LAPORAN HASIL TUGAS KECIL I IF 2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2020/2021

PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT* (PENERAPAN *DECREASE* AND *CONQUER*)

Disusun oleh :

Christopher Chandrasaputra 13519074

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Membuat aplikasi sederhana yang dapat menysun rencana pengambilan kuliah dengan memanfaatkan algoritma *Decrease and Conquer*. Penyusunan rencana kuliah diimplementasikan menggunakan pendekatan *Topological Sorting* dengan spesifikasi sebagai berikut :

1. Aplikasi dibuat dengan bahasa C/C++/Java/Python
2. Menggunakan algoritma *Decrease and Conquer* dengan pendekatan *Topological Sorting* dalam menyusun rencana pengambilan kuliah.
3. *Input*: file teks yang berisi kumpulan mata kuliah (*Course*) beserta prasyarat mata kuliah yang perlu diambil sebelum mengambil matakuliah tersebut.
   Format isi file teks:

<kode_kuliah_1>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah prasyarat - 3>.
<kode_kuliah_2>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>.
<kode_kuliah_3>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah prasyarat - 3>, <kode kuliah prasyarat - 4>.
<kode_kuliah_4>.
.
.
.

Gambar 1. Format isi file teks, sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf, diakses pada tanggal 28 Februari 2021, pukul 23.07 WIB

   Contoh isi file teks:

C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.

Gambar 2. Contoh isi file teks, sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf, diakses pada tanggal 28 Februari 2021, pukul 23.07 WIB

4. *Output*: Tampilan di layar yang menampilkan hasil rencana yang telah dibuat sebagai output pada layar seperti pada contoh *output* berikut.

Semester I : C3
Semester II : C1
Semester III : C4
Semester IV : C2
Semester V : C5

Gambar 3. Contoh hasil output, sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf, diakses pada tanggal 28 Februari 2021, pukul 23.07 WIB

BAB II
PENJELASAN ALGORITMA *DECREASE AND CONQUER* UNTUK PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT*

Penyelesaian persoalan penyusunan rencana kuliah dengan *topological sort* dalam laporan ini menggunakan bahasa *Java*. Bahasa pemrograman ini dipilih sebagai bahasa yang digunakan karena ketertarikan penulis yang ingin mencoba bahasa pemrograman yang penulis jarang gunakan. Selain ketertarikan penulis, bahasa pemrograman ini dipilih karena penggunaan paradigma pemrograman *Object-Oriented Programming* serta fitur yang dimiliki *Java* sangat banyak.

2.1. Algoritma *Decrease and Conquer*

Algoritma *decrease and conquer* adalah algoritma yang mereduksi persoalan menjadi upa-persoalan yang lebih kecil tapi selanjutnya hanya memproses satu upa-persoalan saja. *Decrease* memiliki arti mengurangi persoalan menjadi persoalan yang lebih kecil. *Conquer* memiliki arti memproses upa-persoalan secara rekursif.

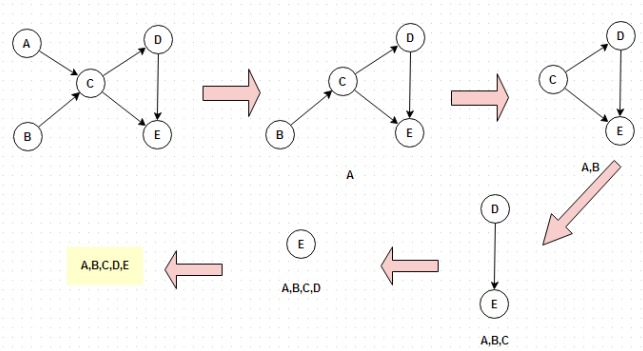
Terdapat tiga varian *decrease and conquer* yaitu:

- *Decrease by a constant*, setiap iterasi algoritma, persoalan akan dikurangi sebanyak konstanta yang ditentukan.
- *Decrease by a constant factor*, setiap iterasi algoritma, persoalan akan dikurangi sebanyak faktor dari konstanta yang ditentukan.
- *Decrease by a variable size*, setiap iterasi algoritma, jumlah persoalan yang akan dikurangi bervariasi.

2.2. Topological Sort

Penyusunan rencana kuliah dengan algoritma *Decrease and Conquer* dapat dilakukan menggunakan pendekatan *topological sort*. *Topological sort* merupakan cara untuk mengurutkan simpul pada graf berarah. Cara ini memerlukan sebuah graf berarah yang tidak memiliki siklus di dalamnya. Berikut adalah cara kerja dari *topological sort*:

1. Ambil sembarang simpul yang memiliki derajat masuk 0,
2. Hilangkan setiap busur yang keluar dari simpul yang telah diambil (derajat dari setiap simpul yang kehilangan busur masuk dari langkah ini akan dikurangi sebanyak busur yang terhubung pada simpul yang telah diambil),
3. Ulangi langkah 1 dan langkah 2 hingga setiap simpul pada graf berarah telah diambil semua.



Gambar 4. Ilustrasi langkah topological sort, sumber : http://geekrai.blogspot.com/2014/08/topological-sort.html, diakses pada tanggal 28 Februari 2021, pukul 23.09 WIB

2.3. Penyelesaian

Penyelesaian persoalan yang diberikan memiliki beberapa tahapan. Tahapan pertama adalah mengubah file *input* menjadi sebuah *ArrayList* (kelas yang disediakan oleh *Java*) yang terdiri dari objek-objek mata kuliah dengan menggunakan *method* yang dimiliki kelas *Tools_13519074* yang telah dibuat oleh penulis. Pada penyelesaian ini, objek dari mata kuliah memiliki kelas yaitu *Course_13519074* yang telah penulis buat.

Setelah file *input* telah diubah menjadi *ArrayList* (untuk memudahkan penjelasan akan dinamakan *ArrayList* hasil langkah pertama akan kita sebut sebagai Data Mata Kuliah), akan dilakukan dua pengecekan. Pengecekan pertama adalah pengecekan adanya siklus pada Data Mata Kuliah atau tidak. Pengecekan selanjutnya adalah pengecekan untuk mengetahui jumlah mata kuliah yang terdapat pada Data Mata Kuliah. Apabila Data Mata Kuliah dapat melewati pengecekan, maka akan dilanjutkan ke tahap selanjutnya.

Tahap selanjutnya adalah pembuatan rencana kuliah menggunakan pendekatan *topological sort*. Dengan menggunakan cara rekursif, untuk setiap iterasi yang dilakukan, akan dilakukan penghapusan mata kuliah yang diambil pada Data Mata Kuliah dan memperbarui setiap mata kuliah pada Data Mata Kuliah sehingga setiap mata kuliah yang memiliki prasyarat dari mata kuliah yang dihapus akan memiliki daftar prasyarat yang baru (sudah berkurang). Iterasi ini dilakukan hingga Data Mata Kuliah sudah tidak memiliki objek mata kuliah lagi.

Setelah pembuatan rencana kuliah selesai, akan ditampilkan ke layar hasil dari rencana yang telah dibuat. Pada penyelesaian ini, pembuatan rencana kuliah dapat dilakukan berkali-kali jika pengguna ingin mendapatkan hasil dari file *input* yang berbeda.

BAB III
KODE PROGRAM

3.1. Kelas *Course_13519074*

```java
package classes_13519074;
import java.util.ArrayList;

public class Course_13519074 {
    private ArrayList<String> prereq;
    private String courseID;

    /*** constructor ***/
    // Course_13519074 : Create objeck with the given input string as the objeck CourseID
    // input :
    //      courseID : String
    public Course_13519074(String courseID){ ···
    }

    /*** getter ***/
    // getCourseID : get object's course ID
    // output : String
    public String getCourseID(){ ···
    }

    // getTotalPrereq : get object's prereq size
    // output : int
    public int getTotalPrereq(){ ···
    }

    // getPrereqList : get object's prereq List
    // output : ArrayList<String>
    public ArrayList<String> getPrereqList(){ ···
    }

    /*** editing attributes ***/
    // addPrereq : add string (course id) to object's prereq List
    // input :
    //      courseID : String
    public void addPrereq(String courseID){ ···
    }

    // removePrereq : remove string (course id) from object's prereq List
    // input :
    //      courseID : String
    public void removePrereq(String courseID){ ···
    }

    // printInfo : print course info along with it's requirement
    public void printInfo(){ ···
    }

}
```

Gambar 5. Kelas Course_13519074

```java
/*** constructor ***/
// Course_13519074 : Create objeck with the given input string as the objeck CourseID
// input :
//      courseID : String
public Course_13519074(String courseID){
    this.prereq = new ArrayList<String>();
    this.courseID=courseID;
}
```

Gambar 6. Method Kelas Course_13519074 untuk membuat objek dengan kelas Course_13519074

```java
/*** getter ***/
// getCourseID : get object's course ID
// output : String
public String getCourseID(){
    return courseID;
}

// getTotalPrereq : get object's prereq size
// output : int
public int getTotalPrereq(){
    return prereq.size();
}

// getPrereqList : get object's prereq List
// output : ArrayList<String>
public ArrayList<String> getPrereqList(){
    return prereq;
}
```

Gambar 7. Method Kelas Course_13519074 untuk mengambil nilai atribut

```java
/*** editing attributes ***/
// addPrereq : add string (course id) to object's prereq List
// input :
//      courseID : String
public void addPrereq(String courseID){
    prereq.add(courseID);
}

// removePrereq : remove string (course id) from object's prereq List
// input :
//      courseID : String
public void removePrereq(String courseID){
    if (prereq.contains(courseID)) prereq.remove(courseID);
}

// printInfo : print course info along with it's requirement
public void printInfo(){
    System.out.println("COURSE ID : " + courseID);
    for (String cID : prereq){
        System.out.println('-' + cID);
    }
}
```

Gambar 8. Method Kelas Course_13519074 untuk mengubah atribut

3.2. Kelas *Romans_13519074*

```
package classes_13519074;

public class Romans_13519074 {
    // intToRomans : return string of roman number with value of given integer
    // NOTE : the given integer must have value > 0 since roman number doesn't have 0 or negative
    // input :
    //      value : int
    // output : String
    public static String intToRomans(int value){···
    }
}
```

Gambar 9. Kelas Romans_13519074

```
// intToRomans : return string of roman number with value of given integer
// NOTE : the given integer must have value > 0 since roman number doesn't have 0 or negative
// input :
//      value : int
// output : String
public static String intToRomans(int value){
    StringBuilder roman = new StringBuilder();
    while (value>0){
        if (value>=1000){
            roman.append("M");
            value -= 1000;
        } else if (value>=900){
            roman.append("CM");
            value -= 900;
        } else if (value>=500){
            roman.append("D");
            value -= 500;
        } else if (value>=400){
            roman.append("CD");
            value -= 400;
        } else if (value>=100){
            roman.append("C");
            value -= 100;
        } else if (value>=90){
            roman.append("XC");
            value -= 90;
        } else if (value>=50){
            roman.append("L");
            value -= 50;
        } else if (value>=40){
            roman.append("XL");
            value -= 40;
        } else if (value>=10){
            roman.append("X");
            value -= 10;
        } else if (value>=9){
            roman.append("IX");
            value -= 9;
        } else if (value>=5){
            roman.append("V");
            value -= 5;
        } else if (value>=4){
            roman.append("IV");
            value -= 4;
        } else {
            roman.append("I");
            value -= 1;
        }
    }
    return roman.toString();
}
```

Gambar 10. Method intToRomans Kelas Romans_13519074

3.3. Kelas *Tools_13519074*

```java
package classes_13519074;
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors

public class Tools_13519074 {
    // convertFileToArray : function to turn .txt file to Array of Course_13519074
    // input :
    //      filedir : String (.txt file name and directory)
    // output : ArrayList of Course_13519074
    public static ArrayList<Course_13519074> convertFileToArray(String filedir) throws FileNotFoundException {···
    }

    // hasCycle : function to check whether any of the Course_13519074 in ArrayList<Course_13519074> has a cycle
    // input :
    //      CourseData : ArrayList<Course_13519074>
    // output : boolean
    public static boolean hasCycle(ArrayList<Course_13519074> CourseData){···
    }

    // hasCycle : function to check if the given Course_13519074 in ArrayList<Course_13519074> has a cycle
    // input :
    //      CourseData : ArrayList<Course_13519074>
    // output : boolean
    public static boolean checkCycle(ArrayList<Course_13519074> CourseData, Course_13519074 currentCourse, ArrayList<String> track){
    }

    // makePlan : procedure to edit StringBuilder output for later be shown, this is also the main
    //            implementation of Topological Sort
    // input :
    //      output     : StringBuilder
    //      Semester   : int
    //      CourseData : ArrayList of Course_13519074
    public static void makePlan(StringBuilder output, int Semester, ArrayList<Course_13519074> CourseData) {···
    }
}
```

Gambar 11. Kelas Tools_13519074

```java
// convertFileToArray : function to turn .txt file to Array of Course_13519074
// input :
//      filedir : String (.txt file name and directory)
// output : ArrayList of Course_13519074
public static ArrayList<Course_13519074> convertFileToArray(String filedir) throws FileNotFoundException {
    ArrayList<Course_13519074> CourseData = new ArrayList<Course_13519074>();
    Scanner scanner = new Scanner(new File(filedir));
    while (scanner.hasNextLine()){
        String[] tempStringList = scanner.nextLine().replaceAll(" ", "").split("[,.]");
        if (tempStringList.length==0 || tempStringList[0].length()==0) continue;
        Course_13519074 tempCourse = new Course_13519074(tempStringList[0]);
        for (String courseID : tempStringList){
            if (courseID!=tempStringList[0]) tempCourse.addPrereq(courseID);
        }
        CourseData.add(tempCourse);
    }
    return CourseData;
}
```

Gambar 12. Method convertFileToArray Kelas Tools_13519074

```java
// hasCycle : function to check whether any of the Course_13519074 in ArrayList<Course_13519074> has a cycle
// input :
//      CourseData : ArrayList<Course_13519074>
// output : boolean
public static boolean hasCycle(ArrayList<Course_13519074> CourseData){
    for (Course_13519074 course : CourseData){
        if (checkCycle(CourseData, course, new ArrayList<String>())) return true;
    }
    return false;
}
```

Gambar 13. Method hasCycle Kelas Tools_13519074

```java
// hasCycle : function to check if the given Course_13519074 in ArrayList<Course_13519074> has a cycle
// input :
//      CourseData : ArrayList<Course_13519074>
// output : boolean
public static boolean checkCycle(ArrayList<Course_13519074> CourseData, Course_13519074 currentCourse, ArrayList<String> track){
    ArrayList<String> prereqList = currentCourse.getPrereqList();

    if (prereqList.size()==0) return false;                                     // if dont have prereq, automatically not a cycle

    if (track.contains(currentCourse.getCourseID())) return true;               // already passed (recorded in the track)
    else track.add(currentCourse.getCourseID());                                // add course to track

    boolean cyclecheck = false;
    for (Course_13519074 course : CourseData){
        if (course==currentCourse) continue;                                    // Skip course if it is the same
        if (prereqList.contains(course.getCourseID())){                         // Check if the iterated course is in prereqlist
            ArrayList<String> branchtrack = new ArrayList<String>(track);       // Create a branch track to prevent original track carried and chang
            cyclecheck = cyclecheck || checkCycle(CourseData, course, branchtrack);  // Check cycle for every branch
        }
    }

    return cyclecheck;
}
```

Gambar 14. Method checkCycle Kelas Tools_13519074

```java
// makePlan : procedure to edit StringBuilder output for later be shown, this is also the main
//            implementation of Topological Sort
// input :
//      output     : StringBuilder
//      Semester   : int
//      CourseData : ArrayList of Course_13519074
public static void makePlan(StringBuilder output, int Semester, ArrayList<Course_13519074> CourseData) {
    if (CourseData.size()==0) return;                                           // Base reccursion, stop if CourseData have no more course

    // Initialization
    ArrayList<Course_13519074> courseTaken = new ArrayList<Course_13519074>();  // to store Course_13519074 taken
    boolean firstRegister = true;                                               // for indicating the first course taken in the semester
    output.append(String.format("Semester %s\t:",Romans_13519074.intToRomans(Semester)));   // add current Semester to StringBuilder

    for (Course_13519074 course : CourseData){                                  // record taken course and adding Course ID to output
        if (course.getTotalPrereq()==0){
            if (firstRegister) {
                output.append(" ");
                firstRegister=false;
            } else output.append(", ");
            output.append(course.getCourseID());
            courseTaken.add(course);
        }
    }
    output.append("\n");                                                        // newline after taking all semesters

    for (Course_13519074 course : courseTaken){                                 // removing taken course from CourseData
        CourseData.remove(course);
    }

    for (Course_13519074 course : CourseData){                                  // removing taken course from every courses prereq list in
        for (Course_13519074 takenCourse : courseTaken){                        // CourseData
            if (course.getPrereqList().contains(takenCourse.getCourseID())){
                course.removePrereq(takenCourse.getCourseID());
            }
        }
    }

    makePlan(output,Semester+1,CourseData);                                     // Reccursive with passed output, reduced CourseData, and new Semester
}
```

Gambar 15. Method makePlan Kelas Tools_13519074

3.4. Kelas *Main_13519074* (Algoritma Utama)

```java
import classes_13519074.*;                                                  // Import user-made classes from package classes_13519074
import java.util.ArrayList;                                                 // Import to use ArrayList
import javax.swing.JOptionPane;                                             // Import to make interfaces
import java.io.File;                                                        // Import the File class
import java.io.FileNotFoundException;                                       // Import this class to handle errors


public class Main_13519074 {
    public static void main(String[] args) throws FileNotFoundException {
        // Initialization
        String path;

        while (true){                                                       // Looping while program is running
            do {
                path = JOptionPane.showInputDialog("Course file directory : ");     // Interface for input
                if (path==null) break;                                      // Break from loop if path is null
                if (new File(path).exists()) break;                         // Break if file is found
                else JOptionPane.showMessageDialog(null,"File not found!");         // Warning interface if file not found
            } while (true);

            if (path==null) break;                                          // Stop program if pat is null

            // Local Initialization
            ArrayList<Course_13519074> CourseData = Tools_13519074.convertFileToArray(path);    // Use convertFileToArray from Tools_13519074
                                                                                                // to convert file to ArrayList of Course_13519074
            StringBuilder output = new StringBuilder();
            int startsemester = 1;

            if (CourseData.size()==0){                                      // Check whether CourseData has courses or not
                output.append("NO COURSES FOUND!\nPlease check your course file again.\n");
            } else if (Tools_13519074.hasCycle(CourseData)){                // Check whether CourseData has cycle or not
                output.append("CYCLE EXIST!\nUnable to create plan.\n");
            } else {                                                        // Passed all check and proceed to edit output
                Tools_13519074.makePlan(output,startsemester,CourseData);   // by using Tools_13519074 makePlan
            }

            JOptionPane.showMessageDialog(null,                             // Show created plan result
                                          output.toString(),
                                          "Course Plan Result",
                                          JOptionPane.INFORMATION_MESSAGE);

            int again = JOptionPane.showConfirmDialog(null,                 // Option to choose wheter user want to generate
                                                      "Do you want to generate another plan?",  // new course plan or not
                                                      "Try Again Confirmation",
                                                      JOptionPane.YES_NO_OPTION);

            if (again==JOptionPane.NO_OPTION) break;                        // if NO option is selected, stop program from running
        }
    }
}
```

Gambar 16. Kelas Main_13519074 (Algoritma Utama)

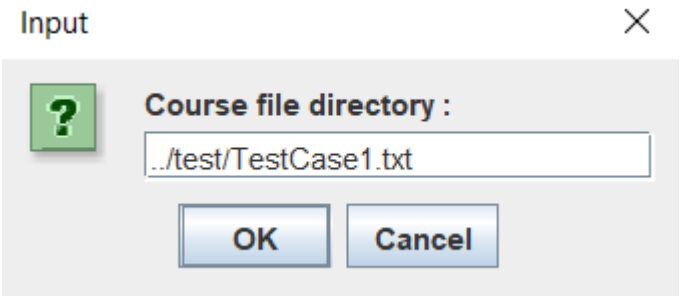
BAB IV
HASIL *SCREENSHOOT INPUT* DAN *OUTPUT*

4.1. TestCase1.txt

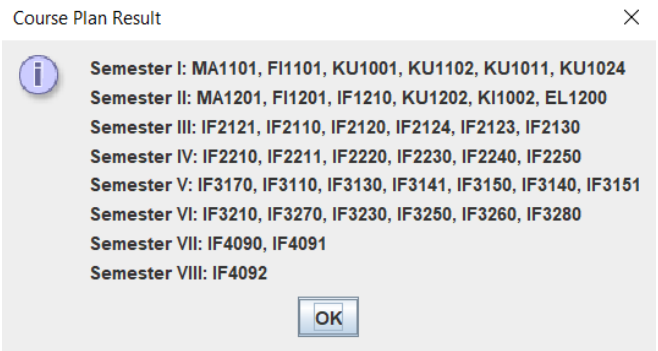
TestCase1.txt

```
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.
```

Gambar 19. Isi file TestCase1.txt



Gambar 18. Proses input directory TestCase1.txt



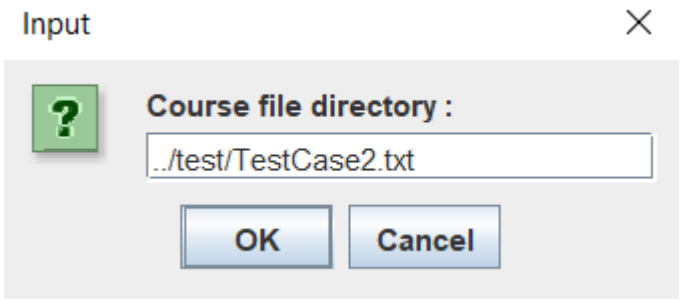
Gambar 17. Hasil output TestCase1.txt

4.2. TestCase2.txt

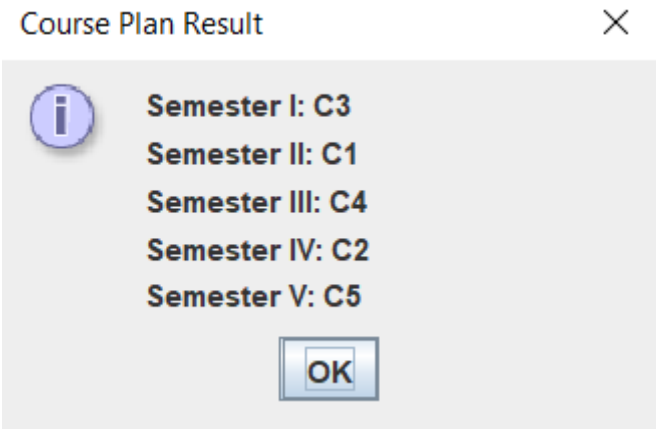
TestCase2.txt

```
C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.
```

Gambar 21. isi file TestCase2.txt

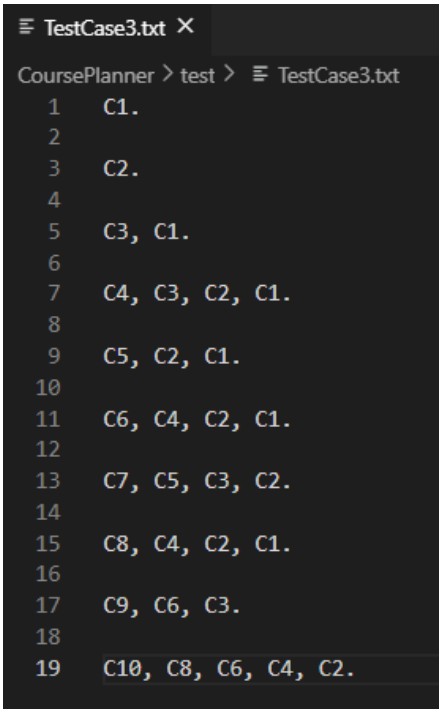


Gambar 20. Proses input directory TestCase2.txt

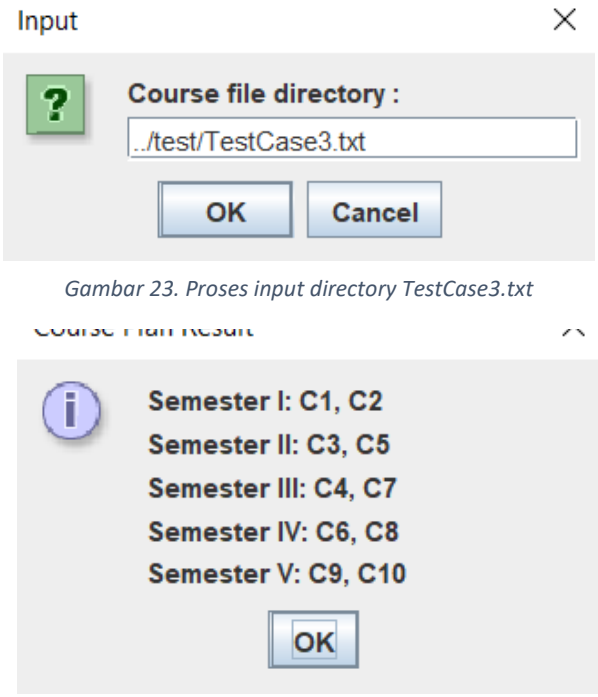


Gambar 22. Hasil output TestCase2.txt

4.3. TestCase3.txt

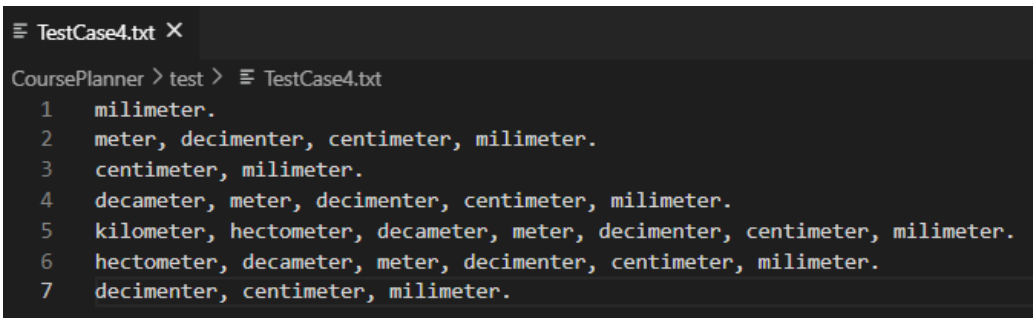


Gambar 25. Isi file TestCase3.txt

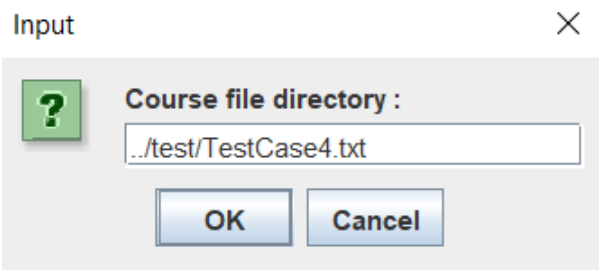


Gambar 24. Hasil output TestCase3.txt

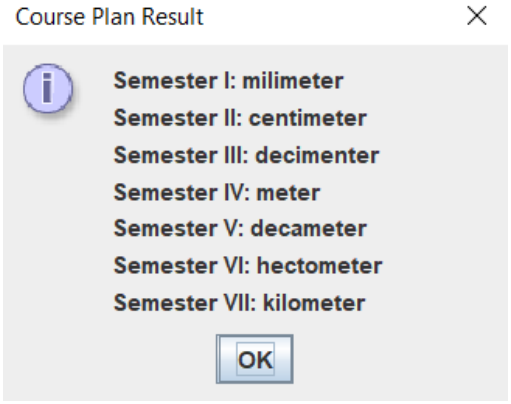
4.4. TestCase4.txt



Gambar 26. Isi file TestCase4.txt



Gambar 28. Proses input directory TestCase4.txt



Gambar 27. Hasil output TestCase4.txt

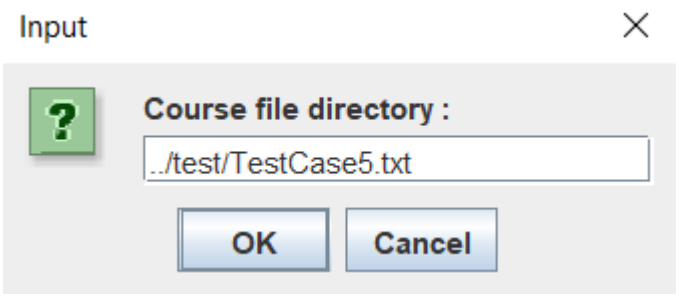
4.5. TestCase5.txt

TestCase5.txt

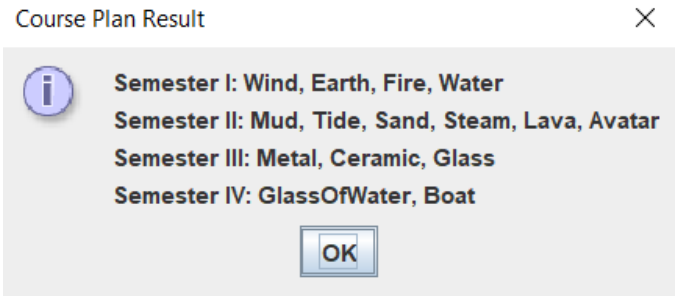
CoursePlanner > test > TestCase5.txt

```
1   Wind.
2   Earth.
3   Fire.
4   Water.
5
6   Mud, Water, Earth.
7   Tide, Wind, Water.
8   Sand, Earth, Wind.
9   Steam, Water, Fire.
10  Lava, Earth, Fire.
11
12  Metal, Lava, Water.
13  Ceramic, Mud, Fire.
14  Glass, Sand, Fire.
15
16  GlassOfWater, Glass, Water.
17
18  Boat, Metal, Tide, Steam.
19
20  Avatar, Wind, Water, Earth, Fire.
```

Gambar 31. Isi file TestCase5.txt



Gambar 30. Proses input directory TestCase5.txt



Gambar 29. Hasil output TestCase5.txt

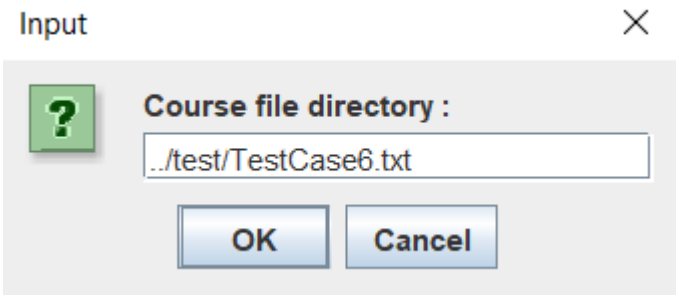
4.6. TestCase6.txt

TestCase6.txt

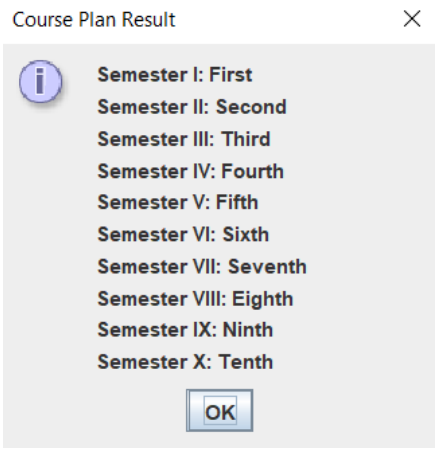
CoursePlanner > test > TestCase6.txt

```
1   First.
2   Second, First.
3   Third, Second.
4   Fourth, Third.
5   Fifth, Fourth.
6   Sixth, Fifth.
7   Seventh, Sixth.
8   Eighth, Seventh.
9   Ninth, Eighth.
10  Tenth, Ninth.
```

Gambar 33. Isi file TestCase6.txt

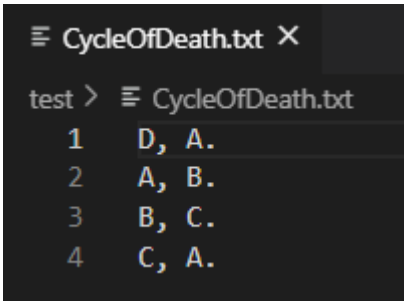


Gambar 32. Proses input directory TestCase6.txt

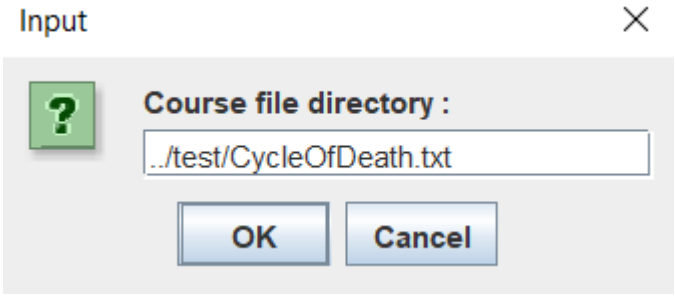


Gambar 34. Hasil output TestCase6.txt

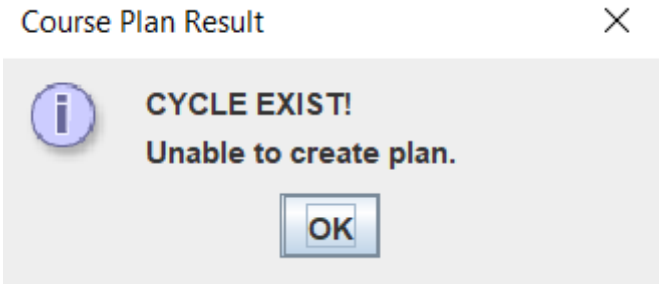
4.7. CycleOfDeath.txt



Gambar 36. Isi file CycleOfDeath.txt

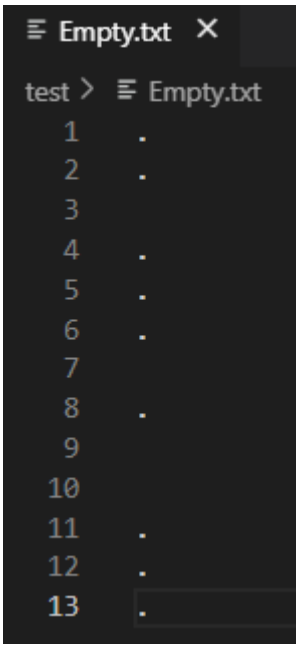


Gambar 35. Proses input directory CycleOfDeath.txt

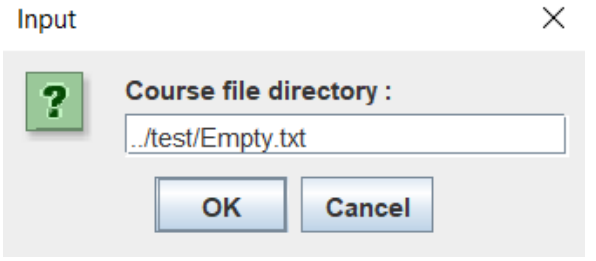


Gambar 37. Hasil output CycleOfDeath.txt

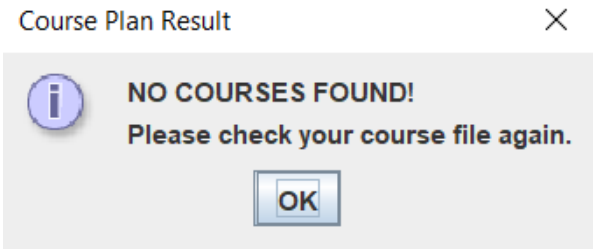
4.8. Empty.txt



Gambar 39. Isi file Empty.txt



Gambar 40. Proses input directory Empty.txt



Gambar 38. Hasil output Empt.txt

BAB V
ALAMAT *REPOSITORY* BERISI KODE PROGRAM

Berikut adalah pranala untuk menuju alamat *Repository GitHub* yang berisi kode program.

https://github.com/BeforeLast/CoursePlanner

BAB VI
TABEL PENILAIAN

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

Tabel 1. Tabel Penilaian