**sequential**).

Optimal On Tape Storage Problem (Lanjutan)

Penerapan dari *Optimal On Tape Storage Problem adalah:*

- *Terdapat pada Pita Kaset*
- *Media penyimpanan pada abad 19*
- *Sebelum era digitalisasi pada abad 20*

Kriteria Greedy pada *Optimal On Tape Storage Problem*:

- *Fungsi tujuan: Optimal Storage* = $D(I) = \sum_{j=1}^{n} \sum_{k=1}^{j} l_{ik}$
- *Fungsi Pembatas : Mean Retrieval Time (MRT)* = $\sum_{j=1}^{n} t_j / n$

Optimal On Tape Storage Problem (Lanjutan)

Studi kasus:

Penyimpanan pada pita kaset terdapat 3 file lagu dengan durasi waktu 5 menit,10 menit, 3 menit). Tentukan urutannya agar dapat menghemat media penyimpanannya?

Penyelesaiannya:

1. Menemukan 2 kriteria greedy
   Fungsi tujuan: optimalisasi media penyimpanan
   Fungsi pembatas : waktu akses file (Mean Retrieval Time)

2. Mencari Feasible Solution
   Alternatif solusi yang dapat digunakan untuk memperoleh optimal solution

Optimal On Tape Storage Problem (Lanjutan)

Jumlah Feasible Solution untuk 3 buah file input adalah:
N Faktorial dimana N: Jumlah File
3!=3x2x1=6

3. Menghitung Fungsi Tujuan & Fungsi Pembatas

Ordering	Panjang	D (I)	MRT
1,2,3	5,10,3	5 + (5+10) + (5+10+3) = 38	38/3=12,66
1,3,2	5,3,10	5 + (5+3) + (5+3+10) = 31	31/3=10,33
2,1,3	10,5,3	10 + (10+5) + (10+5+3) = 43	43/3=14,33
2,3,1	10,3,5	10 + (10+3) + (10+3+5) = 41	41/3=13,66
3,1,2	3,5,10	3 + (3+5) + (3+5+10) = 29	29/3=9,66
3,2,1	3,10,5	3 + (3+10) + (3+10+5) = 34	34/3=11,33

Optimal On Tape Storage Problem (Lanjutan)

$(L_1,L_2,L_3) = (5,10,3)$

Dari tabel tersebut, didapat susunan/order yang optimal,sbb :

1. susunan pertama untuk program ke tiga
2. susunan kedua untuk program kesatu
3. susunan ketiga untuk program kedua

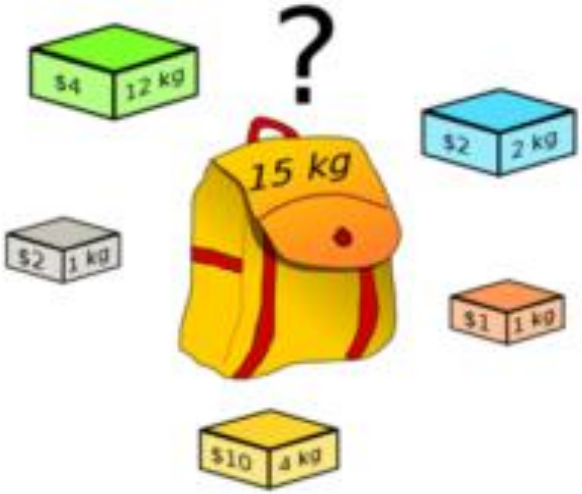
Kunci dari permasalahan Optimal On Tape Storage Problem adalah Susunan File dari ukuran Kecil Kebesar (Increasing)

METODE GREEDY (Lanjutan)

2. KNAPSACK Problem

- Knapsack adalah tas atau karung
- Karung digunakan memuat objek, tentunya tidak semua objek dapat ditampung di dalam karung.
- Karung hanya dapat menyimpan beberapa objek dengan total ukurannya (*weight*) lebih kecil atau sama dengan ukuran kapasitas karung.

Ilustrasi Knapsack Problem



Gambar ilustrasi terdapat tas berkapasitas 15kg, ada 5 barang dengan berat dan keuntungannya masing-masing. Persoalannya adalah barang mana saja yang harus dimasukan ke dalam tas (Aristi, 2015)..

KNAPSACK Problem (Lanjutan)

Studi Kasus:

- Terdapat n obyek (Xi;i=1,2,3,....n)
- Masing-masing mempunyai berat (weight)/Wi
- Masing-masing memiliki nilai (profit)/Pi yang berbeda-beda.

KNAPSACK Problem (Lanjutan)

Masalah KNAPSACK Problem

Bagaimana obyek-obyek tersebut dimuat/dimasukan kedalam ransel (*knapsack*) yang mempunyai kapasitas max=M.

Sehingga timbul permasalahan sbb:

- Bagaimana memilih obyek yang akan dimuat dari *n* obyek yang ada sehingga nilai obyek termuat jumlahnya sesuai dengan kapasitas($\leq$ M)
- Jika semua obyek harus dimuat ke dalam ransel maka **berapa bagian** dari setiap obyek yang ada dapat dimuat ke dalam ransel sedemikian sehingga nilai kumulatif max & sesuai dengan kapasitas ransel ?

KNAPSACK Problem (Lanjutan)

1. Secara Matematika

2. Kriteria Greedy

Penyelesaian Knapsack Problem dapat dilakukan dengan:

3. Algoritma Pemrograman Greedy

KNAPSACK Problem (Lanjutan)

1. Penyelesaian Knapsack Secara Matematika

Fungsi tujuan = fungsi utama/obyektif

Fungsi yang menjadi penyelesaian permasalahan dengan mendapatkan solusi yang optimal.

Solusi dimaksud = menemukan nilai/profit yangg maksimal untuk jumlah obyek yang dimuat dalam ransel sehingga sesuai kapasitas.

$$\text{Fungsi Tujuan Maksimum} : \sum_{i=1}^{n} Pi\ Xi$$

Penyelesaian Knapsack Secara Matematika (Lanjutan)

Fungsi pembatas = fungsi *subyektif*

Fungsi yang bertujuan untuk memberikan batas maksimal dari setiap obyek untuk dapat dimuat dalam ransel sehingga kapasitasnya tidak melebihi dari jumlah maksimal daya tampung ransel.

$$\textbf{Fungsi Pembatas : } \sum_{i=1}^{n} \textbf{Wi Xi} \leq \textbf{M}$$

dimana : $0 \leq Xi \leq 1$; $Pi > 0$;$Wi > 0$

Catatan : karena dengan menggunakan Matematika sangat sulit dan rumit maka tidak dibahas lebih mendalam.

KNAPSACK Problem (Lanjutan)

2. Penyelesaian Dengan Kriteria Greedy

Konsep dari kriteria yang ditawarkan oleh metode Greedy yaitu :

a. Pilih obyek (barang) dengan nilai Pi maximal atau terbesar

b. Pilih obyek (barang) dengan berat Wi minimal dahulu.

c. Pilih obyek (barang) dgn perbandingan nilai & berat yaitu Pi/Wi yang terbesar.

Penyelesaian Dengan Kriteria Greedy (Lanjutan)

Contoh:

Diketahui bahwa kapasitas M = 20kg
Dengan jumlah barang n=3
Berat Wi masing-masing barang
(W1, W2, W3) = (18, 15, 10)
Nilai Pi masing-masing barang
(P1, P2, P3) = (25, 24, 15)

Penyelesaian Dengan Kriteria Greedy (Lanjutan)

Penyelesaian Soal Kriteria Greedy (W_1, W_2, W_3) = (18, 15, 10)
Pilih barang dengan Nilai Profit Maksimal (P_1, P_2, P_3) = (25, 24, 15)

- P1 = 25 → X1 = 1, dimisalkan sebagai atas atas nilai
- P2 = 24 → X2 = 2/15, dihitung dengan Fungsi Pembatas
- P3 = 15 → X3 = 0, dimisalkan sebagai batas bawah nilai

Penyelesaiannya:

- Barang nilai profit terbesar adalah barang ke-1, maka yang pertama kali dipilih adalah barang ke-1 sebanyak X_1=1
- Setelah barang ke-1 terpilih, maka sisa kapasitas ransel adalah

  20 kg-18 kg= 2kg.
- Kemudian pilih barang ke-2 sebanyak X_2=sisa kapasitas ransel/W_2=2/15
- Setelah barang ke-2 terpilih, sisa kapasitas ransel = 0Kg, sehingga barang ke-3 tidak terpilih → X_3=0
- **(X_1,X_2,X_3) = (1, 2/15, 0) adalah feasible solution**

Penyelesaian Dengan Kriteria Greedy (Lanjutan)

Penyelesaian Soal Kriteria Greedy

$(W_1, W_2, W_3) = (18, 15, 10)$
$(P_1, P_2, P_3) = (25, 24, 15)$

Pilih barang dengan Berat Minimal

- W1 = 18 → X1 = 0, sebagai batas bawah
- W2 = 15 → X2 = 2/3,dihitung dgn Fungsi Pembatas
- W3 = 10 → X3 = 1, sebagai batas atas

Penyelesaiannya:

- Barang dengan berat terkecil adalah barang ke-3, maka yang terpilih adalah barang ke-3 sebanyak X_3=1
- Setelah barang ke-3 terpilih, maka sisa kapasitas ransel adalah 20kg-10kg= 10kg
- Pilih barang ke-2 sebanyak X_2= Sisa Kapasitas ransel/W_2 = 10/15=2/3
- Setelah barang ke-2 terpilih, sisa kapasitas ransel adalah 0 Kg, artinya barang ke-1 tidak terpilih → X_1=0
- **$(X_1,X_2,X_3) = (0, 2/3, 1)$ adalah feasible solution**

Penyelesaian Dengan Kriteria Greedy (Lanjutan)

$(W_1, W_2, W_3) = (18, 15, 10)$
$(P_1, P_2, P_3) = (25, 24, 15)$

Pilih barang dengan menghitung perbandingan yang terbesar dari Profit dibagi Berat (Pi/Wi) yang diurut secara tidak naik, yaitu :

- P1/W1 = 25/18 =1.38→ karena terkecil maka X1 = 0
- P2/W2 = 24/15 =1.6→ karena terbesar maka X2 = 1
- P3/W3 = 15/10 =1.5→dengan Fungsi pembatas X3 = 1/2

Penyelesaiannya:

- Barang dengan (P_i/W_i) terbesar adalah barang ke-2, maka yang pertama kali dipilih adalah barang ke-2 sebanyak X_2=1
- Setelah barang ke-2 terpilih, maka sisa kapasitas ransel adalah 20kg - 15kg= 5kg
- Kemudian pilih barang ke-3 sebanyak X_3=sisa kapasitas ransel/W_3 =5/10=1/2
- Setelah barang ke-3 terpilih, sisa kapasitas ransel adalah 0kg, maka barang ke-1 tidak terpilih→ X_1=0
- **$(X_1,X_2,X_3) = (0, 1, 1/2)$ adalah feasible solution**

Penyelesaian Dengan Kriteria Greedy (Lanjutan)

Dibuatkan tabel berdasarkan elemen dr ke-3 kriteria metode Greedy

Solusi ke	(X_1,X_2,X_3)	$\sum W_iX_i$	$\sum P_iX_i$
Pi Max	(1, 2/15, 0)	20	(25.1) + (24.(2/15)) + (15.0) = 28.2
Wi Min	(0, 2/3, 1)	20	(25.0)+(24.(2/3)) +(15.1) = 31
Pi/Wi max	(0, 1, 1/2)	20	(25.0) + (24.1) + (15.(1/2)) = **31.5**

Nilai profit maksimal = 31.5 dengan komposisi yang sama

Penyelesaian Knapsack Problem (Lanjutan)

3. Penyelesaian Algoritma Pemrograman Greedy

Algoritma GREEDY KNAPSACK

```
PROCEDURE GREEDY KNAPSACK (P, W, X, n)
REAL P(1:n), W(1:n), X(1:n), M, isi
INTEGER i, n
X(1:n) = 0
isi = M
FOR i = 1 TO n DO
        IF W(i) > isi THEN EXIT ENDIF
        X(i) = 1
        isi = isi – W(i)
REPEAT
IF i ≤ n THEN X(i) = isi/W(i) ENDIF
END GREEDY KNAPSACK
```

Keterangan:
n = Jumlah objek
Wi = Bobot setiap objek
Pi = Profit setiap objek
Xi = Probabilitas setiap objek
M = Kapasitas media penyimpanan

Algoritma Pemrograman Greedy (Lanjutan)

Efektif jika data (Pi/Wi) disusun secara tidak naik lebih dahulu.

Penyelesaiannya :

Dengan Algoritma Pemrograman Greedy.

Diket. bhw kapasitas **M = 20kg,** degan jumlah barang **n=3**

Berat Wi masing² barang = (W1, W2, W3) = (18, 15, 10)

Nilai Pi masing² barang = (P1, P2, P3) = (25, 24, 15)

Lakukan pengurutan secara tdk naik terhadap hasil Pi/Wi, misalnya :

P1/Wi → 25/18 = 1,39 menjadi urutan ke 3

P2/W2 → 24/15 = 1,60 menjadi urutan ke 1

P3/W3 → 15/10 = 1.50 menjadi urutan ke 2

Sehingga menghasilkan pola urutan data yg baru,yaitu

W1,W2,W3 → 15, 10, 18 dan

P1,P2,P3 → 24, 15, 25

Algoritma Pemrograman Greedy (Lanjutan)

Lalu data² tsb diinputk' pd Alg. Greedy, terjadi proses :

x(1:n) ← 0 ; isi ← 20 ; i = 1

W(i) > isi ? → 15 > 20 ? → kondisi SALAH

x(1) = 1 → b'arti bhw brg tsb dpt dimuat seluruhnya.

Isi = 20 - 15 →kapasitas ransel b'kurang dengan sisa 5kg i =2

W(2) > isi ?? → 10 > 5 ?? → kondisi BENAR

x(2)=5/10=1/2→benda 10kg hanya dpt dimuat 1/2 bagian yaitu 5 kg.

i=3

Endif → diakhiri krn ransel sdh penuh (max =20kg)

Profit nilai yang didapat adalah : P1 + P2 + P3 yaitu:

24.1+ 15.1/2 + 18.0 = 24 + 7.5 = 31.5

Algoritma Pemrograman Greedy (Lanjutan)

Penyelesaiannya:

x(1:n) ← 0 ; isi ← 20 ; i = 1

FOR i ← 1 TO 3

W1,W2,W3 → 15, 10, 18
dan
P1, P2, P3 → 24, 15, 25

Saat i=1 Apakah W[1] > isi? → 15 > 20?

x[1] ← 1 barang dapat dimuat seluruhnya

isi = 20 – 15 = 5 sisa kapasitas 5kg

Saat i=2 Apakah W[2] > isi? → 10 > 5? → exit

Apakah i ≤ n? → 2 ≤ 3?

x[2] = isi/W[2] = 5/10 = 1/2 benda 10kg dimuat ½ bag = 5

ENDIF diakhiri karena ransel sudah penuh (max =20kg)

Profit nilai : P1 + P2 + P3 yaitu:

24(1)+ 15(1/2) + 18(0) = 24 + 7,5 = 31,5

Penyelesaian Knapsack Problem Menggunakan Python

Kodingan Program 1

```
#Program Penyelesaian algoritma geedy pada knapsack
def fractional_knapsack(value, weight, capacity):
    index = list(range(len(value)))
    ratio = [v/w for v, w in zip(value, weight)]
    index.sort(key=lambda i: ratio[i], reverse=True)

    max_value = 0
    fractions = [0]*len(value)
    for i in index:
        if weight[i] <= capacity:
            fractions[i] = 1
            max_value += value[i]
            capacity -= weight[i]
        else:
            fractions[i] = capacity/weight[i]
            max_value += value[i]*capacity/weight[i]
            break

    return max_value, fractions

n = int(input('Enter number of items: '))
value = input('Enter the values of the {} item(s) in order: '
              .format(n)).split()
```

Penyelesaian Knapsack Problem Menggunakan Python

Kodingan Program lanjutan

```
value = [int(v) for v in value]
weight = input('Enter the positive weights of the {} item(s) in order: '
               .format(n)).split()
weight = [int(w) for w in weight]
capacity = int(input('Enter maximum weight: '))

max_value, fractions = fractional_knapsack(value, weight, capacity)
print('The maximum value of items that can be carried:', max_value)
print('The fractions in which the items should be taken:', fractions)
```

Masukan data :

M = 20, n=3
(W1, W2, W3) = (18, 15, 10)
(P1, P2, P3) = (25, 24, 15)

Hasil Program:

Enter number of items:
Enter the values of the 3 item(s) in order:
Enter the positive weights of the 3 item(s) in order:
Enter maximum weight:

Outputnya :

```
Enter number of items: 3

Enter the values of the 3 item(s) in order: 25 24 15

Enter the positive weights of the 3 item(s) in order: 18 15 10

Enter maximum weight: 20
The maximum value of items that can be carried: 31.5
The fractions in which the items should be taken: [0, 1, 0.5]
```

Kesimpulan Knapsack Problem

- Cara matematika dianggap lebih rumit dan tidak cocok untuk digunakan, karena harus memperhatikan nilai probabilitas setiap item, nilai ini merupakan faktor penentu mengingat nilai probabilitas (Xi) $0 \leq Xi \leq 1$. Kisaran nilai-nilai Xi di sini sangat luas, bisa 0, 0,1, 0,01, 0,001, ... 1.

- Cara kriteria greedy dianggap lebih mudah dan lebih optimal dibanding cara yang lain meskipun kekurangannya harus mengerjakan beberapa tahapan terlebih dahulu.

- Cara algoritma greedy lebih cepat penyelesaiannya namun harus tahu algoritma dan harus paham cara penterjemahan algoritma tersebut. Selain itu teknik ini akan efektif jika objek disusun secara tidak naik terlebih dahulu berdasarkan nilai Pi/Wi.