Penggunaan Beberapa Algoritma dalam Menentukan Kelompok

Eric Christopher / 13509037
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
eric.c13@students.itb.ac.id

Brute Force, divide and conquer, program dinamis yang dimaksud disini adalah algoritma pemecahan masalah.

Index: algoritma, brute force, divide and conquer, program dinamis, penentuan kelompok.

I. Latar Belakang

Pertama-tama penulis akan menjelaskan masalah yang penulis angkat. Penulis dalam menulis makalah ini terinspirasi dari salah satu mata kuliah semester 6 jurusan informatika Institut Teknologi Bandung dimana pada saat kuliah terdapat 5 tugas berkelompok dan pada tiap tugas kelompok yang didaftarkan anggota-anggotanya tidak boleh pernah sekelompok dengan angoota lainnya di kelompok-kelompok sebelumnya. Jadi dengan kata lain definisi permasalahannya adalah jika terdapat sekelompok orang dan mereka harus dibagi ke kelompok-kelompok kecil, kemudian setelah dibagi ada suatu saat lagi kelompok orang tadi harus dibagi lagi ke kelompok-kelompok kecil dimana orang-orang yang sudah sekelompok pada kelompok pertama tidak boleh sekelompok lagi pada kelompok ke dua dan seterusnya.

Pada makalah ini penulis ingin menggunakan algoritma brute force dan beberapa algoritma lain untuk mengecek apakah kelompok-kelompok yang akan dibentuk adalah valid. Selain itu penulis juga akan membahas pengembangan dari algoritma brute force tadi agar performansinya meningkat karena pada umumnya algoritma brute force memang mampu memecahkan masalah namun terdapat kelemahan di performansi maupun efisiensi.

Dalam makalah ini akan diambil contoh kelompok yang tidak terlalu rumit yaitu 1 kelompok memiliki maksimum 3 orang anggota. Untuk kelompok yang memiliki jumlah anggota yang lebih besar dapat dipecahkan dengan cara yang sama hanya beda di ukuran saja.

Selain itu walau di dalam makalah ini terlihat fokus membicarakan tentang algoritma brute force namun penulis juga akan membahas penyelesaian masalah pembagian kelompok ini dengan menggunakan algoritma-algoritma lain sebagai pembandingan.

II. Dasar Teori

Kata algoritma berasal dari atinisasi nama seorang ahli matematika dari Uzbekistan Al Khawārizmi (hidup sekitar abad ke-9), seperti yang ada di karyanya pada abad ke-12 dalam bahasa latin "Algorithmi de numero Indorum". Pada abad ke-18, istilah ini berkembang menjadi **algoritma**, yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan.

Dalam matematika dan komputasi, **algoritma** atau **algo ritme** merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Di Wikipedia terdapat beberapa jenis algoritma, tetapi yang di singgung hanyalah 3 yaitu divide and conquer jadi membagi suatu masalah menjadi masalah-masalah kecil sampai tidak bisa dibagi lagi kemudian barulah menyelesaikan masalah-masalah kecil tersebut satu persatu. Kedua adalah dynamic programming secara sekilas algoritma ini mirip dengan divide and conquer tetapi ada perbedaan dalam hal karakter permasalahan yang dihadapi. Terakhir adalah metode serakah yaitu jawaban dari submasalah tidak perlu diketahui dalam setiap tahap, tetapi hanya memilih pilihan "serakah" apa yang terbaik digunakan untuk menyelesaikan masalah yang ada.

Pada sumber lain penulis juga menemukan penjelasan yang lebih mendetil dari algoritma-algoritma di atas. Pada algoritma divide and conquer terdapat 3 tahap yaitu pertama kita bagi persoalan tadi menjadi persoalan yang lebih kecil, kemudian persoalan-persoalan yang kecil tadi akan dipecahkan semua satu per satu. Kemudian perlahan-lahan hasil dari persoalan-persoalan tadi akan digabungkan kembali dan akan menjadi solusi dari persoalan awalnya sebelum dipecah. Pada dasarnya divide and conquer atau yang biasa disingkat D&C ini menggunakan prinsip rekursi dalam hal membagi serta menyatukan kembali solusi yang ada.

Untuk program dinamis, algoritma ini merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah atau tahapan.

Kemudian tiap langkah dikembangkan sehingga menjadi langkah berikutnya yang lebih besar. Hal ini dilakukan terus menerus sampai tahap terakhir merupakan masalah yang ingin dihadapi. Sebenarnya program dinamis ini adalah turunan dari algoritma divide and conquer karena algoritma ini memecahkan suatu permasalah menjadi lebih kecil kemudian mencari solusinya dan terakhir menggabungkannya menjadi solusi dari permasalahan yang semula belum dipecah. Hanya modelnya lebih ke tahap atau langkah (sekuensial) daripada ke perbandingan (paralel).

Metode serakah biasa disebut algoritma greedy. Algoritma ini merupakan metode yang paling popular dalam memecahkan masalah optimasi (menentukan maksimum maupun minimum). Namun sayangnya algoritma ini tidak menghasilkan solusi optimal untuk beberapa permasalahan.

Selain itu terdapat juga algoritma brute force. Algoritma Brute Force adalah algoritma yang paling simple dan dapat digunakan untuk memecahkan hampir semua masalah. Namun algoritma ini kurang disukai karena ketidak mangkusannya. Untuk masalah kecil mungkin tidak kelihatan perbedaan dengan menggunakan algoritma ini atau tidak tetapi jika sudah menyangkut masalah yang besar maka algoritma ini sudah tidak mungkin dipakai lagi karena ketidakmangkusannya.

Kompleksitas dari suatu **algoritma** merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

III. Pembahasan

Pertama-tama penulis akan mereview masalah yang diangkat di dalam karya tulis ini. Penentuan kelompok dimana tidak ada satupun anggota dalam kelompok yang pernah sekelompok sebelumnya dengan metode brute force sebagai pembanding.

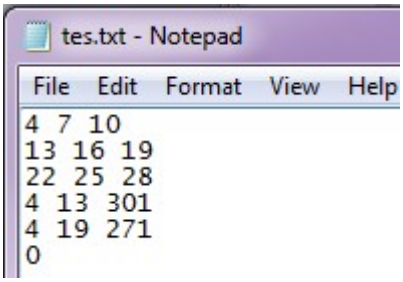
Setelah ditelaah akhirnya penulis menemukan algoritma brute force yang cocok dengan masalah ini. Pertama-tama simpan dulu masukkan kelompok-kelompok sebelumnya disebuah array. Kemudian kelompok baru masukkan user akan dicocokkan dengan array tadi dan akan dicek apakah kelompok baru tadi boleh masuk atau tidak.

Cara pengecekkannya adalah dengan memeriksa tiap array sampai habis atau ditemukan orang yang pernah sekelompok. Jadi dengan teknik pencocokkan masing-masing orang dengan kelompok yang sudah ada, apakah dia pernah sekelompok dengan teman kelompoknya yang sekarang atau belum.

Sebelum membuat makalah ini penulis telah membuat program yang memecahkan masalah yang terdapat dalam makalah ini. Dalam program tersebut akan ditanya file apa yang akan menjadi input sehingga tidak perlu memasukkan satu persatu dari nol. File tersebut juga akan menjadi file temapat save setelah data kelompok-kelompok yang baru dimasukkan. Dalam membaca file dan menyimpan file penulis juga menggunakan enkripsi sederhana dengan menyimpan angka yang merupakan tiga kalinya plus satunya dari angka yang asli.

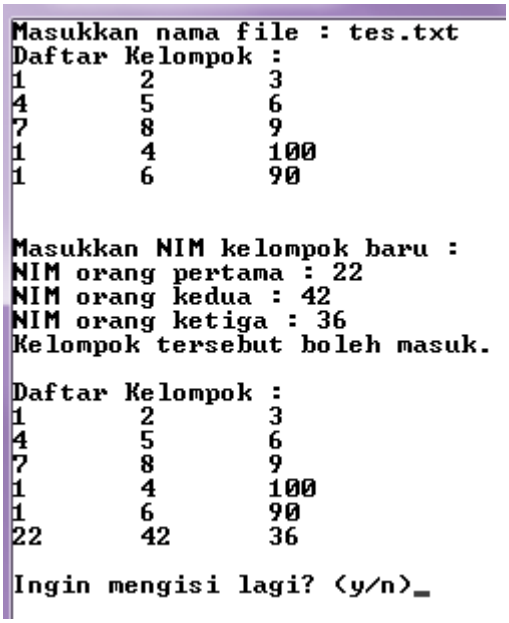
Kemudian dengan membandingkan semua kemungkinan di tiap kelompok yang ada penulis mengecek apakah kelompok baru tersebut bisa masuk atau tidak. Jika bisa maka langsung ditambahkan ke daftar kelompok jika tidak maka akan diberi tahu dan kelompok yang baru tadi tidak akan ditambahkan di daftar kelompok. Kemudian untuk memudahkan penulis menggunakan NIM saja bukan menggunakan nama karena kemungkinan terdapat dua orang yang memiliki nama yang sama, tetapi jika menggunakan NIM tidak akan ada dua orang yang memiliki NIM yang sama.

Berikut adalah contoh isi dari file masukkan.



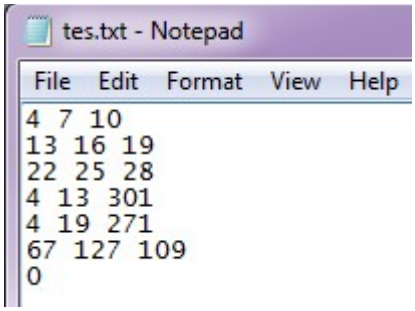
Tanda nol disini adalah end of file sehingga setiap file akan diakhiri oleh angka 0.

Kemudian berikut ini adalah tampilan ketika program dijalankan. Berikut adalah tampilan program yang telah penulis buat:

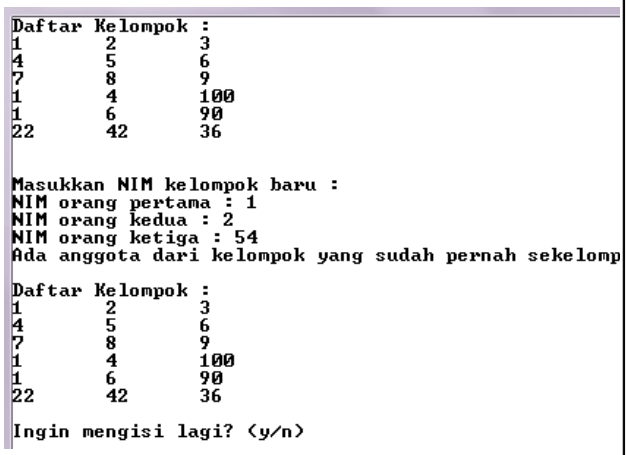


Jadi setelah dicek apakah anggota kelompok tersebut boleh masuk, maka langsung dimasukkan ke daftar kelompok supaya tidak lupa dicek bahwa anggota dengan NIM 22, 42, 36 pernah sekelompok sebelumnya pada saat input anggota kelompok berikutnya.

Kemudian berikut adalah tampilan dari file masukkan ketika program telah selesai dijalankan.



Kemudian jika ada masukkan anggota kelompok yang sudah pernah sekelompok maka berikut ini adalah tampilan dari program penulis.



Jadi anggota kelompok yang baru tidak ditambahkan ke daftar kelompok.

Kompleksitas dari algoritma ini adalah O(n) karena kasus terburuknya adalah ketika anggota kelompok yang dimasukkan adalah valid dan kasus terbaiknya adalah ketika kelompok yang dimasukkan tidak valid dan ditemukannya sewaktu di awal pencocokan.

Algoritma lain adalah divide and conquer jika datanya disimpan ke sebuah array maka dengan divide and conquer dapat dibagi dua terus menerus sehingga sisanya satu dan kemudian dibandingkan. Namun dengan cara begitu tetap saja harus mengecek semua kelompok dan akhirnya menjadi sama dengan kasus terburuk dari brute force yaitu semua array akan diperiksa. Bahkan masih lebih buruk dari brute force karena terdapat usaha untuk membagi dan menggabungkannya kembali.

Berikutnya adalah dinamik programming, karena pada dasarnya dinamik programming mirip dengan divide and conquer maka tetap saja tidak lebih baik dari brute force.

Untuk algoritma greedy juga akan bernasib sama dengan divide and conquer dan dinamik programming. Hal ini disebabkan karena kita perlu mengecek dengan semua kelompok yang ada.

Sebenarnya masih ada lagi beberapa algoritma lain yang mungkin seperti dan setelah dicoba ujung-ujungnya akan memeriksa semua anggotanya untuk mengecek apakah ia sudah pernah sekelompok atau belum sehingga tidak lebih baik dari brute force. Hal ini disebabkan karena inti dari masalah ini adalah membandingkan kelompok yang baru dengan semua kelompok yang ada, sehingga diketahui apakah kelompok itu valid atau tidak.

Berikut ini program sederhana untuk mengetes algoritma brute force yang digunakan.

```
//file : sudoku.cpp
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;
int Kel[256][3];
int T[256];
int n=0;
void Save();
void Load ();
void Tampil();
void Process();
int Isok(int a,int b,int c);
char f[256];

void Save()
{
        string s;
        fstream OutFile;
        OutFile.open(f, fstream::out | fstream::trunc);
   int i=0;
   while(i<n)
   {
      OutFile << (Kel[i][0]*3+1) << " " << (Kel[i][1]*3+1) << " " <<
(Kel[i][2]*3+1) << endl;
       i++;
   }
   OutFile << "0";
   OutFile.close();
}
void Load ()
{
        fstream OutFile;
        string s;
        char* cline, *cc;
        cout<<"Masukkan nama file : ";
        cin>>f;
        //tempat sementara (typecast)
        int i, j, ti;
        cline = new char[256];
        //Open File nya
        OutFile.open(f, fstream::in);
   i=0;
   do
   {
     //ambil line
     getline(OutFile,s);
     strcpy(cline, s.c_str());
     cc = strtok(cline, " "); //load baris pertama
     ti = atoi(cc);
     if(ti!=0)
```

```
        {
          //simpan
          Kel[i][0]=(ti-1)/3;
          n++;
          for(j=1;j<3;j++)
          {
            cc = strtok(NULL, " ");
            ti = atoi(cc);
            //simpan
            Kel[i][j]=(ti-1)/3;
          }
        }
        i++;
    }while(ti!=0);
         OutFile.close();
}
void Tampil()
{
    int i=0;
    cout<<"Daftar Kelompok : \n";
    while(i<n)
    {
        for(int j=0;j<3;j++)
        {
            cout<<Kel[i][j]<<"\t";
        }
        cout<<"\n";
        i++;
    }
}
void Process()
{
    int a,b,c;
    int ulang=1;
    int ulang2=1;
    int ulang3=1;
    char p;
    while(ulang2==1)
    {
        ulang3=1;
        while(ulang==1)
        {
            Tampil();
            cout<<"\n\nMasukkan NIM kelompok baru :\n";
            cout<<"NIM orang pertama : ";
            cin>>a;
            cout<<"NIM orang kedua : ";
            cin>>b;
            cout<<"NIM orang ketiga : ";
            cin>>c;
            ulang=0;
            if(a==b || a==c || b==c || a<1 || b<1 || c<1)
            {
                system("cls");
                cout<<"Maaf input tidak valid\n\n";
                ulang=1;
            }
        }
        if(Isok(a,b,c)==1)
        {
            cout<<"Kelompok tersebut boleh masuk.\n\n";
            Kel[n][0]=a;
            Kel[n][1]=b;
            Kel[n][2]=c;
            n++;
            Tampil();
        }
        else
        {
                cout<<"Ada anggota dari kelompok yang sudah pernah sekelompok.\n\n";
                Tampil();
            }
            ulang2=0;
            while(ulang3==1)
            {
                cout<<"\nIngin mengisi lagi? (y/n)";
                cin>>p;
                if(p=='y')
                {
                    ulang3=0;
                    ulang2=1;
                    ulang=1;
                    system("cls");
                }
                else if(p=='n')
                {
                    ulang3=0;
                    system("cls");
                    cout<<"Terima kasih.";
                }
            }
        }
    }
    int Isok(int a,int b,int c)
    {
        int i=0;
        int benar=1;
        while(i<n && benar==1)
        {
            if(Kel[i][0]==a)
            {
                if(Kel[i][1]==b || Kel[i][2]==b || Kel[i][1]==c || Kel[i][2]==c)
                {
                    benar=0;
                }
            }
            else if(Kel[i][1]==a)
            {
                if(Kel[i][0]==b || Kel[i][2]==b || Kel[i][0]==c || Kel[i][2]==c)
                {
                    benar=0;
                }
            }
            else if(Kel[i][2]==a)
            {
                if(Kel[i][0]==b || Kel[i][1]==b || Kel[i][0]==c || Kel[i][1]==c)
                {
                    benar=0;
                }
            }
            i++;
        }
        return benar;
    }
    int main()
    {
        Load();
        Process();
        Save();
        return 0;
    }
```

Sebenarnya brute force ini bisa dikembangkan sehingga menjadi lebih sederhana yaitu dicek bila orang pertama dan kedua sudah sekelompok maka tidak perlu mengecek orang-orang berikutnya (misalkan 1 kelompok terdiri dari

lebih dari 3 orang). Ada pun jika sebuah kelompok sudah tidak valid maka tidak perlu membandingkannya dengan kelompok yang lain lagi. Selain itu ada juga cara penyerhanaan algoritma ini adalah dengan cara tidak menggunakan buffer sehingga sambil membaca dari file sambil juga membandingkan, sehingga tidak perlu memori untuk array. Kemudian jika valid maka langsung file tadi isinya ditambah.

IV. Kesimpulan

Jadi kesimpulannya penentuan kelompok pun dapat dipecahkan dengan algoritma brute force dengan cara membandingkan dengan semua kelompok dan dengan semua kemungkinan susunan.

Adapun cara untuk meningkatkan performansi dari algoritma ini, yaitu:

- ➔ Setelah ditemukan kalau kelompok tidak valid maka tidak perlu dicek untuk kelompok-kelompok berikutnya.
- ➔ Tidak menggunakan banyak memory sehingga begitu file dibaca langsung dibandingkan tanpa perlu disimpan.

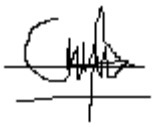
References

[1] Slide Strategi Algoritma 2011
[2] http://id.wikipedia.org/wiki/Algoritma
[3] http://en.wikipedia.org/wiki/Divide_and_conquer_algorithm
[4] http://en.wikipedia.org/wiki/Dynamic_programming
[5] http://en.wikipedia.org/wiki/Greedy_algorithm

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, 4 Desember 2011



Eric Christopher / 13509037