**Makalah Algoritma Greedy**

**Kelas : 3IA11**

**Nama Anggota Kelompok**

**Almira Rahma Ladiba (50412658)**

**Firsty Avisha S (52412993)**

**Nabila Adilah(55412195)**

**Nia Ita Mora (55412298)**

**Stefani Hana Leonita(57412140)**

**Venelia Suci Febrina (57412553)**

**UNIVERSITAS GUNADARMA**

**2014**

**Algoritma Greedy**

**Pengertian Algoritma Greedy**

Algoritma Greedy adalah salah satu jenis algoritma, algoritma greedy menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara dalam setiap langkahnya atau local maxium.

Algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang cukup cepat. Metode ini banyak digunakan dalam berbagai penyelesaian masalah, antara lain adalah :

1. **Optimal Storage on Tapes Problem**

* Bagaimana mengoptimalkan penyimpanan, agar data yang disimpan dapat termuat dengan optimal.
* Bagaimana susunan yang harus dibentuk

**Proses pemecahannya**

1. Tentukan nilai panjang data (L), waktu penyimpanan (t) dan waktu rata-rata (MRT) jika ada
2. Tentukan urutan penyimpanan datanya
3. Hitung total penyimpanan (**jika ada kapasitas media penyimpanan maka total tidak boleh melebihi**)
4. Pilih total yang minimum (sesuai dengan unsur efisiensi dan efektifitas)

**Contoh 1:**

Diketahui 3 program yang akan disimpan dalam media penyimpanan dengan panjang masing-masing 5, 10, dan 3. Bagaimana proses penyimpanan yang optimal dengan metode greedy.

**Jawab**

1. Tentukan nilai panjang, waktu, dan waktu rata-rata

Ada 3 program, dimisalkan panjangnya L1, L2, dan L3 dengan nilai L1=5, L2=10, dan L3=3

Waktu, disini tidak diketahui, berarti dianggap waktu tidak mempengaruhi proses penyimpanan, berarti tidak ada waktu rata-rata.

Berarti dalam kasus ini yang berpengaruh hanya panjang dari setiap datanya.

1. Urutan penyimpanan dengan menggunakan teknik faktorial sesuai dengan jumlah data.

Dari kasus diketahui jumlah data (n) adalah 3, berarti kombinasi yang dibutuhkan adalah n!, yaitu 3!=3\*2\*1 = 6. Jadi dibutuhkan 6 langkah dalam proses penyusunannya.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Order** | **D(L)** | **Total** |
| **1** | **1,2,3** | **5+(5+10)+(5+10+3)** | **38** |
| **2** | **1,3,2** | **5+(5+3)+(5+3+10)** | **31** |
| **3** | **2,1,3** | **10+(10+5)+(10+5+3)** | **43** |
| **4** | **2,3,1** | **10+(10+3)+(10+3+5)** | **41** |
| **5** | **3,1,2** | **3+(3+5)+(3+5+10)** | **29** |
| **6** | **3,2,1** | **3+(3+10)+(3+10+5)** | **34** |

**Ket. No:1 : Order 1 = 5**

**Order 1, 2 =5+10**

**Order 1,2,3 = 5+10+3**

**Jadi Total Order 1,2,3 = 5+(5+10)+(5+10+3)**

1. Dari nilai di atas didapat nilai minimal adalah
2. Nilai terkecil pertama adalah 29, yaitu untuk posisi penyimpanan urutan ke-3 pada posisi pertama
3. Nilai terkecil kedua adalah 31, yaitu untuk posisi penyimpanan urutan ke-1 pada posisi pertama
4. **Nilai terkecil ketiga bukan 34 dan 38**, sebab urutan penyimpanan pada posisi ke-3 dan ke-1 sudah diwakili oleh 29 dan 31, sehingga untuk **urutan ketiga adalah 41.**

**Contoh 1:**

Diketahui 4 program yang akan disimpan dalam media penyimpanan dengan panjang masing-masing 6, 8, 4, dan 2. Bagaimana proses penyimpanan yang optimal dengan metode greedy.

**Jawab**

1. Tentukan nilai panjang, waktu, dan waktu rata-rata
   1. Ada 4 program, dimisalkan panjangnya L1, L2, L3, dan L4 dengan nilai L1=6, L2=8, L3=4 dan L4=2
   2. Waktu, disini tidak diketahui, berarti dianggap waktu tidak mempengaruhi proses penyimpanan, berarti tidak ada waktu rata-rata.
   3. Berarti dalam kasus ini yang berpengaruh hanya panjang dari setiap datanya.
2. Urutan penyimpanan dengan menggunakan teknik faktorial sesuai dengan jumlah data.
   1. Dari kasus diketahui jumlah data (n) adalah 4, berarti kombinasi yang dibutuhkan adalah n!, yaitu 4!=4\*3\*2\*1 = 24. Jadi dibutuhkan 24 langkah dalam proses penyusunannya.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Order** | **D(L)** | **Total** |
| **1** | **1,2,3,4** | **6+(6+8)+(6+8+4)+(6+8+4+2)** |  |
| **2** | **1,2,4,3** | **6+(6+8)+(6+8+2)+(6+8+2+4)** |  |
| **3** | **1,3,2,4** | **6+(6+4)+(6+4+8)+(6+4+8+2)** |  |
| **4** | **1,3,4,2** | **......(selesaikan seperti di atas)** |  |
| **5** | **1,4,2,3** | **......** |  |
| **6** | **1,4,3,2** | **......** |  |
| **7** | **2,1,3,4** | **......** |  |
| **8** | **2,1,4,3** | **......** |  |
| **9** | **2,3,1,4** | **......** |  |
| **10** | **2,3,4,1** | **......** |  |
| **11** | **2,4,1,3** | **......** |  |
| **12** | **2,4,3,1** | **......** |  |
| **13** | **3,1,2,4** | **......** |  |
| **14** | **3,1,4,2** | **......** |  |
| **15** | **3,2,1,4** | **......** |  |
| **16** | **3,2,4,1** | **......** |  |
| **17** | **3,4,1,2** | **......** |  |
| **18** | **3,4,2,1** | **......** |  |
| **19** | **4,1,2,3** | **......** |  |
| **20** | **4,1,3,2** | **......** |  |
| **21** | **4,2,1,3** | **......** |  |
| **22** | **4,2,3,1** | **......** |  |
| **23** | **4,3,1,2** | **......** |  |
| **24** | **4,3,2,1** | **......** |  |

1. Dari tabel nilai di atas carilah nilai minimalnya pada setiap level
2. **Knapsack Problem**

Knapsack problem adalah suatu masalah bagaimana cara menentukan pemilihan barang dari sekumpulan barang di mana setiap barang tersebut mempunyai berat dan profit masing masing, sehingga dari pemilihan barang tersebut didapatkan profit yang maksimum.

**Knapsack 0/1**

knapsack 0/1, yaitu suatu objek diambil seluruh bagiannya atau tidak sama sekali.

* + Cara terbaik agar menguntungkan : bukan hanya dari hasilnya optimal tetapi juga banyaknya langkah yang dibutuhkan

Diberikan n buah objek dan sebuah knapsack dengan kapasitas bobot W.

Setiap objek memiliki properti bobot (weigth) wi dan keuntungan(profit) pi.

persoalan adalah memilih memilih objek-objek yang dimasukkan ke dalam knapsack sedemikian sehingga memaksimumkan keuntungan. Total bobot objek yang dimasukkan ke dalam knapsack tidak boleh melebihi kapasitas knapsack.

Solusi persoalan dinyatakan sebagai vektor n-tupel:

X = {x1, x2, … , xn}

xi = 1 jika objek ke-i dimasukkan ke dalam knapsack,

xi = 0 jika objek ke-i tidak dimasukkan.

Persoalan 0/1 Knapsack **dapat kita pandang** :

sebagai mencari himpunan bagian (subset) dari keseluruhan objek yang muat ke dalam knapsack dan memberikan total keuntungan terbesar.

Penyelesaian dengan Greedy:

1. Greedy by Profit

Pada setiap langkah Knapsack diisi dengan obyek yang mempunyai keuntungan terbesar.

Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang paling menguntungkan terlebih dahulu.

Pertama kali dilakukan adalah menurutkan secara menurun obyek-obyek berdasarkan profitnya . Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan).

Data awal :

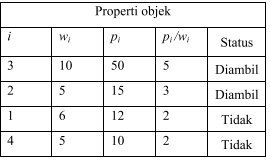
w1 = 6; p1 = 12

w2 = 5; p2 = 15

w3 = 10; p3 = 50

w4 = 5; p4 = 10

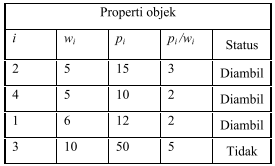
Kapasitas knapsack W = 16



1. Greedy by Wight

Pada setiap langkah, knapsack diisi dengan objek yang mempunyai berat paling ringan. Strategi ini mencoba memaksimumkan keuntungan dengan memasukan sebanyak mungkin objek kedalam knapsack.

Pertama kali yang dilakukan adalah mengurutkan secara menaik objek-objek berdasarkan weight-nya. Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan).



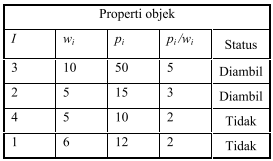
1. Greedy By Density

Pada setiap langkah, knapsack diisi dengan obyek yang mempunyai densitas terbesar (perbandingan nilai dan berat terbesar).

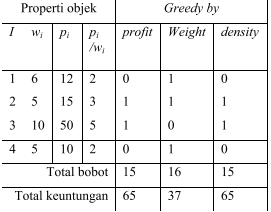
Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang mempunyai keuntungan per unit berat terbesar.

Pertama kali yang dilakukan adalah mencari nilai profit per unit/ density dari tiap-tiap objek. Kemudian obyek-obyek diurutkan berdasarkan densitasnya.

Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan).



Perbandingan hasil:



Penggunaan 3 strategi diatas tidak menjamin akan memberikan solusi optimal.

CONTOH 2:

Kapasitas M=20, dengan jumlah barang =3

Berat Wi masing-masing barang (W1,W2,W3) 🡪 (18,15,,10)

Nilai Profit masing-masing barang (P1,P2,P3) 🡪 (25,24,15)

Pilih Barang dengan nilai profit maksimal

P1=25 🡪 x1=1. batas atas nilai

P2=24 🡪 x2=2/15.

P3=15 🡪 x3 =0. batas bawah nilai.

Pilih barang dengan berat minimal

W1 = 18 🡪 x1=0. batas bawah

W2=15 🡪 x2 = 2/3

W3=10 🡪 x3=1. batas atas.

Pilih barang dengan menghitung perbandingan yang terbesar dari profit dibagi Berat (Pi/Wi) diurut secara tidak naik.

P1/w1=25/18 (1.38) 🡪 x1=0. karena terkecil x1=0

P2/w2=24/ 15 (1.6)🡪 x2=1. karena terbesar x2=1

P3/w3=15/10 (1.5)🡪 x3=1/2 dicari dengan fungsi pembatas x3=1/2.

Buat Tabel

|  |  |  |  |
| --- | --- | --- | --- |
| Solusi | (x1,x2,x3) | Σwixi | Σpixi |
| Pi max | 1,2/15,0 | 20 | 28.2 |
| Wi min | 0,2/3,1 | 20 | 31.0 |
| Pi/Wi max | 0,1,1/2 | 20 | 31.5 |

Nilai profit maksimal = 31.5.

1. **Minimum Spanning Tree Problem**

Permasalahan umum dari minimum spanning tree adalah mencari minimum biaya (cost) spanning tree dari setiap ruas (edge) suatu graph yang membentuk pohon (tree).

Dalam mendapatkan solusi yang diharapkan maka akan dipilih ruas menurut kriteria optimisasi yang menghasilkan biaya minimum. Dengan demikian penambahan jumlah biayanya relatif kecil dari setiap ruas yang telah terpilih dan membentuk spanning tree.

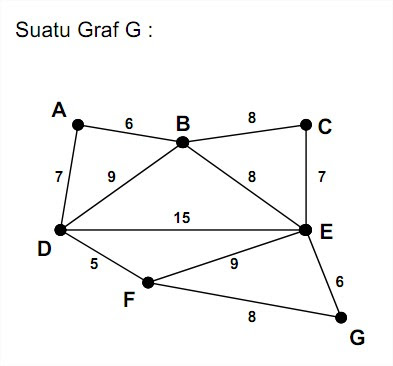
Untuk masalah minimum spanning tree, syarat graph dapat dicari minimum spanning treenya adalah :

* Graph harus terhubung
* Ruasnya punya bobot / nilai
* Graphnya tidak berarah

Algoritma yang dapat dipakai untuk menentukan minimum spanning tree adalah :

* Algoritma Solin
* Algoritma Kruskal
* Algoritma Prim’s
* **Algoritma Solin**

Suatu Graph G, seperti gambar di bawah ini.Ini adalah graf berbobot awal. Graf ini bukan pohon karena ada sirkuit. Nama yang lebih tepat untuk diagram ini adalah Graf atau Network. Angka-angka dekat garis penghubung/ruas adalah bobotnya. Nilai bobot dari Graf tesebut adalah : 86



Kita akan mencari MST dengan menggunakan Algoritma Solin dan Kruskal untuk Graf G   
diatas.   
  
Penyeselaian :

a. Urutkan Ruas Graf (G) menurut bobotnya dari bobot yang terbesar sampai bobot   
yang terkecil.

Bobot RUAS

15 D,E

9 B,D E,F

8 B,C B,E F,G

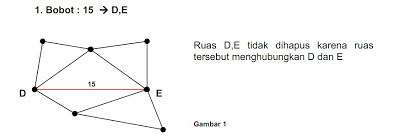
7 A,D C,E

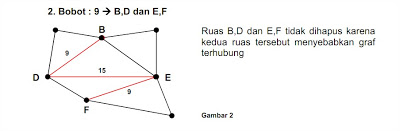
6 A,B E,G

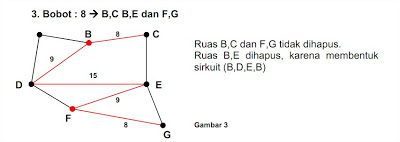
5 D,F

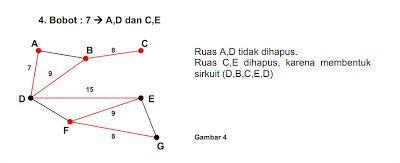
b. Lakukan penghapusan masing-masing ruas yang tidak menyebabkan graf menjadi   
tidak terhubung atau membentuk sirkuit.

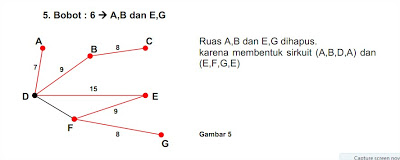
Kita mulai melakukan tahapan penghapusan dengan ruas dengan nilai bobot   
terbesar sampai bobot terkecil :

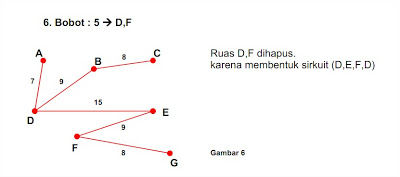










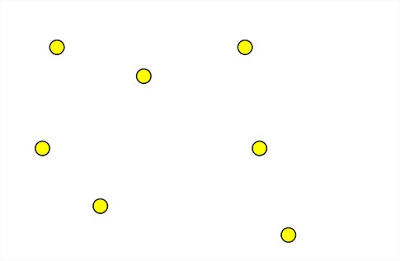


Tahap Penghapusan Selesai, Gambar 6 adalah Minimun Spanning Tree dari Graf G   
dengan Nilai Bobot : 56

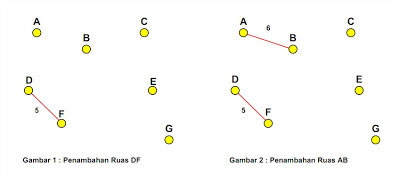
* **Algoritma Kruskal**

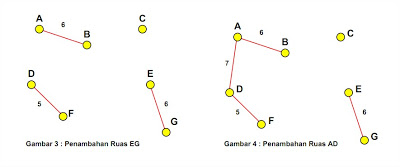
Untuk mencari pohon rentang minimum dari graph dengan algoritma yang ditemukan Kruskal, mula-mula semua garis dalam graph diurut berdasarkan bobotnya dari kecil ke besar. Kemudian pilih garis dengan bobot terkecil. Pada setiap langkah dipilih garis dengan bobot terkecil, tetapi tidak membentuk loop garis-garis yang sudah dipilih terdahulu.

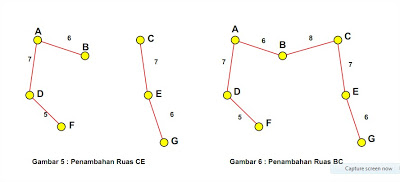
Dengan Graph yang sama, kita akan mencari Minimun Spanning Tree dengan algoritma Kruskal.  
1. Mula-mula kita buat Graf G hanya terdiri dari Simpul saja.

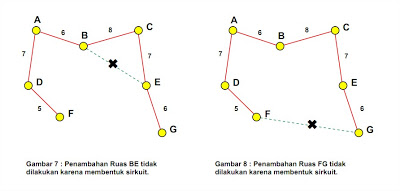


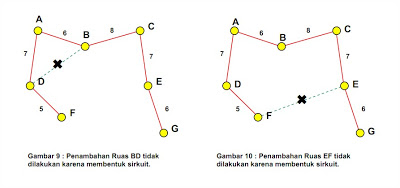
1. Urutkan Ruas dari bobot kecil ke besar (DF, AB, EG, AD, CE, BC, BE, FG, BD,   
   EF,DE), kemudian berdasarkan urutan tersebut, kita menambahkan ruas dengan   
   mencegah terbentuknya sirkuit.

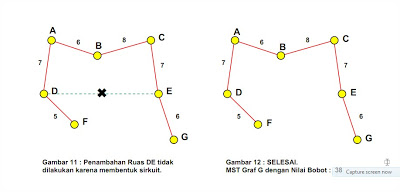












1. **Shortest Path Problem**

* Permasalahan = menghitung jalur terpendek dari sebuah graph berarah.
* bagaimana mencari sebuah jalur pada graf yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut.

Ada beberapa macam persoalan lintasan terpendek,

antara lain:

1. Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*).
2. Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shoertes path*).
4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

Kriteria utk permasalahan jalur terpendek/SP problem tsb :

1. Setiap ruas pd graph hrs mpy nilai (*label graph*)
2. Setiap ruas pd graph tdk hrs terhubung (*unconnected*)
3. Setiap ruas pd graph tsb hrs mempunyai arah (graph berarah).

Skema umum Algoritma Greedy Algoritma greedy disusun oleh elemen-elemen berikut:

* + - 1. Himpunan kandidat.

Berisi elemen-elemen pembentuk solusi.

* + - 1. Himpunan solusi Berisi kandidat-kandidat yang terpilih sebagai solusi  persoalan.
      2. Fungsi seleksi (selection function).

Memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

* + - 1. Fungsi kelayakan (feasible)

Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (constraints) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

* + - 1. Fungsi obyektif yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain).