Kompleksitas Algoritma untuk Penyelesaian Persoalan Penukaran Koin dengan Algoritma Greedy

Dita Anindhika 13509023[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509023@std.stei.itb.ac.id

***Abstrak*—Dalam melakukan pemrograman untuk mengimplementasikan suatu rancangan perangkat lunak, dapat ditempuh dengan berbagai strategi algoritma. Banyaknya algoritma yang dapat digunakan tersebut menimbulkan suatu pertanyaan yaitu algoritma manakah yang paling baik untuk menyelesaikan persoalan tersebut. Untuk itulah diperlukan suatu model abstrak pengukuran ruang/waktu yang terbebas dari pertimbangan arsitektur komputer dan compiler bahasa pemrograman. Model abstrak ini dikenal dengan sebutan kompleksitas algoritma. Pada makalah ini akan dibahas bagaimana kompleksitas algoritma dalam menyelesaikan persoalan penukaran koin dengan menggunakan algoritma greedy. Selain itu juga dijbahas bagaimana implementasi algoritma tersebut pada program sederhana yang dapat mengatasi persoalan penukaran uang dalam bahasa C.**

***Index Terms*—Algoritma Greedy, Kompleksitas Algoritma, Persoalan Penukaran Koin.**

I. PENDAHULUAN

Algoritma adalah urutan langkah-langkah dalam memecahkan suatu masalah. Dalam melakukan pemrograman untuk mengimplementasikan suatu rancangan perangkat lunak, dapat ditempuh dengan berbagai strategi algoritma. Strategi Algoritma merupakan kumpulan metode atau teknik untuk memecahkan masalah guna mencapai tujuan yang ditentukan, yang dalam hal ini deskripsi metode atau teknik tersebut dinyatakan dalam suatu urutan langkah-langkah penyelesaian.

Secara umum, strategi pemecahan masalah dapat dikelompokan sebagai berikut:

1. Strategi solusi langsung (direct solution strategies)
    a. Algoritma Brute force
    b. Algoritma Greedy
2. Strategi berbasis pencarian pada ruang status (state-space base strategies)
    a. Algoritma Backtracking
    b. Algoritma Branch and Bound
3. Strategi solusi atas-bawah (top-down solution strategies)
    a. Algoritma Divide and Conquer.
4. Strategi solusi bawah-atas (bottom-up solution strategies)
    a. Dynamic Programming

Pada makalah ini akan dibahas bagaimana kompleksitas algoritma dalam penyelesaian persoalan penukaran koin dengan salah satu strategi algoritma, yaitu algortima greedy. Pemilihan algoritma greedy untuk penyelesaian persoalan penukaran koin dilakukan untuk mengukur seberapa optimum algoritma greedy dalam penyelesaian persoalan ini karena permasalahan yang sering muncul dari penggunaan algoritma greedy ini adalah penyelesaian persoalan terjebak di optimum lokal dan tidak mencapai optimum global. Selain pembahasan mengenai kompleksitas algoritma tersebut, dibahas pula bagaimana implementasi algoritma greedy pada program sederhana yang dapat mengatasi pesoalan penukara koin dalam bahasa C.

II. DASAR TEORI

A. Kompleksitas Algoritma

Kita sering bertanya mengenai algortima mana yang lebih baik dalam menyelesaikan suatu masalah. Untuk menjawab masalah tersebut tentunya dibutuhkan suatu alat ukur agar kita bisa menilai apakah algoritma tersebut lebih baik atau tidak.

Apabila kita mencoba mengeksekusi program dengan algoritma A pada suatu komputer dan program pada algoritma B pada komputer lainnya, tidak dapat mengatakan algoritma A lebih baik dibandingkan dengan algoritma B hanya karena program dengan algoritma A jauh lebih cepat dieksekusi. Tidak semua komputer yang digunakan memiliki arsitektur yang sama, sehingga waktu komputasinya pun juga berbeda. Begitupula dengan compiler bahasa pemrograman dalam menghasilkan kode mesin. Sehingga pada kasus di atas, bisa jadi program dengan algoritma A jauh lebih cepat dieksekusi dikarenakan arsitektur komputer yang satu lebih baik dibandingkan arsitektur komputer lainnya.

Oleh karena itu kita memerlukan model abstrak pengukuran waktu/ruang yang bebas dari pertimbangan arsitektur komputer dan compiler bahasa pemrograman. Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma.

Ada dua macam kompleksitas algoritma, yaitu :

kompleksitas waktu dan kompleksitas ruang. Kompleksitas waktu disimbolkan dengan T(n) dan kompleksitas ruang S(n). Kompleksitas waktu, T(n), diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankkan algoritma sebagai fungsi dari ukuran masukan n. Komplesitas ruang, S(n), diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n.

Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan laju peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n.1

Di dalam sebuah algoritma terdapat bermacam jenis operasi:

- Operasi baca/tulis,
- Operasi aritmetika (+, -, *, /)
- Operasi pengisian nilai (assignment)
- Operasi pengaksesan elemen larik
- Operasi pemanggilan fungsi/prosedur
- dll

Dalam prakteknya, yang dihitung hanyalah jumlah operasi khas (tipikal) yang mendasari suatu algoritma.

Terkadang tidak terlalu dibutuhkan kompleksitas waktu yang detil dari sebuah algoritma. Yang dibutuhkan terkadang adalah besaran kompleksitas waktu yang menghampiri kompleksitas waktu yang sebenarnya. Kompleksitas waktu yang demikian disebut kompleksitas waktu asimptotik yang dinotasikan dengan "O" (baca : "O-Besar"). Kompleksitas waktu asimptotik diperoleh dengan mengambil term yang memberikan kompleksitas waktu terbesar. Misalkan T(n) = 3n3 + 2n2 + n + 1. Maka kompleksitas waktu asimptotiknya adalah $O(n^3)$. Karena n^3 yang memberikan kompleksitas waktu terbesar. Tidak perlu ditmbahkan pengali dari term dari n^3.

B. Algoritma Greedy

Algoritma Greedy merupakan algoritma yang paling sering digunakan dalam menyelesaikan masalah optimasi. Walaupun dengan menggunakan Algoritma Greedy belum tentu didapatkan hasil yang paling optimum (optimum global), algoritma ini tetap digunakan sebagai basis untuk pendekatan heuristik. Algortima ini membentuk solusi tersebut dengan langkah demi langkah (step by step). Tentunya akan banyak kemungkinan langkah yang dapat ditelusuri untuk mencapai solusi, maka dengan algoritma Greedy ini akan dilakukannya pengambilan keputusan yang memenuhi suatu kondisi optimum lokal yang diharapkan nantinya dapat menghasilkan suatu solusi yang optimum global.

Persoalan optimasi pada algoritma Greedy disusun oleh berbagai elemen yang dijelaskan sebagai berikut :

1. Himpunan kandidat
   Himpunan ini berisi elemen-elemen yang mungkin mejadi kandidat solusi
2. Himpunan solusi
   Himpunan ini berisi elemen-elemen yang terpilih sebagai solusi persoalan
3. Fungsi seleksi
   Fungsi yang memilih kandidat untuk menjadi sebuah solusi yang dianggap optimal
4. Fungsi kelayakan
   Fungsi yang menilai layak atau tidaknya suatu kandidat solusi sehingga tidak melanggar constraint-constraint yang ada dalam penyelesaian masalah.
5. Fungsi objektif
   Fungsi yang meminimalkan atau memaksimumkan nilai solusi

Dibawah ini merupakan skema umum 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 ≠ {} ) do
      x ← SELEKSI(C)        {   pilih   sebuah
    kandidat dari C}
      C ← C - {x}             { elemen himpunan
    kandidat berkurang satu }
       if LAYAK(S ∪ {x}) then
           S ← S ∪ {x}
       endif
    endwhile
    {SOLUSI(S) or C = {} }

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

Dengan kata lain, algoritma Greedy akan mencari sebuah himpunan solusi, S, yang merupakan himpunan bagian dari himpunan kandidat, C. Dalam hal ini, himpunan S merupakan himpunan kandidat, C, yang dikenai fungsi seleksi dan kelayakan. Dan solusi akhir yang diambil adalah himpunan solusi, S, yang dikenai fungsi objektif. Pilihan yang dibuat menggunakan algoritma Greedy ditentukan oleh pilihan-pilihan yang telah dibuat sampai saat ini namun tidak oleh pilihan-pilihan yang akan datang. Secara iteratif, pilihan Greedy dilakukan sehingga membuat permasalahan yang ada menjadi permasalahan yang lebih kecil. Perbedaan mendasar antara algoritma Greedy dengan program dinamis adalah pada algoritma Greedy tidak pernah merubah pilihan yang telah dibuat sebelumnya. Perbedaan lain adalah pada algoritma Greedy hanya dilakukan perhitungan untuk sebuah kemungkinan solusi saja sedangkan pada program dinamis akan dilakukan perhitungan untuk banyak kemungkinan solusi.

III. PERSOALAN PENUKARAN KOIN

Misalnya terdapat A nilai uang yang ditukar. Dengan himpunan koin $\{d_1,d_2,d_3, \ldots ,d_n\}$ dan himpunan solusi $X = \{x_1x_2,x_3, \ldots , x_n\}$ dimana $x_i = 1$ jika dipilih dan $x_i = 0$ jika tidak dipilih.

Maka objektif persoalan diatas adalah

minimisasi $F = \sum_{i=1}^{n} x_i$ (fungsi obyektif)

dengan kendala $\sum_{i=1}^{n} d_i x_i = A$

Ide utama dalam menyelesaikan persoalan dengan menggunakan algoritma greedy adalah pada setiap langkah dilakukan pemilihan koin dengan nilai terbesar dari himpunan koin yang tersisa. Berikut ini merupakan algoritma greedy untuk menyelesaikan persoalan tersebut.

```
function CoinExchange(input C : himpunan_koin, A
: integer) → himpunan_koin
{ mengembalikan koin-koin yang total nilainya =
A, tetapi jumlah koinnya minimum }

Deklarasi
   S : himpunan_koin
   x : koin

Algoritma
  S ← {}
  while (∑(nilai semua koin di dalam S) ≠ A) and
(C ≠ {} ) do
     x ← koin yang mempunyai nilai terbesar
     C ← C - {x}
     if  (∑(nilai semua koin di dalam S) + nilai
koin x ≤ A then
       S ← S ∪ {x}
     endif
  endwhile

  if (∑(nilai semua koin di dalam S) = A then
     return S
  else
     write('tidak ada solusi')
  endif
```

Kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi perbandingan elemen larik ((∑(nilai semua koin di dalam S) + nilai koin x ≤ A dan ∑(nilai semua koin di dalam S) = A).

Sehingga berdasarkan hal tersebut kompleksitas waktu

$$T(n) = n + n + n + n + n.$$
$$= 5n$$

Karena nilai $T(n) = 5n$, maka didapati nilai

$O(n)$.

Namun, algoritma tersebut hanya dapat diterapkan pada pecahan koin yang sudah terurut mengecil. Untuk itu, dalam menyelesaikan persoalan penukaran koin ini, perlu ditambahkan algoritma sorting untuk menangani kasus pecahan koin yang belum terurut.

```
procedure Sort(input/output C : himpunan_koin, n
: integer)

Deklarasi
   Hold,j,i : integer

Algoritma
for j←0 to n-1  do
    for i←0 to n-2 do
      if Cj+1 < Cj+2 then
        hold←Cj+1
        Cj+1←Cj+2
        Cj+2←hold

      endif
    endfor
  endfor
```

Jumlah operasi perbandingan elemen pada algoritma sorting diatas dijelaskan sebagai berikut.

Untuk setiap *pass* ke-*j*,

$j = A–2 \rightarrow$ jumlah perbandingan $= n – 1$

$j = A–1 \rightarrow$ jumlah perbandingan $= n – 2$

$j = n– 2 \rightarrow$ jumlah perbandingan $= n – 3$

$\vdots$

$j = 2 \rightarrow$ jumlah perbandingan $= 1$

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah

$$T(n) = (n–1)+(n–2)+\ldots+1$$
$$= \sum_{i=1}^{n-1} n - k = \frac{n(n-1)}{2}$$

Sedangkan jumlah operasi pertukarannya untuk setiap *i* dari 1 sampai $n – 1$, terjadi satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya adalah

$$T(n) = n – 1.$$

Jadi, algoritma pengurutan seleksi membutuhkan $n(n – 1)/2$ buah operasi perbandingan elemen dan $n – 1$ buah operasi pertukaran. Menjadikan nilai

$$T(n) = (n^2– n)/2$$

Karena nilai $T(n) = (n^2– n)/2$, maka didapati nilai

$O(n^2)$.

Maka untuk penyelesaian persoalan penukaran koin yang belum terurut mengecil kompleksitas waktu

$$T(n) = 5n +(n^2– n)/2$$

Hal tersebut menjadikan nilai

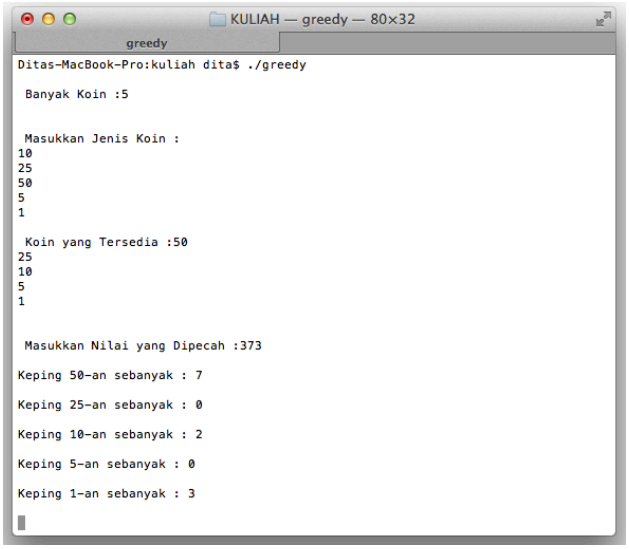
$O(n^2)$.

IV. Implementasi Algoritma Greedy Dalam Penyelesaian Persoalan Penukaran Koin dalam Bahasa C

Dengan menggunakan bahasa C, dilakukan pembuatan program sederhana dengan menggunakan algoritma Greedy yang dapat mengatasi persoalan penukaran koin.

```
  #include <stdlib.h>
  #include <stdio.h>

  #define size 99
  void sort(int[], int);

  int main()
  {
        int x[size],i,n,uang,hasil[size], t;
        printf("\n Banyak Koin :");
        scanf("%d", &n);
        printf("\n \n Masukkan Jenis Koin : \n");

        for(i=1;i<=n;i++)
        {
                scanf("%d", &x[i]);
        }

        sort(x,n);
        printf("\n Koin yang Tersedia : \n");

        for(i=1;i<=n;i++)
        {
        printf("%d \n", x[i]);
        }

        printf("\n \n Masukkan Nilai yang Dipecah
:");
        scanf("%d", &uang);
        printf("\n");

        /*-------Algoritma Greedy--------*/
        for(i=1;i<=n;i++)
        {
                if uang>=x[i] then
                   uang=uang-x[i];
                   hasil[i]=hasil[i]+1;
                else i++;

        }
        /*-------Algoritma Greedy--------*/

        for(i=1;i<=n;i++)
        {
                printf("Keping  %d-an  sebanyak  :
%d \n \n", x[i], hasil[i]);
        }
  scanf("%d",&t);
  return 0;
  }

  /* Prosedur sort */
  void sort(int a[], int siz)
  {
        int pass,hold,j;
        for(pass=1;pass<=siz-1;pass++)
        {
                for(j=0;j<=siz-2;j++)
                {
                        if(a[j+1] < a[j+2])
                        {
                                hold=a[j+1];
                                a[j+1]=a[j+2];
                                a[j+2]=hold;
                        }
                }
        }
  }
```

Kode diatas merupakan *source code* dari program sederhana dalam bahasa C++ yang menggunakan algoritma Greedy untuk menyelesaikan persoalan penukaran koin. Pada source code terlihat bahwa nilai uang yang ingin ditukar akan dicek apakah lebih besar dari nilai pecahan koin terbesar. Apabila lebih besar sama dengan maka nilai uang yang ingin ditukar tersebut dikurangi dengan nilai pecahan tersebut. Apabila nilai uang yang ingin ditukar lebih kecil dari nilai pecahan koin, maka counter i akan ditambah.



Gambar 1 Hasil Implementasi Algoritma Greedy pada Penyelesaian Persoalan Penukaran Uang dengan C

Setelah program tersebut dijalankan maka akan meminta masukan banyaknya pecahan uang dan nilai masing-masing pecahan uang tersebut. Setelah itu program akan menampilkan urutan hasil sorting dari pecahan uang yang tersedia. Pada gambar, nampak masukan banyaknya koin adalah 5 dengan nilai pecahan 10,25, 50, 5, dan 1.

Kemudian program akan meminta masukan nilai uang yang ingin dipecah, pada gambar nampak nilai uang yang ingin dipecah adalah 373. Setelah itu program akan menampilkan jumlah pecahan uang yang dibutuhkan untuk masing-masing pecahan. Untuk nilai uang 373 dibutuhkan pecahan uang 50 sebanyak 7 buah, pecahan uang 25 sebanyak 0 buah pecahan uang 10 sebanyak 2 buah, pecahan uang 5 sebanyak 0 buah, dan pecahan uang 1 sebanyak 3 buah.

V. Kesimpulan

Kompleksitas algoritma untuk penyelesaian persoalan penukaran koin dengan algoritma greedy apabila nilai pecahan koin sudah terurut mengecil adalah $O(n)$ dengan kompleksitas waktu $T(n) = 5n$. Apabila pecahan koin belum terurut mengecil maka kompleksitas

algoritma untuk penyelesaian persoalan penukaran koin dengan algoritma greedy adalah $O(n^2)$ dengan kompleksitas waktu $T(n) = 5n + (n^2 - n)/2$.

VII. Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas selesainya makalah ini. Tidak lupa penulis juga mengucapkan terima kasih kepada bapak Rinaldi Munir dan ibu Harlili selaku pengajar telah memberikan kesempatan bagi penulis untuk menuntut ilmu sebanyakbanyaknya. Selain itu, ucapan terima kasih juga diberikan kepada orang tua dan teman-teman informatika, atas semangat dan dukungannya dalam menjalani hari-hari kuliah. Akhir kata, penulis berharap karya tulis ini dapat bermanfaat bagi orang lain dan ilmunya dapat disebarluaskan.

Referensi

[1] Munir, Rinaldi. Diktat Kuliah IF2091 Struktur Diskrit. Edisi keempat. Program Studi Teknik Informatika, Institut Teknologi Bandung

[2] Munir. Rinaldi, Diktat Kuliah IF3051 Strategi Algoritma, Bandung, 2009, hal : 41-49

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, 18 Desember 2012

Dita Anindhika 13509023