Laporan Tugas Kecil 2 Strategi Algoritma IF2211

Penerapan *Topological Sort* dengan Algoritma *Decrease and Conquer*

Disusun oleh:

Nama: Alvin WIlta

NIM: 13519163

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma Decrease and Conquer

Algoritma *decrease and conquer* adalah sebuah desain algoritma yang bertujuan untuk menyelesaikan suatu permasalahan dengan membagi-bagi permasalahan tersebut menjadi sub-permasalahan yang lebih kecil dan kemudian hanya diselesaikan dari satu demi satu sub-persoalan hingga persoalan tersebut sudah terselesaikan secara keseluruhan. Biasanya algoritma ini diaplikasikan secara rekursif.

Pada tugas kecil ini, desain algoritma divide and conquer diterapkan pada proses penyelesaian suatu permasalahan pengambilan mata kuliah per semesternya dengan memanfaatkan *Topological Sort* pada *Directed Acyclic Graph* (DAG), persoalan ini diselesaikan dengan membagi-bagi graf menjadi lebih kecil dan menyelesaikannya dengan menggunakan *Topological Sort*.

Untuk algoritma dari pencarian solusi adalah:

1. Menelusuri DAG dengan mengiterate graph secara rekursif (mengurangi per node dan mengecek apakah sudah divisit) dan menghitung jumlah inbound pada setiap node
2. Mulai pengulangan
3. Mengecek node mana yang jumlah inboundnya bernilai 0 dan masukkan semua nodenya pada baris matriks hasil
4. Menelusuri graph pada node yang jumlah inboundnya bernilai 0, kemudian simpan node yang memiliki asal node yang bernilai 0 tadi
5. Mengurangi jumlah inbound dari setiap node yang tersimpan tadi
6. Diulangi kembali ke proses 2 tetapi untuk baris selanjutnya dari matriks hasil
7. Setelah semua node selesai dikunjungi, pengulangan berhenti dan matriks hasil dengan baris berupa urutan “semester” dan kolom merupakan list “mata kuliah” yang bisa diambil pada “semester” tersebut.

Sedangkan algortma program tersebut adalah:

1. Menanyakan nama file pada user (dengan asumsi nama file benar dan file terletak pada folder test)
2. Program memparsing file dan memasukkannya pada vector of string
3. hasil parse dibersihkan dari delimiter seperti koma dan titik, untuk titik program akan memasukkan “\n” sebagai pembatas antara requirement dan required sedangkan koma untuk menunjukkan bahwa string tersebut masuk ke elemen berikutnya
4. Hasil parse dibuat mapnya dengan key berupa integer dari 0 hingga jumlah “mata kuliah” – 1 dan value berupa nama mata kuliah
5. Program meng-encode hasil parse menjadi integer untuk memudahkan proses, kemudian mengubah hasil parse menjadi bentuk directed graph (asumsi masukan akan selalu membentuk acyclic graph) dengan bentukan linked list
6. Program menjalankan proses pencarian solusi (terdapat di atas)

7. Program menanyakan user mengenai keterangan yang diinginkan untuk diprint (misalnya “semester” atau “modul”, jika dikosongkan defaultnya adalah “Semester”
8. Program memprint hasil sesuai dengan keterangan
9. Program selesai

B. Source Code Program (C++)

main.cpp

```
/*
Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
*/

#include "parser13519163.hpp"
#include "graph13519163.hpp"
#include <iostream>
using namespace std;


void print_map(map<int, string> dict)
{
    for (auto const& pair: dict) {
        std::cout << "{" << pair.first << ": " << pair.second << "}\n";
    }
}

int main() {
    string path = inputFilePath();
    vector<string> hasilparse = parseFile(path);
    map<int, string> dict = dictionary(hasilparse);
    DAGraph G(dict.size());
    //printHasilParse(hasilparse);
    G.parseToGraph(dict,hasilparse);
    // print_map(dict);
    G.topoSort();
    G.sortSemester();
    string ket = getKet();
    G.printHasilSemester(dict, ket);

    return 0;
}
```

graph.hpp

```
/*
Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
*/

#ifndef _GRAPH_HPP_
#define _GRAPH_HPP_

#include <iostream>
#include <string>
#include <vector>
#include "parser13519163.hpp"
#include <list>
using namespace std;


class DAGraph {
    private:
    // VARIABEL KEBUTUHAN GRAPH
        // jumlah sudut dari graph
        int jmlsudut;
        // "list" dari sudut-
sudut pada graph (graph dengan implementasi linked list)
        list<int>* node;

    // VARIABEL KEBUTUHAN TOPOSORT
        // keep track yang sudah dikunjungi dan belum dikunjungi
        bool* vis;
        // Menyimpan list sudut yang akan dihapus inboundnya
        vector<int> temp;
        // Menyimpan jumlah inbound
        int* inbound;
        // Menyimpan node per semesternya
        vector<vector<int>> hasil;

    // FUNGSI REKURSIF iterate
        /**
         * Fungsi untuk melakukan Depth First Search, untuk membantu toposort
(rekursif)
         * @param {int} a sebagai node yang di-"visit"
         **/
        void iterate(int a);
```

```
    // FUNGSI PEMBANTU
        /**
         * Mengecek sudut mana yang memiliki inbound bersangkutan dengan a dan
 memasukkan ke vector temp
         * @param {int} a sebagai node incoming
         **/
        void checkInboundFrom(int a);
    public:
        /**
         * Konstruktor Directed Acyclic Graph
         * @param {int} jumlah sudut
         **/
        DAGraph(int t_jmlsudut);

        /**
         * Menambahkan sudut pada DAG
         * @param {int} id sudut asal
         * @param {int} id sudut tujuan
         **/
        void addSisi(int asal, int tujuan);

        /**
         * Fungsi untuk memasukkan sisi ke dalam graph sesuai dengan hasil par
ser/ meng-encode hasil parse,
         * kemudian memasukannya ke dalam graph
         * @param {map<int, string>} map untuk meng-encode hasil parse
         * @param {vector<string>} hasil parse yang akan di-encode
         * @param {DAGraph} Graph yang akan ditambahkan sisinya
         **/
        void parseToGraph(map<int, string> ndict, vector<string> hasilparse);

        /**
         * Melakukan toposort dan membagi hasil sorting sesuai dengan prerequs
itesnya
         **/
        void topoSort();

        /**
         * Menyusun hasil dari penghitungan inbound dengan menguranginya
         **/
        void sortSemester();


        // Print
        /**
         * Print hasil dari semester
         **/
        void printHasilSemester(map<int, string> dict, string ket);
```

```
        /**
         * Print hasil dari toposort
         **/
        void printHasil();


};
```

graph.cpp

```
/*
Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
*/

#include "graph13519163.hpp"
using namespace std;


DAGraph::DAGraph(int t_jmlsudut) {
    this->jmlsudut = t_jmlsudut;
    node = new list<int>[t_jmlsudut];
    vis = new bool[this->jmlsudut];
    inbound = new int[this->jmlsudut];
    for (int i = 0; i < this->jmlsudut; i++) {
        vis[i] = false;
        inbound[i] = 0;
        hasil.push_back(vector<int>());
    }

}

void DAGraph::addSisi(int asal, int tujuan) {
    node[asal].push_back(tujuan);
}

void DAGraph::parseToGraph(map<int, string> ndict, vector<string> hasilparse)
{
    int code = 0;
    int tempasal;
    int temptuj;
    for (int i=1; i<hasilparse.size()-1; i++) {
        if (hasilparse[i]=="\n") {
            code = i+1;
            i++;
        } else {
```

```
            tempasal = dictKey(ndict,hasilparse[i]);
            temptuj = dictKey(ndict,hasilparse[code]);
            addSisi(tempasal,temptuj);
        }
    }
}
```

toposort.cpp

```
/*
Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
*/

#include "graph13519163.hpp"
using namespace std;

void DAGraph::topoSort() {
    for (int i = 0; i < this->jmlsudut; i++) {
        if (vis[i] == false) {
            iterate(i);
        }
    }
    cout << endl;
    // printHasil();
}

// void DAGraph::printHasil() {
//     cout << "jumlah node" << endl;
//     for (int i=0; i<this->jmlsudut; i++) {
//         cout << i << " ";
//         cout << inbound[i] << endl;
//     }
// }


void DAGraph::iterate(int a) {
    vis[a] = true;
    list<int>::iterator i;
    for (i = node[a].begin(); i != node[a].end(); ++i) {
        ++inbound[*i];
        if (!vis[*i]) {
            iterate(*i);
        }
    }
}
```

```
void DAGraph::sortSemester() {
    int sem = 0;
    for (sem; sem<this->jmlsudut; sem++) {          // majuin semester
        for (int i=0; i<this-
>jmlsudut; i++) {      // iterasi untuk nyari yang inbound = 0
            if (inbound[i]==0) {
                hasil[sem].push_back(i);            // sudut yang inbound = 0
ke semester terkait
                checkInboundFrom(i);
                inbound[i]=-1;
            }
        }
        for (int i=0; i<temp.size(); i++) {
            --inbound[temp[i]];
        }
        temp.clear();
    }
}

void DAGraph::checkInboundFrom(int a) {
    int x;
    list<int>::iterator it;
    for (it=node[a].begin(); it!=node[a].end(); ++it) {
        x = *it;
        temp.push_back(x);
    }
}

void DAGraph::printHasilSemester(map<int, string> dict, string ket) {
    for (int i=0; i<hasil.size(); i++) {
        if (!hasil[i].empty()) {
            cout << ket << " " << i+1 << ": ";
            for (int j=0; j<hasil[i].size(); j++) {
                cout << dict.find(hasil[i][j])->second << " ";
            }
            cout << endl;
        }
    }
}
```

parser.hpp

```
/*
Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
```

```
*/

#ifndef _PARSER_HPP_
#define _PARSER_HPP_

#include "graph13519163.hpp"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
using namespace std;

/**
 * Fungsi untuk parsing file dan memasukkan semua nilai didalam file ke dalam vector<string>
 * @param {string} filepath dari file test
 * @return {vector<string>} hasil dari parsing file
 **/
vector<string> parseFile(string &filepath);

/**
 * Fungsi untuk membuat mapping/dictionary dari hasil parsefile
 * @param {vector<string>} vector of string dari hasil parsing file
 * @return {map<int, string>} map dari hasil parse file dengan int sebagai key
 **/
map<int, string> dictionary(vector<string> hasilparse);

/**
 * Fungsi untuk mengembalikan key dari map berdasarkan isinya
 * @param {map<int, string>} mapping sudut dan key nya
 * @param {string} nilai yang mau dicari key nya dari map
 * @return {int} nilai dari key yang sudah dicari
 **/
int dictKey(map<int, string> ndict, string key);

/**
 * Fungsi untuk meminta filepath dan memparse file
 * @param {string} nama file beserta extensionnya (filetype)
 * @return {string} path lengkap dari file tersebut (di dalam test)
 **/
string inputFilePath();

/**
 * Fungsi untuk meminta keterangan
 * @return {string} keterangan
 **/
string getKet();
```

```
#endif
```

parser.cpp

```
      /*

Nama    : Alvin Wilta
NIM     : 13519163
Kelas   : K03
Tugas   : Tucil 2 STIMA
*/

#include "parser13519163.hpp"

using namespace std;
extern string ket;

vector<string> parseFile(string &filepath) {
    // Membaca file input
    vector<string> calongraph;
    ifstream in(filepath);
    // define comma dan period sebagai delimiter
    string comma = ", ";
    string period = ".";
    // Membuka file
    if (in.is_open()) {
        string temp;
        while (getline(in, temp)) {
            size_t position = 0;
            string token;
            while ((position = temp.find(comma)) != string::npos || (position
= temp.find(period)) != string::npos ) {
                // Token diambil dari file dan dimasukkan ke dalam calongraph
                token = temp.substr(0, position);
                calongraph.push_back(token);

                // Membuang delimiter comma dan period dengan menghapus sebany
ak delimiter
                if (temp.find(comma)) {
                    temp.erase(0, position + 2);
                } else if (temp.find(period)) {
                    temp.erase(0, position + 1);
                }
            }
            calongraph.push_back("\n");
        }
    } else {
```

```
        cout << "file gagal dibuka, mungkin penamaan file anda salah?" << endl
;
    }
    // Handling error jika gagal DAN belum mencapai EOF pada file
    if(in.fail() && !in.eof())
        cout << "File error saat dibuka" << endl;

    // File ditutup kembali
    in.close();

    return calongraph;
}



map<int, string> dictionary(vector<string> hasilparse) {
    // Referensi dari dictionary python
    map<int, string> dict;
    dict.insert(make_pair(0,hasilparse[0]));
    int idx = 1;
    for (int i=0; i<hasilparse.size()-1; i++) {
        if (hasilparse[i]=="\n") {
            dict.insert(make_pair(idx,hasilparse[i+1]));
            idx++;
        }
    }
    return dict;
}


int dictKey(map<int, string> ndict, string key) {
    map<int, string>::iterator itr;
    for (itr = ndict.begin(); itr != ndict.end(); ++itr) {
        if (itr->second==key) {
            return itr->first;
        }
    }
    return 0; // sebagai pengaman
}


string inputFilePath() {
    string input;
    cout << "\n";
    cout << "Selamat datang di Postulationem!\n";
    cout << "Program untuk mengetahui urutan kebutuhan anda!\n";
    cout << "\n";
    cout << "mohon masukkan nama file yang ingin anda parse beserta extensionn
ya (misal test1.txt)\n";
```

```
    cout << "*** tolong pastikan file anda sudah ada di folder test ***\n";
    cout << "input nama file anda: ";
    getline(cin,input);
    cout << endl;
    return "../test/" + input;
}

string getKet() {
    string ket;
    cout << "Proses selesai!\n\n";
    cout << "Masukkan format nama yang anda inginkan!\n";
    cout << "Misalnya 'Semester' atau 'Modul' tanpa tanda petik\n";
    cout << "jika anda kosongkan (press enter) maka defaultnya adalah Semester
" << endl;
    getline(cin,ket);
    if (ket.length()==0) {
        ket = "Semester";
    }
    return ket;
}
```

C. Screenshot Input & Output

Tampilan penuh program:

```
alvin@Alvin:/mnt/c/Users/Lenovo X1 Carbon/Alvin/Kuliah/Stima/Tucil2_13519163/src$ ./main


Selamat datang di Postulationem!
Program untuk mengetahui urutan kebutuhan anda!

mohon masukkan nama file yang ingin anda parse (beserta extensionnya (misal test1.txt)
*** tolong pastikan file anda sudah ada di folder test ***
input nama file anda: test1.txt




Masukkan format nama yang anda inginkan!
Misalnya 'Semester' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester

Semester 1: C3
Semester 2: C1
Semester 3: C4
Semester 4: C2
Semester 5: C5
```

D. Output program:

1) Masukan 1
   a. isi file

```
1   C1, C3.
2   C2, C1, C4.
3   C3.
4   C4, C1, C3.
5   C5, C2, C4.
```

   b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Contoh
Contoh 1: C3
Contoh 2: C1
Contoh 3: C4
Contoh 4: C2
Contoh 5: C5
```

2) Masukan 2
   a. isi file

```
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.
MA1201, MA1101.
FI1201, FI1101.
IF1210.
KU1202.
EL1200, MA1101.
IF2121.
IF2110.
IF2120.
IF2124.
IF2123, MA1101.
IF2130.
IF2210, IF2110.
IF2211.
IF2220, MA1101, MA1201, IF2120.
IF2230.
IF2240.
IF2250.
IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140.
IF3151, IF2250.
IF3210, IF2130, IF2110.
IF3270, IF3170, IF2110.
IF3230, IF3130.
IF3250, IF3150, IF2250.
IF3260, IF2130, IF2110, IF2123.
IF3280.
IF4090, IF3280.
IF4091.
KU2071.
IF4092, IF4091.
KU206X.
AS2005.

```

b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester

Semester 1: MA1101 FI1101 KU1001 KU1102 KU1011 KU1024 IF1210 KU1202 IF2121 IF2110 IF2120 IF2124 IF2130 IF2211 IF2230 IF2240 IF2250 IF3140 IF3280 IF4091 KU2071 KU206X AS2005
Semester 2: MA1201 FI1201 EL1200 IF2123 IF2210 IF3130 IF3141 IF3150 IF3151 IF3210 IF4090 IF4092
Semester 3: IF2220 IF3110 IF3230 IF3250 IF3260
Semester 4: IF3170
Semester 5: IF3270
```

3) Masukan 3

a. isi file

```
B, A.
C, B.
D1, C.
H1, A, G1.
D2, C.
H2, A, G2.
G1, E, F.
A.
G2, F, E.
E, A, D1.
F, E.
```

b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Modul
Modul 1: A
Modul 2: B
Modul 3: C
Modul 4: D1 D2
Modul 5: E
Modul 6: F
Modul 7: G1 G2
Modul 8: H1 H2
```

4) Masukan 4

a. isi file

```
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.

MA1201, MA1101.
FI1201, FI1101.
IF1210, KU1102.
KU1202, KU1102.
KI1002, KU1011.
EL1200, FI1101.

IF2121, IF1210, MA1101, MA1201.
IF2110, KU1102, IF1210.
IF2120, MA1201, MA1101.
IF2124, EL1200.
IF2123, MA1201.
IF2130, KU1202.

IF2210, IF2110.
IF2211, IF2110.
IF2220, MA1101, MA1201, IF2120.
IF2230, IF2130.
IF2240, IF2121, IF2120.
IF2250, KU1202, IF2110.

IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140, IF2240.
IF3151, IF2250.

IF3210, IF2110, IF2130, IF3110.
IF3270, IF2210, IF3170.
IF3230, IF3130.
IF3250, IF2250, IF3150.
IF3260, IF2123, IF2110, IF2130, IF3151.
IF3280, IF3151, IF3150.

IF4090, IF3280.
IF4091, IF3280.

IF4092, IF4091.
```

b. output

```
Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester

Semester 1: MA1101 FI1101 KU1001 KU1102 KU1011 KU1024 MA1201
 IF2121
 IF2210
 IF3170
 IF3210
 IF4090
 IF4092
Semester 2: FI1201 IF1210 KU1202 KI1002 EL1200 IF2120 IF2123 IF3270
Semester 3: IF2110 IF2124 IF2130 IF2220 IF2240
Semester 4: IF2211 IF2230 IF2250 IF3110 IF3140
Semester 5: IF3130 IF3141 IF3150 IF3151
Semester 6: IF3230 IF3250 IF3260 IF3280
Semester 7: IF4091
```

5) Masukan 5
   a. isi file

```
1   IP.
2   Saya.
3   Pasti, IP.
4   Bagus, Pasti.
5   Dan, IP, Pasti, Bagus.
6   Lulus, Dan.
7   Tepat waktu, Dan, Bagus.
```

   b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Harapan
Harapan 1: IP Saya
Harapan 2: Pasti
Harapan 3: Bagus
Harapan 4: Dan
Harapan 5: Lulus Tepat waktu
```

6) Masukan 6
   a. isi file

```
1   ke pasar.
2   beli.
3   kelapa.
4   parut.
5   eh, ke pasar.
6   bisa, beli.
7   jadi, kelapa, parut.
8   urut, parut.
```

   b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Kalimat pantun
Kalimat pantun 1: ke pasar beli kelapa parut
Kalimat pantun 2: eh bisa jadi urut
```

7) Masukan 7
   a. isi file

```
A, B.
C.
D, E.
F.
G.
H, F, G, I.
J.
K.
L, C.
M, N, G, F.
N, I, C.
O, J.
P, N, L.
Q, R, G.
E, Q.
I.
B, F, C.
R.
N.
```

   b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Modul
Modul 1: C F G J K I R N
Modul 2: H L N O Q B
Modul 3: A M P E
Modul 4: D
```

8) Masukan 8
   a. isi file

```
Lorem.
ipsum.
dolor.
sit.
amet.
consectetur.
adipiscing.
elit.
Fusce, ipsum.
eget, elit, amet, sit.
velit, dolor, ipsum.
dapibus, consectetur.
mattis, sit, ipsum.
nec, Lorem, amet.
fermentum enim, sit, amet, dolor.
Donec, nec.
luctus, elit, Fusce.
nunc, mattis.
Vivamus, Donec.
sollicitudin, velit, fermentum enim, luctus.
vulputate, luctus.
```

b. output

```
Proses selesai!

Masukkan format nama yang anda inginkan!
Misalnya 'Semester' atau 'Modul' tanpa tanda petik
jika anda kosongkan (press enter) maka defaultnya adalah Semester
Line
Line 1: Lorem ipsum dolor sit amet consectetur adipiscing elit
Line 2: Fusce eget velit dapibus mattis nec fermentum enim
Line 3: Donec luctus nunc
Line 4: Vivamus sollicitudin vulputate
```

E. Tabel Penilaian

Poin	Ya	Tidak
Program berhasil dikompilasi	√	
Program berhasil running	√	
Program dapat membaca file masukan dan menuliskan luaran	√	
Program dapat menerima berkas input dan menuliskan output.	√	
Luaran sudah benar untuk semua kasus input.	√	

F. Sumber Program

bit.ly/Tucil2Alvin13519163