Algorima *Greedy* Pada *Self Serve Gas Station*

Rifky Hamdani (13508024)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18024@students.itb.ac.id

Abstract*—Stasiun pengisian bahan bakar sangat banyak terdapat di Indonesia. Hal ini disebabkan karena banyaknya kendaraan bermotor yang ada di Indonesia. Untuk lebih memaksimalkannya alangkah baiknya jika dibuat *self serve gas station.

Menggunakan teori algoritma *Greedy* penulis ingin mencoba untuk membuat aplikasi dalam mesin pengisian bahan bakar. Pada mesin pengisian bahan bakar uang kembalian yang akan diterima akan di optimasi jumlah pecahannya. Hal ini karena pada tempat penyimpanan uang dalam mesin tersebut jumlahnya terbatas. Jadi jika uang kembalian yang diberikan lebih sedikit maka transaksi bisa semakin banyak. *Greedy* ini digunakan terhadap besarnya nilai nominal dari uang tersebut.

***kata Kunci*—Optimasi, algoritma *greedy*, self serve gas station**

I. PENDAHULUAN

Dengan banyaknya perkembangan dibidang otomotif dan otomatisasi. Penulis ingin membahas salah satu aplikasi otomatis di bidang industri penjualan. Yang penulis ambil adalah tentang *self serve gas station*. Mesin ini memang belum ada di Indonesia, tetapi sudah banyak terdapat di negara maju.

Mesin pengisian bahan bakar adalah mesin yang memungkinkan terjadinya transaksi pembelian bahan bakar secara mandiri (mengisi sendiri, tanpa operator). Mesin ini menerima masukan uang dan mengembalikan uang apabila terdapat sisa dari transaksi.

Yang ingin dibahas oleh penulis adalah tentang bagaimana mesin dapat memberikan kembalian secara optimal. Yang dimaksud optimal adalah uang kembalian mempunyai jumlah koin paling sedikit diantara semua kemungkinan uang kembalian.

Untuk itu program dari mesin akan mengaplikasikan algoritma *Greedy* untuk menentukan pecahan berapa saja yang bisa jadi kembaliannya. Diharapkan dengan menggunakan algoritma *Greedy* pembeli akan mendapatkan uang kembalian dengan jumlah koin sesedikit mungkin.

II. DASAR TEORI

Algoritma *Greedy* adalah salah satu metode pemecahan persoalan optimasi yang paling populer. Secara harafiah, *greedy* memiliki arti tamak atau rakus. Orang yang tamak akan mengambil sebanyak mungkin apa yang tersedia tanpa memikirkan konseskuensi ke depan. Algoritma *greedy* pun demikian, algoritma ini bersifat sederhana dan lempang dengan prinsip, "take what you can get now". Pada tiap langkah algoritma *greedy* dipilih pilihan optimum lokal, dengan harapan bahwa langkah sisanya mengarah ke solusi yang optimum global.

A. Elemen-elemen Algoritma Greedy

- Himpunan kandidat

  Himpunan yang berisi elemen-elemen pembentuk solusi. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

- Himpunan solusi

  Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Himpunan solusi adalah himpunan bagian dari himpunan kandidat.

- Fungsi seleksi

  Fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya. Biasanya setiap kandidat, x, di-assign sebuah niilai numerik, dan fungsi seleksi memilih x yang mempunyai bilangan nilai terbesar atau memilih x yang memiliki nilai terkecil.

- Fungsi kelayakan

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

- Fungsi objektif

  Yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi. Untuk menentukan solusi algoritma *greedy* memiliki kendala (constraint) dan fungsi optimasi. Solusi yang memenuhi semua kendala disebut solusi layak, dan solusi layak yang mengoptimumkan fungsi optimasi disebut solusi optimum.

B. Sketsa Umum Algoritma Greedy

Semua algoritma *greedy* mempunyai skema umum yang sama. Secara umum, skema algoritma *greedy* dapat dirumuskan sebagai berikut:

1. Inisialisasi S dengan nilai kosong
2. Pilih sebuah kandidat dengan fungsi seleksi dari C
3. Kurangi C dengan kandidat yang sudah dilih dari langkah (2) di atas
4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak atau feasible (dilakukan dengan fungsi kelayakan). Jika ya, masukan kandidat tersebut ke dalam himpunan solusi. Jika tidak, buang kandidat tersebut dan tidak perlu dipertimbangkan lagi
5. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap (dengan menggunakan fungsi solusi). Jika ya, berhenti (selesai). Jika tidak, ulangi dari langkah (2).

C. Pseudocode Algoritma Greedy

```
function greedy(input C:
  himpunan_kandidat)->
  himpunan_kandidat

{Mengembalikan solusi dari persoalan
  optimasi dengan algoritma greedy
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi yang
  bertipe himpunan_kandidat}

Deklarasi
  x : kandidat
  S : himpunan_kandidat

Algoritma:
  S <- {}
  {inisialisasi S dengan kosong}

  while (not SOLUSI(S)) and (C ¹ {}
  ) do
    x <- SELEKSI(C)     { pilih
  sebuah kandidat dari C}
    C <- C - {x}  { elemen himpunan
  kandidat berkurang satu }
  if LAYAK(S È {x}) then
    S <- S È {x}
    endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then
    return S
  else
    write('tidak ada solusi')
  endif
```

- Pada akhir setiap lelaran, solusi yang terbentuk adalah optimum lokal. Pada akhir kalang while-do diperoleh optimum global.
- Namum adakalanya optimum global merupkan solusi sub-optimum atau pseudo-optimum. Alasan:
  1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode exhaustive search)
  2. Pemilihan fungsi SELEKSI mungkin saja terdapat beberapa fungsi SELEKSI yang berbeda, sehingga kita harus memilih

fungsi yang tepat jika kita ingin algoritma bekerja dengan benar dan menghasilkan solusi yang benar-benar optimum

- Karena itu, pada sebagian masalah algoritma *greedy* tidak selalu berhail menemukan solusi yang benar-benar optimum.
- Jika jawaban terbaik mutlak (benar-benar optimum) tidak diperlukan, maka algoritma *greedy* sering berguna untuk menghasilkan solusi yang menghampiri (approximation) optimum, daripada menggunakan algoritma yang lebih rumit untuk menghasilkan solusi yang eksak.
- Bila algoritma *greedy* optimum, maka keoptimalan itu dapat dibuktikan secara matematis.

III. PENERAPAN ALGORITMA

Permasalahan yang dihadapi dalam makalah ini adalah tentang bagaimana sebuah mesin pengisi bahan bakar dapat memberikan kembalian dalam pecahan yang optimal. Untuk itu kita akan menggunakan prinsip *Greedy*.

Greedy yang akan dilakukan adalah *Greedy* dengan terhadap nilai nominal uang. Nilai nominal yang akan diambil adalah nilai nominal terbesar. Dengan harapan bahwa nilai yang terbesar diambil agar jumlah uang yang nilainya kecil tidak terlalu banyak. Misal kembalian 1000 jika ada satu koin lima ratus lebih optimal daripada 2 buah koin 500.

Misalkan uang yang digunakan dalam mesin hanya pecahan 100, 200, 500, dan 1000. Jika kita mengisi bahan bakar dengan nilai total mencapai 8100. Pembeli membayar ke mesin dengan uang sebesar 10000. Maka kembalian yang akan didapat adalah sebesar 1900.

Dengan algoritma *Greedy* kita akan menguraikan uang tersebut dengan langkah-langkah sebagai berikut:

1. pilih 1 buah koin 1000 rupiah sebagai nilai tertinggi yang masuk kedalam fungsi kelayakan. Jadi kembalian tinggal 900.
2. Pilih pecahan yang paling besar tetapi kurang dari kembalian yang tersisa. Pecahan 500 yang kita ambil.
3. Kemudian 200
4. Dan 200

1900 = 1000 + 500 + 200 + 200

Hasilnya kita akan mendapat 1 buah koin 1000, 1 buah koin 500, dan 2 buah koin 200. Pembeli akan mendapatkan 4 buah koin.

Algoritma *Greedy* memberikan solusi yang optimal. Kita coba lagi dengan kembalian contoh lain misalnya kita membeli bahan bakar dengan harga total 3200. Pembeli membayar denagn uang sebesar 5000 rupiah. Kita akan mencoba dengan cara yang sama. Seharusnya pembeli menerima kembalian sebesar 1800.

1800= 1000 + 500 + 200 + 100

Jadi kita punya 1 koin 1000 ,1 koin 500, 1 koin 200 dan 1 koin 100.

Jumlah koin kembalian adalah 4 buah.

Apabila diperiksa semua kemungkinan pilihan (*exhaustive search*) akan didapat kemungkinan optimal yang seperti di atas. Dapat diartikan bahwa solusi *greedy* tersebut merupakan solusi yang optimal.

Algoritma *greedy* tidak selalu memeberikan solusi yang optimal. Diberikan contoh sebagai berikut

Misal nilai pecahannya 100 , 400, 600 , 1000.

Pembeli membeli bahan bakar seharga 3800 dan membayar dengan 5000 maka pembeli menerima kembalian 1200.

Jika menggunakan algoritma *Greedy* kita akan mengambil pecahan 1000 rupiah lalu pecahan 100 rupiah, lau 100 rupiah lagi. Jadi pecahan yang diambil

1200 = 1000 + 100 + 100

Pembeli mendapatkan 3 koin. Dengan *exhaustive search* kita mendapatkan solusi optimal dengan dua koin yaitu 2 buah koin 600. *Greedy* tidak mendapatkan solusi optimal.

Dari contoh tersebut dapat disimpulkan bahwa

algoritma *greedy* tak pasti memberikan solusi optimal.

Memang untuk kasus ketiga memang tidak optimal. Akan tetapi, setelah diteliti lebih lanjut kasus ketiga tidak terjadi optimal karena pecahan koinnya. Pecahan koin memenuhi syarat agar selalu optimal adalah jika koin dikali 2 tidak akan lebih besar dari koin sesudahnya.

Contoh koin 200 jika ditambah koin 200 lebih kecil dari koin 500. Ini adalah contoh pecahan yang memenuhi syarat.

Contoh lainnya 400 jika ditambahkan 400 maka lebih besar dari 600.

Jadi jika koin yang digunakan pecahannya memenuhi syarat terbukti bahwa akan selalu optimal pada semua kasus.

Pseudo-code untuk algoritma *Greedy*

```
Function kembalian (input C
  :himpunan pecahan koin, A:
  integer)-> himpunan koin

Deklarasi
  S: himpunan koin
  X: koin

Algoritma
  S<-{}
  While ((nilai semua koin di dalam
  S) != A) and (C!= {}) do
    X <- koin dengan nilai terbesar
    C <- C-{X}
    If ((nilai semua koin didalam S)
      + nilai koin X < A) then
      S <- S U {X}
    Endif
  Endwhile
  If (nilai semua koin didalam S==A)
   then
    Return S
  Else
    Write ('tak ada solusi')
  Endif
```

IV. Conclusion

Algoritma *greedy* dapat dipakai dalam optimasi permasalahan uang kembalian.

Pengoptimalan disini berarti jumlah koin yang didapat berjumlah paling sedikit dari semua kemungkinan yang ada.

Algoritma *greedy* tidak selalu memberikan solusi optimal untuk semua kasus pecahan yang digunakan. Karena kepastian optimasi tergantung dari himpunan pecahan koin dari mata uang tersebut.

Algoritma *greedy* akan selalu memberikan solusi optimal apabila terdapat dua pecahan yang berurutan besarny apabila pecahan yang lebih kecil dikali 2 hasilnya tidak akan lebih besar dari nilai pecahan sesudahnya.

V. Ucapan Terima Kasih

The Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena berkat anugerah yang diberikan-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Ir. Rinaldi Munir, M.T. selaku dosen pengajar kuliah IF3051 Strategi Algoritma karena berkat kuliah yang diberikan dan buku diktat yang ditulis oleh beliau makalah ini dapat disempurnakan.

Referensi

[1] Munir, Rinaldi. Diktat Kuliah Strategi Algoritmik IF3051 Strategi Algoritmik. Departemen Teknik Informatika ITB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 7 Desember 2010

Rifky Hamdani (13508024)