PENEMPATAN KANTOR POS DENGAN ALGORITMA PROGRAM DINAMIS

Hanson Prihantoro Putro (13505045)

Sekolah Teknik Elektro dan Informatika ITB
Jl. Ganesha 10 Bandung 40135
if15045@students.if.itb.ac.id

ABSTRAK

Makalah ini membahas tentang salah satu penggunaan algoritma program dinamis yaitu bagaimana cara penempatan suatu posisi agar mendapatkan keuntungan yang maksimum. Persoalan ini mungkin dialami oleh bebeberapa orang seperti pemerintah yang akan melakukan pembangunan gedung pada suatu wilayah atau hanya sekedar beberapa orang yang akan melakukan penempatan beberapa barang. Dengan algoritma program dinamis ini kita bisa menyelesaikan permasalahan ini dengan waktu yang lebih cepat dan dengan hasil yang akurat. Dalam penyusunan algoritma program dinamis ini, kita membutuhkan tabel yang setiap selnya digunakan untuk menyelesaikan bagian permasalahan selanjutnya. Dari tabel inilah, kita bisa mengetahui nilai keuntungan yang ingin kita dapatkan.

Kata kunci: *greedy*, *divide-and-conquer*, *top-down*, *bottom-up*, *exhaustive serach*.

1. PENDAHULUAN

Dalam kehidupan sehari-hari, kita pasti selalu berhadapan dengan berbagai permasalahan. Dari permasalahan-permasalahan tersebut, tidak sedikit yang merupakan persoalan mengenai pilihan, ingin mengambil yang mana, mana yang memberikan keuntungan yang maksimal. Permasalahan untuk membawa barang yang penting ke dalam mobil yang hanya muat beberapa barang saja, permasalahan untuk membangun rumah sakit ataupun kantor pos sesedikit mungkin pada suatu wilayah namun dengan cakupan seluas mungkin. Persoalan-persoalan ini kita sebut sebagai persoalan optimasi. Persoalan optimasi ini bisa kita modelkan dalam bentuk matematis yang tentu saja dapat kita selesaiakan dengan perhitungan matematis pula, yang dengan bantuan komputer kita bisa menyelesaikannya dengan lebih mudah.

Kita bisa membagi beberapa contoh permasalahan tersebut menjadi bagian-bagian kecil, untuk kemudian kita selesaikan bagian kecil permasalahan itu, lalu kita gabungkan solusi dari bagian-bagian kecil ini untuk mendapatkan solusi dari permalasalahan utama. Kadang kala kita mendapati bahwa cara yang kita lakukan membawa kita pada beberapa contoh bagian yang sama. Jika kita selesaikan tiap bagian ini secara terpisah, maka kita akan membuat suatu contoh penyelesaian yang sama juga. Jika kita tidak memperhatikan perulangan ini, kita telah menyusun suatu algoritma yang tidak efisien. Kita dapat mengambil keuntungan dari perulangan ini dan menyusun setiap bagian hanya sekali, sehingga kita bisa menyimpan solusi ini untuk perhitungan selanjutnya. Dengan demikian kita bisa mendapatkan algoritma yang lebih efisien, inilah pemrograman dinamis. Ide dari program dinamis ini cukup sederhana yaitu menghindari perhitungan hal yang sama 2 kali, biasanya dengan membuat sebuah tabel yang diisi oleh bagian-bagian kecil permasalahan yang telah diselesaikan.

Divide-and-Conquer adalah metode *top-down*. Saat sebuah masalah diselesaikan dengan *divide-and-conquer*, kita langsung menyelesaikan permasalahan utama, yaitu dengan membaginya menjadi bagian-bagian yang lebih kecil sebagai bagian dari proses algoritma *divide-and-conquer*.

Cara penyelesaian dengan metode program dinamis hampir serupa dengan metode *greedy*, dimana metode *greedy* juga membentuk solusi secara bertahap. Pada metode *greedy*, kita membuat keputusan pada setiap tahap dengan mengambil pilihan yang paling sesuai (yang memenuhi ukuran optimasi yang digunakan). Pengambilan keputusan pada setiap tahap didasarkan hanya pada informasi lokal dan pada setiap tahap itu, kita tidak pernah membuat keputusan yang salah. Pada contoh-contoh persoalan tertentu, algoritma *greedy* bekerja dengan baik, tetapi pada kebanyakan persoalan yang lain tidak. Hal ni terjadi karena pengambilan keputusan pada setiap langkah greedy tidak pernah mempertimbangkan lebih jauh apakah pilihan tersebut pada langkah-langkah selanjutnya merupakan pilihan yang tepat. Dari sejumlah pilihan yang cocok, kita tidak tahu pilihan mana yang terbaik sampai kita menuju proses yang lebih jauh ke depan.

Cara lain yang bisa digunakan untuk menyelesaikan persoalan adalah dengan mengenumerasi semua rangkaian

keputusan yang mungkin dan memilih rangkaian keputusan yang terbaik, atau bisa kita sebut dengan *exhaustive search*. Untuk persoalan dengan batasan angka yang besar, tentu saja *exhaustive search* tidak mangkus.

2. DASAR TEORI

2.1 Pengenalan Program Dinamis

Program Dinamis (*Dynamic Programming*) adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian sehingga solusi dari persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan. Pada penyelesaian engan metode ini:

1. Terdapat sejumlah berhingga pilihan yang mungkin.
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya.
3. Persyaratan optimasi dan kendala digunakan untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. Prinsip ini berbunyi : "Jika solusi total optimal, maka bagian solusi sampai ke tahap ke-*k* juga optimal." Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap *k* ke tahap ke *k+1*, kita dapat menggunakan hasil optimal dai tahap ke *k* tanpa harus kembali ke tahap awal. Jika pada setiap tahap kita menghitung ongkos (*cost*) maka dapat dirumuskan secara umum:

Ongkos tahap *k+1* = (ongkos yang dihasilkan pada tahap *k*) + (ongkos dari tahap *k* ke *k+1*)

Dengan prinsip optimalitas, dijamin bahwa pengambilan keputusan pada suatu tahap, adalah keputusan yang benar untuk tahap-tahap selanjutnya. Jadi, bisa kita katakan perbedaan antara metode *greedy* denan metode program dinamis adalah pada metode *greedy* hanya 1 rangkaian keputusan yang pernah dihasilkan, sedangkan pada metode program dinamis lebih dari 1 rangkaian keputusan.

Program dinamis mengacu pada perhitungan dengan menggunakan tabel. Seperti halnya algoritma greedy, program dinamis juga digunakan untuk memecahkan masalah optimasi, yaitu masalah yang memiliki banyak kemungkinan solusi, dan kita diminta untuk mencari solusi yang memberikan nilai optimal. Program dinamis merupakan teknik *bottom-up*. Kita biasanya langsung memulai dengan bagian yang terkecil, bagian persoalan yang paling sederhana. Dengan menggabungkan solusi-solusi itu, kita mendapatkan jawaban dari permasalahan ini mulai dari yang ukurannya kecil hingga permasalahan dengan ukuran yang lebih besar, sehingga akhirnya kita bisa menyelesaikan permasalahan utama.

2.2 Karakteristik Program Dinamis

Program Dinamis dapat diterapkan pada persoalan yang memiliki karakteristik sebagai berikut:

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang pada setiap tahap hanya diambil satu keputusan.
2. Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan bermacam-macam kemungkinan masukan yang ada pada tahap tersebut. Jumlah status bisa berhingga (*finite*) atau tidak berhingga (*infinite*).
3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan ditambah dengan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *k* memberikan keputusan terbaik untuk setiap status pada tahap *k+1*.
8. Prinsip optimalisasi berlaku pada persoalan tersebut.

Secara umum ada 4 langkah yang dilakukan dalam mengembangkan algoritma program dinamis:

1. Karakterisikkan struktur solusi optimal.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai optimal secara maju atau mundur.
4. Konstruksikan solusi optimal.

2.3 Jenis Penyelesaian dengan Program Dinamis

Dalam menyelesaikan persoalan dengan program dinamis, kita dapat menggunakan 2 pendekatan berbeda: maju (*forward*) dan mundur (*backward*). Misalnya $x_1,x_2,x_3,...,x_n$ menyatakan peubah (*variable*) keputusan yang harus dibuat masing-masing untuk tahap 1,2,3,...,*n*. Maka,

1. Program dinamis maju bergerak mulai dari tahap 1, terus maju ke tahap 2, 3 dan seterusnya sampai tahap n. Runtutan peubah keputusan adalah $x_1,x_2,x_3,...,x_n$.
2. Proram dinamis mundur bergerak mulai dari tahap *n*, terus mundur ke tahap *n-1*, *n-2* dan seterusnya

hingga tahap 1. Runtutan keputusannya adalah x_n, x_{n-1},x_{n-2},...,x_1.

3. PERMASALAHAN

Persoalan ini diambil dari salah satu soal *International Olympiads in Informatics* yang diadakan pada tahun 2000 (IOI'00) di Beijing, China.

Problem

Ada sebuah jalur jalan raya dengan perkampungan-perkampungan di sepanjang jalan raya tersebut. Jalan raya ini direpresentasiakn sebagai sebuah sumbu bilangan integer dan posisi dari setiap perkampungan dinyatakan dengan satu integer tunggal. Tidak ada 2 perkampungan yang berada pada posisi yang sama. Jarak antara 2 kampung merupakan nilai selisih dari perbedaan koordinat kedua kampung tersebut.

Akan dibangun beberapa kantor pos, namun mungkin tidak akan dibangun pada setiap kampung. Kantor pos akan dibangun pada posisi yang sama pada beberapa kampung. Untuk membangun, posisi kantor pos ini harus dipilih pada suatu kampung sehingga jumlah dari jarak antara setiap kampung dengan kantor pos terdekatnya, adalah seminimum mungkin.

Diberikan pada kita posisi dari setiap kampung yang ada pada jalan raya tersebut serta jumlah kantor pos yang akan dibangun. Kita diminta untuk menghitung jumlah jarak minimum dari setiap kampung ke kantor pos terdekat, serta posisi dari setiap kantor pos yang dibangun agar jumlah jarak tersebut minimum.

Contoh dari permasalahan ini diberikan sebagai berikut:

```
10  5
1  2  3  6  7  9  11  22  44  50
```

Contoh tersebut dapat dijelaskan sebagai berikut. Bilangan pertama pada baris pertama menujukkan jumlah perkampungan yang ada pada jalan raya ini, yang diperhatikan sebagai tempat pembangunan kantor pos. Masih di baris pertama, bilangan kedua merupakan jumlah kantor pos yang akan dibangun. Pada baris kedua terdapat sederatan bilangan yang menunjukkan sejumlah posisi perkampungan sebanyak jumlah perkampungan seperti yang sudah disebutkan di awal.

Maka jika contoh permasalahannya adalah bilangan-bilangan itu, maka hasil yang diperoleh adalah:

```
9
2  7  22  44  50
```

Hasil tersebut dapat diartikan sebagai berikut. Bilangan pada baris pertama menunjukkan jumlah minimum dari jarak setiap perkampungan terhadap kantor pos terdekatnya. Sederatan bilangan pada baris kedua menunjukkan posisi kantor pos yang akan dibangun agar jumlah jarak yang dihasilkan tersebut minimum.

4. PENYELESAIAN

4.1 Ide Penyelesaian

Ini merupakan salah satu contoh permasalahan yang mungkin dihadapi pemerintah dalam pembangunan suatu wilayah. Persoalan ini dapat diselesaikan dengan berbagai cara. Jika diserahkan pada para pemrogram, mungkin mereka akan menyelesaiakan permasalahan ini dengan algoritma program dinamis.

Ada banyak tahap dalam menyelesaikan permasalahan ini. Untuk penyelesaian kali ini, tahapan dibagi sebagai jumlah kantor pos. Setiap tahap ditunjukkan sebagai satu kantor pos yang akan dibangun. Kemudian status-status permasalahan dibagi sebagai jumlah perkampungan yang ada. Jumlah perkampungan yang ada haruslah berhingga.

Kemudian, ongkos yang akan dicapai tujuannya adalah jumlah dari jarak suatu kampung ke kantor pos terdekat Setiap bertambah perkampungan, bertambah pula ongkos yang didapatkan. Ongkos ini juga dipengaruhi jumlah kantor pos yang akan dibangun.

Untuk mencapai fungsi objektif yang minimum, kita memerlukan hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap *k* memberikan keputusan terbaik untuk setiap status pada tahap *k+1*. Fungsi inilah yang biasanya sulit kita dapatkan. Namun dengan perhitungan yang cermat, kita seharusnya bisa mendapatkan fungsi hubungan ini. Untuk penyelesaian kali ini, fungsi rekursif yang diperoleh adalah:

$$f(i,j)\begin{cases} c(1,j) & ; i=1 \\ \min\limits_{k=i-1 \rightarrow j-1} (f(i-1,k) + c(k+1,j)) & ; i>1 \end{cases}$$

Fungsi tersebut berlaku dimana i merupakan tahapan pembangunan kantor pos sedangkan j merupakan status pada setiap perkampungan. C merupakan fungsi yang mencari jumlah jarak antara perkampungan dengan kantor pos terdekat dimana kantor pos diletakkan pada kampung yang paling tengah.

Fungsi tersebut bisa dibayangkan sebagai berikut. Untuk pembangunan hanya 1 kantor pos, kita langsung cari fungsinya dengan fungsi C, yaitu meletakkan kantor pos di kampung yang paling tengah. Dengan demikian, diharapkan setiap kampung akan mendapatkan jarak ke kantor pos tersebut secara rata. Untuk pembangunan kantor pos kedua dan juga kantor pos selanjutnya, kita akan mulai mencari posisi yang paling menguntungkan. Dalam kantor pos selanjutnya ini, kita akan coba

membangun ditengah setiap status, mulai dari kampung awal hingga tempat kampung tersebut dicari.

Sebagai contoh kita akan mencari f(3,5) yaitu fungsi dengan 3 kantor pos pada 5 perkampungan. Fungsi ini didapat dengan melakukan percobaan:

1. Membangun 2 kantor pos diantara 1 dan 2(f(2,2), pada kampung itu sendiri) dan 1 kantor pos lagi ditengah kampung 3 dan 5 (c(3,5)).
2. Membangun 2 kantor pos diantara 1 dan 3 (f(2,3)) dan 1 kantor pos lagi ditengah kampung 4 dan 5 (c(4,5)).
3. Membangun 2 kantor pos diantara 1 dan 4 (f(2,4)) dan 1 kantor pos lagi ditengah kampung 5 dan 5 (c(5,5), pada kampung itu sendiri).

Dengan cara program dinamis ini bisa kita lihat bahwa untuk bisa membangun 3 kantor pos, kita harus telah mengetahui dulu solusi minimum dari pembangunan 2 kantor pos terlebih dahulu, untuk kemudian dikombinasikan dengan harga pembangunan kantor pos ketiga.

Jika kita telah memahami bagaimana fungsi tersebut digunakan, maka kita dapat langsung mengisi tabel yang memperlihatkan pembangunan kantor pos pada setiap tahap dan status. Kita akan menggambar tabel dengan kondisi sesuai pada salah satu contoh masalah diatas.

Tabel 1 Tabel Pengisian Program Dinamis untuk Kantor Pos

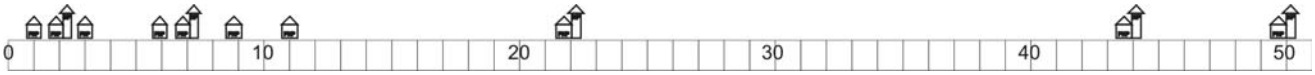
	PKP 1	PKP 2	PKP 3	PKP 4	PKP 5	PKP 6	PKP 7	PKP 8	PKP 9	PKP 10
KP 1	0	1	2	6	10	16	21	37	74	117 (4)
KP 2	0	0	1	2	3	5	9	21	37	43 ((),)
KP 3	0	0	0	1	2	3	5	9	21	27
KP 4	0	0	0	0	1	2	3	5	9	15
KP 5	0	0	0	0	0	1	2	3	5	9

Keterangan: KP = Kantor Pos, PKP = Perkampungan

Dari tabel diatas dapat kita ketahui jawaban dari setiap contoh permasalahan, mulai dari jika hanya 1 kantor pos dibangun pada 1 kampung hingga 5 kantor pos yang dibangun pada wilayah dengan 10 kampung. Untuk pertanyaan pada contoh persoalan diatas, maka jawaban dapat diperoleh dengan melihat pada baris 5 kantor pos dengan 10 perkampungan, yaitu 9.

Untuk mendapatkan posisi 5 kantor pos tersebut, kita harus menelusuri ke belakang jawaban kita. 9 diperoleh dari 4 kantor pos pada 9 perkampungan. Maka kantor pos kelima diletakkan pada kampung ke 10 (posisi 50). Jawaban f(4,9) diperoleh dari 3 kantor pos pada 8 perkampungan, maka kantor pos keempat diletakkan pada kampung ke 9 (posisi 44). Begitu seterusnya kantor pos ke 3 pada kampung ke 8 (posisi 22). Pada pembangunan kantor pos kedua, nilai 9 diperoleh dari peletakkan 1 kantor pos dengan 3 kampung dan kantor pos kedua diletakkan diantara kampung ke 4 dan kampung ke 7. Dari sini kita peroleh bahwa kantor pos pertama diletakkan ditengah kampung 1 dan 3 (kampung 2 dengan posisi 2) kemudian kantor pos kedua ditengah kampung ke 4 dan 7 (kampung 5 dengan posisi 7). Sehingga kita peroleh jawaban utuhnya:

```
9
2 7 22 44 50
```



Gambar 1. Ilustrasi Penempatan 5 Kantor Pos pada 10 perkampungan

Dari gambar diatas telah dideskripsikan, dimana posisi setiap kampung serta kampung mana saja yang dibangun kantor pos. Dengan penempatan seperti itu kita bisa menghitung jumlah dari jarak setiap kampung dengan kantor pos terdekat:

F(5,10) = |(1-2)| + |(2-2)| + |(3-2)| + |(6-7)| + |(7-7)| + |(9-7)| + |(11-7)| + |(22-22)| + |(44-44)| + |(50-50)|

= 1 + 0 + 1 + 1 + 0 + 2 + 4 + 0 + 0 + 0 = 9

4.2 Kode Program

```
Program Post;
Const
  MaxOfficeCount=30;
  MaxVillageCount=400;

Var
  OfficeCount,VillageCount : integer;
  Position : array[1..MaxVillageCount] of
integer;

{inisialisasi dan pembacaan}
Procedure Init;
var vc:integer;
begin
  read(VillageCount,OfficeCount);

  for vc:=1 to VillageCount do
    read(Position[vc]);
end;

{mendapatkan jumlah jarak jika kantor pos
diletakkan di tengah}
Function GetCost(left,right:integer):Longint;
var
  mean:integer;
  v:integer;
  result:longint;
begin
  mean:=Position[(left+right)div 2];
  result:=0;

  for v:=left to right do
    inc(result,abs(mean-Position[v]));

  GetCost:=result;
end;


Type
TbackWay = array
[1..MaxOfficeCount,1..MaxVillageCount]of
integer;
Var
  Cost    : array
[1..MaxOfficeCount,1..MaxVillageCount] of
longint;
  BackWay : TBackWay;

{proses utama untuk mengisi setiap sel tabel}
Procedure Process;
var
   o,v,i:integer;
   temp:Longint;

begin
  for o:=1 to OfficeCount do
    for v:=1 to VillageCount do
    begin
      Cost[o,v]:=MaxLongint;
      BackWay[o,v]:=-1;
    end;

  for v:=1 to VillageCount do
    Cost[1,v]:=GetCost(1,v);
  for o:=2 to OfficeCount do
    for v:=1 to VillageCount do
      for i:=o-1 to v-1 do
      begin
        temp:=Cost[o-1,i]+GetCost(i+1,v);

        if temp<Cost[o,v] then
        begin
          Cost[o,v]:=temp;
          BackWay[o,v]:=i;
        end;
      end;
end;

{mencetak nilai yang diminta serta menelusuri
kantor pos yang akan dibangun}
Procedure Print;
var officelist:array[1..MaxOfficeCount]of
integer;
  v,o:integer;
begin
  writeln(Cost[OfficeCount,VillageCount]);
  v:=VillageCount;
  for o:=OfficeCount downto 1 do
  begin
    if o=1 then
      officelist[o]:=(v+1)div 2
    else
      officelist[o]:=(v+BackWay[o,v]+1)div 2;

    v:=BackWay[o,v];
  end;
  write(Position[officelist[1]]);

  for o:=2 to OfficeCount do
    write(' ',Position[officelist[o]]);
end;

{Program Utama}
BEGIN
  Init;
  Process;
  Print;
END.
```

Kode ini dibuat dengan bahasa pemrograman Pascal. Dengan kode ini, kita bisa menyelesaikan berbagai contoh penempatan kantor pos pada suatu wilayah jika masukkannya sesuai dengan yang diminta.

Analisis yang bisa diberikan pada program ini adalah bahwa program ini berjalan dengan kompleksitas $O(n^2)$ dengan mengenaralisir jumlah kantor pos dan jumlah perkampungan sama, yaitu n. Walaupun dalam kode program terlihat ada 3 kalang yang bersarang, namun kalang terakhir bisa dibilang hanya menutupi kalnag kedua yang tidak lengkap. Hal ini masih jauh lebih cepar dibandingkan dengan jika kita

mencoba melakukan penempatan satu per satu di setiap kampung.

5. KESIMPULAN

Permasalahan penempatan kantor pos dapat diselesaikan dengan algoritma program dinamis hanya dalam periode kuadratis. Dengan program dinamis pun kita juga bisa melihat contoh permasalahan yang merupakan bagian kecil dari permasalahan utama. Jika di permasalahan utama kita membutuhkan nilai untuk penempatan 5 kantor pos pada 10 perkampungan, kita juga bisa melihat nilai jika kita hanya ingin menempatkan 4 kantor pos pada 5 perkampungan. Hal ini tentunya akan membayar waktu yang dibutuhkan untuk perhitungan pencarian nilai pada permasalahan utamanya.

REFERENSI

[1] Website International Olympiads in Informatics
http://www.olympiads.win.tue.nl/
Tanggal akses: 21 Mei 2007 pukul 16:30.

[2] Brassard, Gilles. (1995). Fundamental of Algorithms. Prentice Hall, New Jersey.

[3] Munir, Rinaldi. (2005), Strategi Algoritmik. Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.