PERJAMUAN ALAKAPURI

Jansen

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if16028@students.if.itb.ac.id

ABSTRAK

Permasalahan optimasi merupakan permasalahan umum dalam kehidupan setiap manusia. Dalam usaha mengoptimasikan persoalan, manusia selalu berusaha untuk menemukan cara-cara pengoptimasian yang efektif. Salah satu pendorong untuk mengoptimasi adalah batasan-batasan yang sering kali muncul dan menyulitkan manusia.

Namun, banyak sekali permasalahan-permasalahan tersebut sebenarnya dapat diselesaikan dengan algoritma yang paling sederhana sekalipun, misalnya algoritma *greedy*.

Salah satu tipe dari permasalahan yang dapat diselesaikan dengan algoritma *greedy* adalah tipe Perjamuan Alakapuri. Penjelasan mengenai Perjamuan Alakapuri akan dijelaskan selanjutnya.

Masalah ini merupakan masalah yang diajukan pada Pelatihan Nasional (Pelatnas) ke-2 Tim Olimpiade Komputer Indonesia Tim Olimpiade Komputer Indonesia (TOKI) tahun 2006 yang bertempat di Institut Teknologi Bandung.

Kata kunci: Perjamuan Alakapuri, *greedy*.

1. PENDAHULUAN

Berikut adalah definisi persoalan yang disadur secara langsung dari situs resmi Pelatnas 2 TOKI 2006.

Alkisah, bertujuan untuk memamerkan kekayaannya sekaligus meningkatkan pengaruhnya di kalangan para dewa, dewa kekayaan Kubera berencana mengundang para dewa untuk mengikuti jamuan makan di kotanya, Alakapuri.

Kubera sangat tahu tabiat para dewa yang lain. Para dewa sangat tidak suka diduakan, oleh sebab itu, Kubera harus mengatur jadwal perjamuannya sedemikian rupa sehingga pada suatu saat tertentu, dia hanya menjamu satu dewa saja.

Dengan bantuan para pembantunya, Kubera telah menyusun sebuah daftar yang berisi: nama dewa, waktu dimana dewa tersebut bersedia menghadiri perjamuan, dan perkiraan lama waktu makan dewa tersebut.

Karena Kubera tidak pilih-pilih dalam mengundang para dewa, untuk membuat daftar lebih ringkas, Kubera kemudian berinisiatif mengganti kolom nama dewa menjadi nomor urut saja.

Hasilnya, Kubera mempunyai sebuah tabel yang kira-kira berisi seperti ini:

Tabel 1 Contoh Jadwal Kesediaan Tiap Dewa Menghadiri Perjamuan

Dewa Ke- (i)	Waktu Awal Jamuan (S)	Durasi (D)
1	2	5
2	9	7
3	15	6
4	9	3

Jelas bahwa Kubera mungkin tidak dapat mengundang semua dewa yang ada karena bisa saja jadwal di mana satu dewa bersedia dijamu beririsan dengan jadwal dewa yang lain. (Kubera tidak mungkin mengusir seorang dewa di tengah - tengah perjamuan. Satu perjamuan berakhir ketika dewa yang sedang dijamu selesai makan, dan lama waktu makan dewa tersebut tidak akan lebih lama daripada waktu perkiraan yang telah dibuat pembantu Kubera)

Contohnya, mengacu pada daftar di atas, dia tidak mungkin mengundang keempat dewa yang ada (karena jadwal dewa 2 dan 3 beririsan sehingga hanya salah satu dari mereka yang bisa diundang). Anggap saja, yang diundang adalah dewa 1, 3, dan 4. Dari satuan waktu ke - 2 sampai 6, Kubera akan menjamu dewa 1. Kemudian pada waktu ke - 9, dia akan mulai menjamu dewa 4 sampai waktu ke - 11. Dan terakhir, ia menjamu dewa 3 pada waktu 15 – 20.

Sesuai dengan tujuannya semula, Kubera tentu ingin menjamu sebanyak - banyaknya dewa.

Sebagai ahli hitung terbaiknya, Anda diminta oleh Kubera untuk membantunya menghitung berapa banyak dewa yang dapat ia undang ke perjamuannya.

Input
Baris pertama masukan berisi sebuah integer *N*, menunjukkan banyaknya dewa yang ada. Baris 2, 3, …, *N*+1 mendeskripsikan jadwal ke - *N* dewa tersebut. Baris ke *i*+1 berisikan dua buah integer S_i dan D_i, yang merepresentasikan waktu di mana dewa ke - *i* bisa hadir serta perkiraan durasi perjamuan untuk dewa tersebut.

Output
Terdiri atas sebuah integer *M*, yaitu banyak maksimum dewa yang dapat dijamu.

Contoh Input
4
2 5
9 7
15 6
9 3

Contoh Output
3

Contoh Input
4
1 2
1 3
1 4
1 5

Contoh Output
1

Contoh Input
11
4 7
1 9
3 20
4 7
3 1
5 2
10 3
9 1
7 1
10 5
11 4
Contoh Output
4

Penjelasan Input/Output
Seperti yang diuraikan pada deskripsi soal.

Batasan
N <= 100 000, 1 <= Si <= 1 000 000, 1 <= Di <= 1 000.

2. DASAR TEORI

Algoritma *greedy* adalah salah satu jenis algoritma yang untuk masalah optimasi secara langkah per langkah (*step by step*) yaitu pendekatan penyelesaian masalah yang dalam langkah demi langkahnya mencari pendekatan lokal dengan harapan mencapai solusi optimal secara global. Persoalan optimasi terdiri dari dua jenis yaitu maksimasi dan minimasi. Adapun tiap keputusan yang diambil oleh algoritma *greedy* ini tidak dapat diubah lagi pada langkah berikutnya.

Algoritma *greedy* memiliki beberapa elemen yaitu fungsi kendala dan fungsi obyektif (atau sering disebut fungsi optimasi). Tiap solusi yang memenuhi semua kendala disebut solusi layak (*feasible solution*). Solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.

Sesuai dengan namanya, prinsip algoritma *greedy* adalah "*take what you can get now!*".

Telah disebutkan sebelumnya bahwa algoritma *greedy* adalah algoritma yang memecahkan masalah langkah per langkah. Pada setiap langkahnya, algoritma ini:

1. mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now!*")
2. berharap bahwa dengan memilih optimum lokal pada setiap langkahnya akan berakhir dengan optimum global.

Contoh masalah sehari-hari yang menggunakan prinsip *greedy*:

- Memilih beberapa jenis investasi (penanaman modal)
- Mencari jalur tersingkat dari Bandung ke Surabaya
- Memilih jurusan di Perguruan Tinggi
- Bermain kartu remi

Adapun skema umum algoritma *greedy* adalah:

1. Himpunan kandidat.
2. Himpunan solusi.
3. Fungsi seleksi (*selection function*).
4. Fungsi kelayakan (*feasible*).
5. Fungsi obyektif (fungsi optimasi).

3. PENYELESAIAN

3.1 Ide Penyelesaian

Untuk menyelesaikan masalah ini, kita harus memikirkan fungsi obyektif yang baik. Sebab jika menggunakan fungsi obyektif yang salah pada algoritma *greedy*, hasil yang didapat tidak akan dapat maksimal.

Dalam persoalan ini, kita akan menggunakan strategi *greedy* sebagai berikut:
Pertama-tama, urutkanlah secara menurun tiap dewa menurut waktu awal jamuan dan untuk waktu awal jamuan yang sama memprioritaskan durasi yang lebih kecil. Dan pada tiap langkah, pililhlah secara sekuensial dewa yang masih dapat dijamu menurut keterurutan tersebut dengan fungsi kendala (fungsi syarat) jadwal perjamuan dewa yang satu tidak *overlap* dengan jadwal perjamuan dewa yang lain.

Misalkan untuk contoh pada soal di atas, keterurutan perjamuan dewa akan menjadi:

Tabel 2 Keterurutan dewa setelah diurutkan menurut waktu awal jamuan dan durasi

Dewa Ke- (i)	Waktu Awal Jamuan (S)	Durasi (D)
3	15	6
4	9	3
2	9	7
1	2	5

Setelah itu, pemilihan dewa akan sesuai dengan keterurutan di atas dan dengan fungsi batasan yang disertakan di atas. Dengan demikian, akan terpilih dewa 3, dewa 4 dan kemudian dewa 1.

3.2 Kode Program

Berikut adalah kode program dalam bahasa Pascal

```
program PerjamuanAlakapuri;

const
   MAX_DEWA = 500;

type tDewa = record
               indeks : integer;
               waktuAwalJamuan : integer;
               durasi : integer;
             end;

var
  dewa : array[1..500] of tDewa;
  jlhDewa : integer;

{ menginisialisasi dan membaca nilai dari
input keyboard }
procedure Baca;
var
  i : integer;
begin
  readln( jlhDewa );
  for i := 1 to jlhDewa do
  begin
    dewa[i].indeks := i;
    readln( dewa[i].waktuAwalJamuan,
            dewa[i].durasi );
  end;
end;

{mengurutkan data menurut Selection Sort}
procedure UrutData;
var
  i, j : integer;
  minimum : integer;
  temp : tDewa;
begin
  for i := 1 to jlhDewa - 1 do
  begin
    minimum := i;
    for j := i+1 to jlhDewa do
      if (dewa[minimum].waktuAwalJamuan >
          dewa[j].waktuAwalJamuan) or
         ((dewa[minimum].waktuAwalJamuan =
          dewa[j].waktuAwalJamuan) and
          (dewa[minimum].durasi >
          dewa[j].durasi))
      then
        minimum := j;
    temp := dewa[i];
    dewa[i] := dewa[minimum];
    dewa[minimum] := temp;
  end;
end;

{ Proses utama untuk melakukan semua jenis
kalkulasi dan pemilihan, sekaligus
mencetak dewa yang terpilih }
procedure Proses;
var
  i, pemilihanTerakhir : integer;
  jlhTerpilih : integer;
begin
  UrutData;
  pemilihanTerakhir := jlhDewa;
  jlhTerpilih := 1;
  for i := jlhDewa-1 downto 1 do
  begin
    if (dewa[i].waktuAwalJamuan +
      dewa[i].durasi <
      dewa[pemilihanTerakhir].
      waktuAwalJamuan) then
    begin
      pemilihanTerakhir := i;
      inc( jlhTerpilih );
    end;
  end;
  writeln( jlhTerpilih );
end;

{ Program Utama }
begin
  Baca;
  Proses;
end.
```

4. KESIMPULAN

Permasalahan Perjamuan Alakapuri dapat menggunakan algoritma *greedy* hanya dengan periode linear. Berbeda dengan jika menggunakan *brute force*, pencarian solusi dapat mencapai hingga faktorial.

REFERENSI

[1] Ir. Rinaldi Munir, S.T. , "Strategi Algoritmik", ITB, 2006.

[2] Neapolitan, Richard E., & Naimipour, Kumarss, "Foundations of Algorithms", D. C. Heath and Company, 1996

[3] Levitin, Anany, "The Design & Analysis of Algorithms", Pearson Education, Inc., 2003