LAPORAN TUGAS KECIL 2

Mata Kuliah Strategi Algoritma IF 2211
Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T., M.Sc.

Penyelesaian Cryptarithmetic dengan Brute Force

Disusun Oleh :

Jesson Gosal Yo – 13519079

13519079@std.stei.itb.ac.id

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma Decrease and Conquer

Langkah-Langkah untuk mencari solusi dalam permasahan pengurutan pengambilan mata kuliah dengan strategi decrease and conquer secara umum adalah dengan merepresentasikan mata kuliah dan *prerequisite* sebagai DAG(*Directed Acyclic Graph)* dan melakukan topological sort. Mata kuliah akan direpresentasikan sebagai node pada graph dan relasi *prerequisite* akan dilambangkan dengan *edge*. Misalkan ada 2 buah mata kuliah sebut A dan B, untuk kasus ini diumpamakan B merupakan *prerequisite dari* A. Maka akan ada hubungan *edge* berarah keluar dari B masuk ke A. Setelah DAG dikonstruksi maka hanya perlu dilakukan pengambilan *node-node* yang tidak memiliki *prerequisite*. Setelah diambil *node* akan dibuang bersamaan dengan relasi-relasi yang mengandung *node* tersebut. Hal ini dilakukan secara terus menerus hingga tidak ada *node* yang tersisa dalam himpunan graph kita.

Pendekatan penulis dalam mengimplementasi DAG adalah menggunakan matrix of boolean atau boolmatrix. Baris pada matrix merepresentasikan *node* dan kolom pada matrix merepresentasikan *edge*. Jika ada hubungan antara *node* i dan j maka pada boolmatrix akan direpresentasikan sebagai matrix[i][j] = true dan jika sebaliknya tidak ada hubungan akan direpresentasikan sebagai false. Pertama penulis melakukan mapping dari kumpulan nama mata kuliah menjadi index menggunakan list 1 dimensi untuk menyimpan identitas dari index, index ini akan dipakai untuk menyimpan mata kuliah pada matrix.

Penulis mengasumsikan input selalu memiliki solusi dan dapat direpresentasikan sebagai graf asiklis berarah.Penulis mengimplementasikan graf sebagai matrix yang dikonstruksi dengan double pointer karena secara hardware, mesin akan lebih cepat ketika mengakses elemen menggunakan pointer dibandingkan memanfaatkan STL yang diberikan c++ misalkan vektor. Pendekatan iteratif dapat memiliki kompleksitas asimtotik waktu dan ruang yang lebih baik daripada pendekatan rekursif. Namun agar dapat menggambarkan hubungan rekursivitas decrease and conquer pada topological sort maka penulis akan membuat fungsi sort dalam bentuk rekursif. Secara sederhana, cara kerja algoritma adalah sebagai berikut.

1. Membaca file untuk melakukan mapping nama mata kuliah menjadi index yang bertipe data integer.
2. Mereset pointer pembaca file ke awal file
3. Melakukan pembacaan ulang file untuk mengkonstruksi boolmatrix dengan memanfaatkan mapping yang dibuat pada poin 1.
4. Melakukan konstruksi 2 buah stack yaitu reversedplan dan plan
5. Memanggil method topological sorting dari object boolmatrix
6. Topological sorting akan menghapus *node* yang tidak memiliki pendahulu dengan cara menghapus *row* dan *column* dari *node* sekaligus mengurangi ukuran efektif dari matrix tersebut
7. *Node* yang dihapus pada poin 6 akan di-*push* ke dalam stack reversedplan
8. Poin 6 dan 7 akan dilakukan hingga matrix kosong

9. Isi dari stack reversedplan akan dipop dan dipush ke dalam plan, hal ini dilakukan agar pencetakan mata kuliah tidak terbalik

10. Isi plan akan dipop sambil dicetak ke layar.

Penulis menyediakan *Graphical User Interface* untuk program ini menggunakan *framework* Qt5. Alur cara kerja GUI yang dibuat adalah mengkonstruksi object MainWindow yang memiliki event listener click pada tombol choose file. Setelah file dipilih maka akan dipanggil algoritma yang ada di atas untuk melakukan pengurutan mata kuliah dan mencetaknya ke layar. Perlu diperhatikan bahwa executable hanya bisa dijalankan pada OS Windows dan jika ingin melakukan kompilasi ulang file untuk OS diperlukan Qt5 pada OS target.

B. Source Code Program

1. main.cpp

```
#include <bits/stdc++.h>

#include "boolmatrix.h"
#include "mainwindow.h"

#include <QApplication>
using namespace std;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

2. mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "boolmatrix.h"
#include <bits/stdc++.h>

using namespace std;

//Mengubah angka desimal menjadi huruf romawi
string romanNumeralsEncoder(int n){
    int val[13] = {1000,900,500,400,100,90,50,40,10,9,5,4,1};
    string sym[13] = {"M","CM","D","CD","C","XC","L","XL","X","IX","
V","IV","I"};
    string roman = "";
    for(int i=0;i<13;i++){
        while(n>=val[i]){
```

```
            roman.append(sym[i]);
            n -= val[i];
        }
    }
    return roman;
}

//Mendapatkan index dari sebuah vektor yang elemennya = isi dari cou
rse, fungsi mereturn -1 jika tidak ditemukan
int getIndex(vector<string> v, string course){
    for(int i=0;i<v.size();i++){
        if(course==v[i]){
            return i;
        }
    }
    return -1;
}

//Mendapatkan index node dengan edge masuk = 0
int findNodeZero(bool **DAG,int n){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(DAG[i][j]==1){
                break;
            } else if(j==n-1 && DAG[i][j] == 0){
                return i;
            }
        }
    }
    return -1;
}


MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

//Destruktor object MainWindow
MainWindow::~MainWindow()
{
    delete ui;
}


//Event Listener ketika tombol Choose File ditekan
void MainWindow::on_pushButton_clicked()
```

```
{
    QString filename = QFileDialog::getOpenFileName(this, "Open a fi
le", "C://"); //Melakukan pemanggilan syscall untuk FileDialog
    QScrollBar *sBar = new QScrollBar();
    QTextCursor cursor = ui->textEdit->textCursor();
    ui->textEdit->setVerticalScrollBar(sBar);
    //Memberi scrollbar untuk bagian text edit
    fstream courses_file;

    vector<string> courses;

    courses_file.open(filename.toStdString(), ios::in);
    if(!courses_file){
        cout << "File not found\n";
    } else{
        string line;
        string tempString;

        while(getline(courses_file,line)){
            for(char &c: line){
                if(c!=',' && c!='.' && !isspace(c)){
                    tempString.push_back(c);
                } else if(tempString!=""){
                    if(count(courses.begin(),courses.end(),tempStrin
g)==0){
                        courses.push_back(tempString);
                    }
                    tempString = "";
                }
            }
        }

        //Deklarasi 2D variable sized matrix
        BoolMatrix DAG(courses.size());

        courses_file.clear();
        courses_file.seekg (0, ios::beg);

        tempString = "";

        while(getline(courses_file,line)){
            vector<int> nodeAdj;
            for(char &c : line){
                if(c!=',' && c!='.' && !isspace(c)){
                    tempString.push_back(c);
                } else if(tempString!=""){
                    nodeAdj.push_back(getIndex(courses,tempString));
                    tempString = "";
```

```
                }
            }

            for(int i=1;i<nodeAdj.size();i++){
                DAG.matrix[nodeAdj[0]][nodeAdj[i]] = true;
            }
        }

        stack<vector<string>> reversedplan;
        stack<vector<string>> plan;

        DAG.topologicalsort(reversedplan,courses);

        //Melakukan pencetakan isi stack
        while(reversedplan.size()){
            plan.push(reversedplan.top());
            reversedplan.pop();
        }
        int x=1;
        while(plan.size()){
            string semester= "Semester ";
            semester.append(romanNumeralsEncoder(x));
            ui->textEdit->append(QString::fromStdString(semester));
            for(int i=0;i<plan.top().size();i++){
                ui->textEdit-
>append(QString::fromStdString(plan.top()[i]));
            }
            ui->textEdit-
>append("==================================================");
            plan.pop();
            x++;
        }
    }
    courses_file.close();
    ui->textEdit->selectAll();
    ui->textEdit->setFontPointSize(20);
    ui->textEdit->setTextCursor( cursor );
}
```

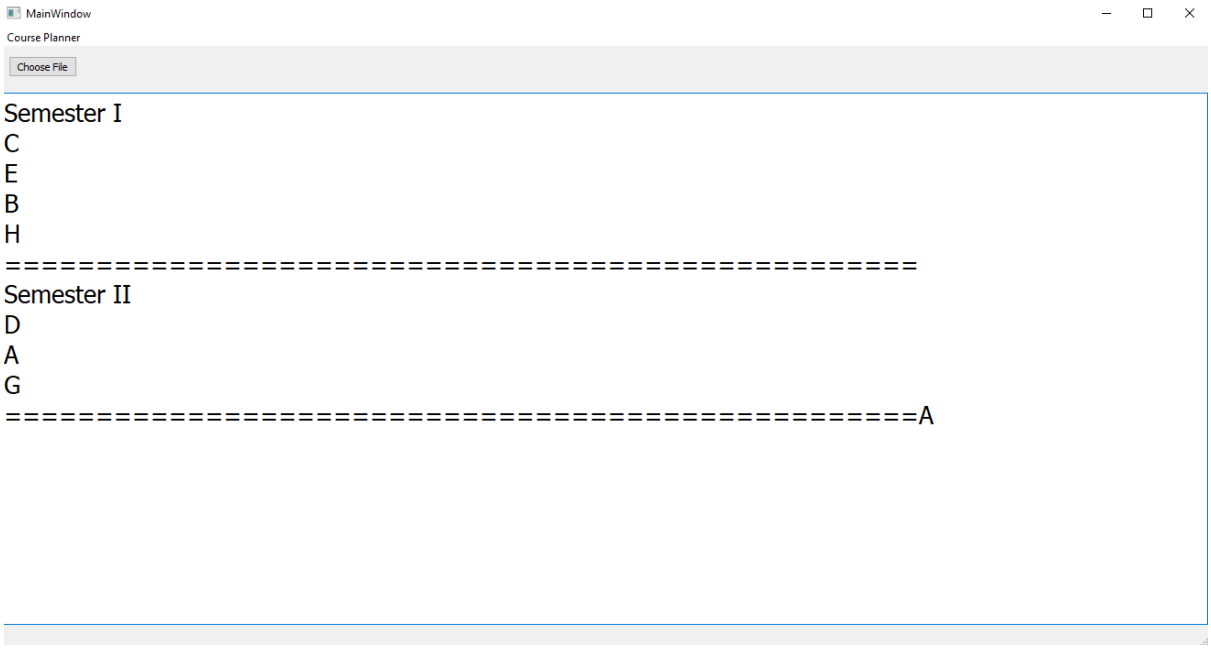
3. boolmatrix.cpp

```
#include "boolmatrix.h"

//Konstruktor BoolMatrix
BoolMatrix::BoolMatrix(int size){
    this->size=size;
    bool **DAG = new bool*[size];
    for(size_t i = 0; i < size; ++i){
        DAG[i] = new bool[size];
    }
```

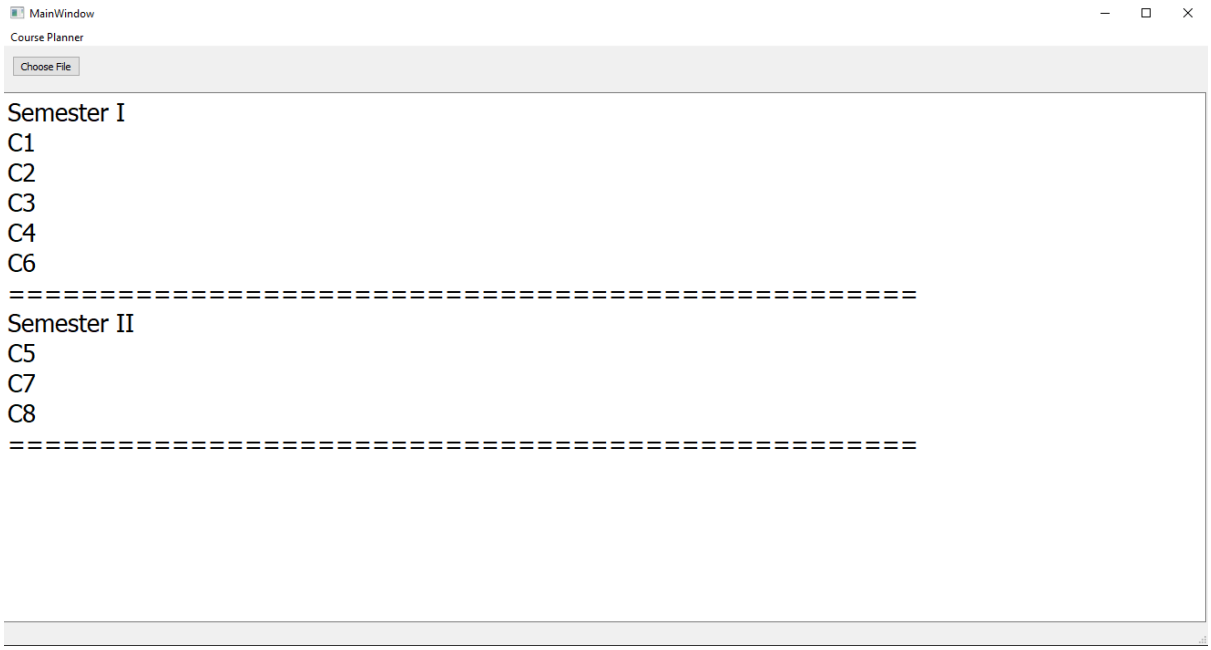
```
    for(int i=0;i<size;i++){
        for(int j=0;j<size;j++){
            DAG[i][j] = false;
        }
    }
    this->matrix = DAG;
    vector<string> newVector;
}

int BoolMatrix::getSize(){
    return this->size;
}

//Menghapus node pada graf sekaligus menghapus edge yang keluar dari nod
e tersebut
void BoolMatrix::removeNode(int n){
    bool **decreasedDAG = new bool*[this->size-1];
    for(int i = 0; i < this->size-1; ++i){
        decreasedDAG[i] = new bool[this->size-1];
    }
    for(int i=0;i<size-1;i++){
        for(int j=0;j<size-1;j++){
            decreasedDAG[i][j] = false;
        }
    }
    int k=0; int l=0;
    for(int i = 0; i < size;i++){
        l=0;
        for(int j = 0; j < size;j++){
            if(i!=n && j!=n){
                decreasedDAG[k][l] = this->matrix[i][j];
                l++;
            }
        }
        if(i!=n){
            k++;
        }
    }
    this->matrix = decreasedDAG;
    this->size -=1;
}

//Mengambil index node-node yang memiliki derajat simpul masuk = 0
vector<int> BoolMatrix::getNodes(){
    bool found;
    vector<int> nodes;
    for(int i=0;i<this->size;i++){
        found = true;
```

```
        for(int j=0;j<this->size;j++){
            if(this->matrix[i][j] != 0){
                found = false;
            }
        }
        if(found){
            nodes.push_back(i);
        }
    }
    return nodes;
}

//Method untuk melakukan sorting dan memasukkan hasil ke dalam Stack S s
ecara rekursif
void BoolMatrix::topologicalsort(stack<vector<string>> &S,vector<string>
 &C){
    vector<int> nodes = this->getNodes();
    vector<string> courseplan;
    if(this->size==0){
        return;
    } else{
        for(int i=0;i<nodes.size();i++){
            cout << nodes[i] << " ";
            courseplan.push_back(C[nodes[i]]);
        }
        S.push(courseplan);
        while(nodes.size()>0){
            int temp = nodes[0];
            this->removeNode(nodes[0]);
            for(int i=0;i<nodes.size();i++){
                if(nodes[i]>temp){
                    nodes[i] = nodes[i] - 1;
                }
            }
            C.erase(C.begin()+nodes[0]);
            nodes.erase(nodes.begin());
        }
        this->topologicalsort(S,C);
    }
}
```

C. Screenshot Output

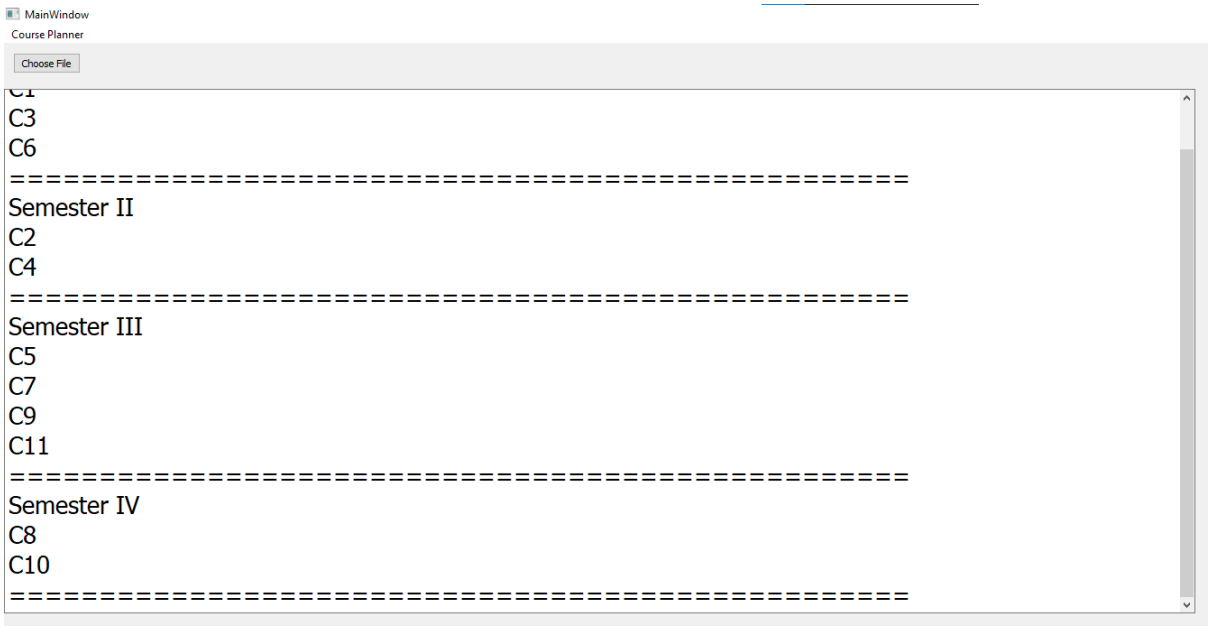


Test Case 1



Test Case 2


MainWindow
Course Planner
Choose File
Semester I
C1
C3
C6
==========
Semester II
C2
C4
==========
Semester III
C5
C7
C9
C11
==========
Semester IV
C8


MainWindow
Course Planner
Choose File
C3
C6
==========
Semester II
C2
C4
==========
Semester III
C5
C7
C9
C11
==========
Semester IV
C8
C10
==========

Test Case 3

```
==================================================
Semester I
MatematikaIA
Olahraga
==================================================
Semester II
MatematikaIIA
==================================================
Semester III
MatematikaDiskrit
==================================================
Semester IV
Probstat
Kriptografi
==================================================
```

Test Case 4

```
--------------------------------------------------
Semester I
C1
C5
C8
==================================================
Semester II
C2
C3
C6
==================================================
Semester III
C4
C7
==================================================
Semester IV
C9
==================================================
```

Test Case 5

```
Semester I
C1
C2
C3
C6
C9
==========
Semester II
C4
C7
C10
==========
Semester III
C5
C8
==========
```

Test Case 6

```
==========
Semester I
C3
==========
Semester II
C1
==========
Semester III
C4
==========
Semester IV
C2
==========
Semester V
C5
==========
```

Test Case 7

```
Semester I
C1
C2
C4
C6
==================================================
Semester II
C3
C5
==================================================
Semester III
C7
C8
C9
==================================================
Semester IV
C10
```

D. Link Github

https://github.com/Aphostrophy/CoursePlanner.git

E. Tabel Ceklist:

Poin	**Ya**	**Tidak**
1. Program berhasil dikompilasi	**V**	
2. Program berhasil running	**V**	
3. Program dapat menerima berkas input dan menuliskan output	**V**	
4. Luaran sudah benar untuk semua kasus input	**V**	