TUGAS KECIL IF2211 STRATEGI ALGORITMA
RENCANA KULIAH

Oleh

13519202 Farhan Yusuf Akbar

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

BAB 1

DESKRIPSI TUGAS

Pada tugas kali ini, mahasiswa diminta membuat aplikasi sederhana yang dapat menyusun rencana pengambilan kuliah, dengan memanfaatkan algoritma Decrease and Conquer. Penyusunan Rencana Kuliah diimplementasikan dengan menggunakan pendekatan Topological Sorting. Berikut akan dijelaskan tugas yang dikerjakan secara detail.

1. Aplikasi akan menerima daftar mata kuliah beserta prasyarat yang harus diambil seorang mahasiswa sebelum mengambil mata kuliah tersebut. Daftar mata kuliah tersebut dituliskan dalam suatu file teks dengan format:

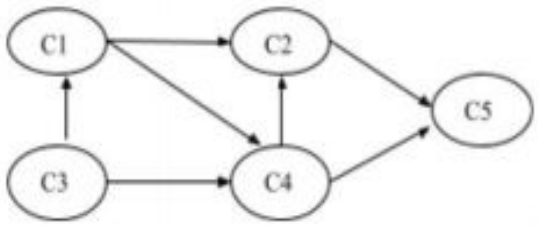
<kode_kuliah_1>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah
prasyarat - 3>.
<kode_kuliah_2>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>.
<kode_kuliah_3>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah
prasyarat - 3>, <kode kuliah prasyarat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

Sebuah kode kuliah mungkin memiliki nol atau lebih prasyarat kuliah. Kode kuliah bisa diambil pada suatu semester jika semua prasyaratnya sudah pernah diambil di semester sebelumnya (tidak harus 1 semester sebelumnya). Asumsi semua kuliah bisa diambil di sembarang semester, baik semester ganjil maupun semester genap. Sebagai contoh, terdapat 5 kuliah yang harus diambil seorang mahasiswa dengan daftar prerequisite

dalam file teks sebagai berikut. Dari file teks terlihat bahwa kuliah C3 tidak memiliki prerequisite.

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

Asumsi untuk persoalan ini, kuliah dan prerequisite nya pasti berupa Directed Acyclic Graph (DAG) dan representasinya dapat dilihat pada gambar di bawah ini.



2. Dari file teks yang telah diterima, ditentukan kuliah apa saja yang bisa diambil di semester 1, semester 2, dan seterusnya. Sebuah kuliah tidak mungkin diambil pada semester yang sama dengan prerequisite-nya. Untuk menyederhanakan persoalan, tidak ada Batasan banyaknya kuliah yang bisa diambil pada satu semester.

BAB 2

LANDASAN TEORI

2.1 Algoritma Decrease and Conquer

Algoritma Decrease and Conquer adalah algoritma yang menyelesaikan suatu permasalahan komputasi dengan mereduksi persoalan menjadi dua upa-persoalan (sub-problem) yang lebih kecil. Namun, berbeda dengan Algoritma Divide and Conquer, algoritma ini hanya memproses satu sub-persoalan saja.

Algoritma Greedy terdiri dari dua tahapan.

1. Decrease: mereduksi persoalan menjadi beberapa persoalan yang lebih kecil (biasanya dua upa-persoalan).
2. Conquer: memproses satu upa-persoalan secara rekursif

Algoritma Greedy dapat memecahkan beberapa persoalan sebagai berikut.

1. Decrease by a constant: ukuran instans persoalan direduksi sebesar konstanta yang sama setiap iterasi algoritma. Biasanya konstanta = 1. 2.
2. Decrease by a constant factor: ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi algoritma. Biasanya faktor konstanta = 2. 3.
3. Decrease by a variable size: ukuran instans persoalan direduksi bervariasi pada setiap iterasi algoritma

2.2 Langkah-Langkah Algoritma

Pada tugas kecil ini, Algoritma Decrease and Conquer diimplementasikan dengan pendekatan Topological Sorting. Pendekatan Topological Sorting ini hanya berlaku pada Directed Acyclic Graph (DAG). Dalam definisinya, Topological Sorting adalah pengurutan linear sedemikian rupa sehingga untuk setiap u -> v, simpul u muncul sebelum v dalam urutannya.

Langkah-langkah Topological Sorting pada tugas ini sebagai berikut.

1. Dari graf (DAG) yang terbentuk, hitung semua derajat-masuk (in-degree) setiap simpul, yaitu banyaknya busur yang masuk pada simpul tersebut. Pada contoh kasus di spesifikasi tugas, maka derajat-masuk tiap simpul adalah sebagai berikut.

   C1 : 1

   C2 : 2

C3 : 0

C4 : 2

C5 : 2

2. Pilih sembarang simpul yang memiliki derajat-masuk 0. Pada kasus di spesifikasi tugas, pilih simpul C3.
3. Ambil simpul tersebut, dan hilangkan simpul tersebut beserta semua busur yang keluar dari simpul tersebut pada graf, dan kurangi derajat simpul yang berhubungan dengan simpul tersebut dengan 1. Setelah simpul C3 dipilih, maka derajat simpul yang lain menjadi sebagai berikut.

   C1 : 0

   C2 : 2

   C4 : 1

   C5 : 2
4. Ulangi langkah (2) dan (3) hingga semua simpul pada DAG terpilih. Untuk kasus pada spesifikasi tugas, setelah simpul terakhir dipilih rencana kuliah yang dihasilkan adalah sebagai berikut.

   Semester I : C3

   Semester II : C1

   Semester III : C4

   Semester IV : C2

   Semester V : C5.

BAB 3

IMPLEMENTASI PROGRAM

3.1 Source Code

Bahasa yang digunakan untuk mengimplementasikan program adalah bahasa Java. Source code, cara kompilasi dan cara run program dapat diakses melalui link berikut.

https://github.com/farhanyusuf/TucilStima2

Class Graph

```
class Graph {
    //Prosedur untuk menge-print Graph dengan representasi list of list adjacent nodes
    public static void makeGraph(ArrayList<ArrayList<String>> matkulPrereqList)
    {
        for (int i = 0 ; i < matkulPrereqList.size(); i++) {
            for (int j = 0 ; j < matkulPrereqList.get(i).size(); j++) {
                if (matkulPrereqList.get(i).size() == 1) {
                    System.out.print(matkulPrereqList.get(i).get(j) + ".");
                }
                else{
                    if (j+1 < matkulPrereqList.get(i).size()){
                        if(j+1 == matkulPrereqList.get(i).size()-1){
                            System.out.print(matkulPrereqList.get(i).get(0) + " <- " + matkulPrereqList.get(i).get(j+1) + ".");
                        }
                        else{
                            System.out.print(matkulPrereqList.get(i).get(0) + " <- " + matkulPrereqList.get(i).get(j+1) + ", ");
                        }
                    }

                }
            }
            System.out.print('\n');
        }
    }
```

```
    //Fungsi untuk membaca file.txt dan data dalam file disimpan menjadi list of list matkul dan prerequisite
    public static ArrayList<ArrayList<String>> ReadFile(String filename) throws IOException{
        File file = new File(filename);
        ArrayList<String> arrayaku = new ArrayList<String>(100);
        ArrayList<ArrayList<String>> matkulPrereqList = new ArrayList<ArrayList<String>>(100);
        Scanner sc = new Scanner(file).useDelimiter(System.getProperty("line.separator"));


        while(sc.hasNextLine()){
            arrayaku.add(sc.nextLine());
        }

        for(int i = 0 ; i < arrayaku.size() ; i++){
            ArrayList<String> tempList = new ArrayList<>(Arrays.asList(arrayaku.get(i).split(",")));
            matkulPrereqList.add(tempList);
        }
        return matkulPrereqList;
    }
```

```java
//Fungsi untuk menghitung banyaknya edge yang masuk ke node matkul
public static ArrayList<Integer> HitungEdgeMasuk(ArrayList<ArrayList<String>> matkulPrereqList){
    ArrayList<Integer> banyakEdge = new ArrayList<Integer>(100);
    for(int i = 0;i < matkulPrereqList.size();i++){
        banyakEdge.add(matkulPrereqList.get(i).size()-1);
    }
    return banyakEdge;
}


//Prosedur untuk menghapus node matakuliah dari Graph
public static void deleteFromGraph(ArrayList<ArrayList<String>> matkulPrereqList, String node){
    for(int m = 0; m < matkulPrereqList.size() ; m++){
        for(int n = 0; n < matkulPrereqList.get(m).size() ; n++){
            if (node.equals(matkulPrereqList.get(m).get(n))){
                matkulPrereqList.get(m).remove(n);
            }
        }
        if(matkulPrereqList.get(m).isEmpty()){
            matkulPrereqList.remove(m);
            m = m-1;
        }
    }
}
```

```java
//Fungsi yang mengembalikan List mata kuliah dari Graph
public static ArrayList<String> ListMatkul(ArrayList<ArrayList<String>> matkulPrereqList){
    ArrayList<String> listMatkul = new ArrayList<String>(100);
    for(int i = 0; i < matkulPrereqList.size() ; i++){
        listMatkul.add(matkulPrereqList.get(i).get(0));
    }
    return listMatkul;
}


//Fungsi yang memproses Graph mata kuliah dengan pendekatan decrease and conquer melalui topological sort
public static ArrayList<String> DecreaseandConquer(ArrayList<ArrayList<String>> _matkulPrereqList, ArrayList<String> _listMatkul,
ArrayList<Integer> _banyakEdge) {
    int a = 1;
    ArrayList<String> rencanakuliah = new ArrayList<String>(100);
    while(!_matkulPrereqList.isEmpty()){
        int i = 0;
        boolean found = false;
        while (i < _banyakEdge.size() && !found){
            found = _banyakEdge.get(i) == 0;
            if(found == true){
                break;
            }
            else{
            i++;
            }
        }
        deleteFromGraph(_matkulPrereqList,_listMatkul.get(i));
        rencanakuliah.add(_listMatkul.get(i));
        _banyakEdge = HitungEdgeMasuk(_matkulPrereqList);
        _listMatkul = ListMatkul(_matkulPrereqList);
    }
    return rencanakuliah;
}
```

```java
//Prosedur untuk menge-print rencana kuliah
public static void printRencanaKuliah(ArrayList<String> rencanakuliah, ArrayList<ArrayList<String>> _matkulPrereqListconsts){
    int x = 1;
    for (int i = 0 ; i < rencanakuliah.size() ; i++ ){
        if (i == 0){
            System.out.print("Semester " + x + " : " + rencanakuliah.get(i));
            x++;}
        if (i>0){
            if (!isKuliahdiambilbersamaPrerequisite(rencanakuliah.get(i),rencanakuliah.get(i-1), _matkulPrereqListconsts)){
                System.out.print(", " + rencanakuliah.get(i));
            }
            else{
                System.out.print("\nSemester " + x + " : " + rencanakuliah.get(i));
                x++;
            }
        }
    }
}


//Fungsi yang menge-check apakah kuliah yang diambil pada suatu semester bersamaan dengan prerequisitenya
public static boolean isKuliahdiambilbersamaPrerequisite(String kuliah, String apakahPrereq, ArrayList<ArrayList<String>>
_matkulPrereqListconsts){
    boolean found = false;
    for (int i = 0 ; i < _matkulPrereqListconsts.size() ; i++){
        if (kuliah.equals(_matkulPrereqListconsts.get(i).get(0))){
            int iterator = 0;
            while (iterator < _matkulPrereqListconsts.get(i).size() && !found){
                found = apakahPrereq.equals(_matkulPrereqListconsts.get(i).get(iterator));
                iterator++;
            }
        }
    }
    return found;
```

Class Main

```java
public class Main {
    public static void main(String[] args)throws Exception
    {
        String namafile = "test3.txt";
        ArrayList<ArrayList<String>> matkulPrereqList = Graph.ReadFile(namafile);
        ArrayList<ArrayList<String>> matkulPrereqListconsts = Graph.ReadFile(namafile);
        System.out.println("-----------------GRAPH-----------------");
        Graph.makeGraph(matkulPrereqList);
        ArrayList<Integer> banyakEdge = Graph.HitungEdgeMasuk(matkulPrereqList);
        ArrayList<String> listMatkul = Graph.ListMatkul(matkulPrereqList);
        ArrayList<String> rencanakuliah = Graph.DecreaseandConquer(matkulPrereqList, listMatkul, banyakEdge);
        System.out.println("-------------RENCANA KULIAH--------------");
        Graph.printRencanaKuliah(rencanakuliah, matkulPrereqListconsts);
        System.out.println("\n---------------------------------------");
```

BAB 4

PENGUJIAN

4.1 Hasil Eksekusi Program

No	Input	Output
1	src > test1.txt 1 KU1102,MA1101 2 IF1210,KU1102,KU1024 3 MA1101 4 KU1024,KU1102,MA1101 5 IF2110,IF1210,KU1024	----------GRAPH---------- KU1102 <- MA1101. IF1210 <- KU1102, IF1210 <- KU1024. MA1101. KU1024 <- KU1102, KU1024 <- MA1101. IF2110 <- IF1210, IF2110 <- KU1024. ----------RENCANA KULIAH---------- Semester 1 : MA1101 Semester 2 : KU1102 Semester 3 : KU1024 Semester 4 : IF1210 Semester 5 : IF2110 ----------
2	src > test2.txt 1 KU1102,MA1101 2 IF1210,KU1102,KU1024 3 MA1101 4 KU1024,KU1102,MA1101 5 IF2110,IF1210,KU1024 6 IF2230,IF2110,IF2130 7 IF2123,IF2110 8 IF2130,EL1200 9 EL1200	----------GRAPH---------- KU1102 <- MA1101. IF1210 <- KU1102, IF1210 <- KU1024. MA1101. KU1024 <- KU1102, KU1024 <- MA1101. IF2110 <- IF1210, IF2110 <- KU1024. IF2230 <- IF2110, IF2230 <- IF2130. IF2123 <- IF2110. IF2130 <- EL1200. EL1200. ----------RENCANA KULIAH---------- Semester 1 : MA1101 Semester 2 : KU1102 Semester 3 : KU1024 Semester 4 : IF1210 Semester 5 : IF2110 Semester 6 : IF2123, EL1200 Semester 7 : IF2130 Semester 8 : IF2230 ----------

3	src > test3.txt 1 IF2230,IF2110,IF2130 2 IF2123,IF2110 3 IF2130 4 IF2110	----------GRAPH---------- IF2230 <- IF2110, IF2230 <- IF2130. IF2123 <- IF2110. IF2130. IF2110. ----------RENCANA KULIAH---------- Semester 1 : IF2130, IF2110 Semester 2 : IF2230, IF2123 ----------
4	src > test4.txt 1 IF1210 2 MA1101 3 KU1024 4 IF2110 5 IF2230 6 IF2123 7 IF2130	----------GRAPH---------- IF1210. MA1101. KU1024. IF2110. IF2230. IF2123. IF2130. ----------RENCANA KULIAH---------- Semester 1 : IF1210, MA1101, KU1024, IF2110, IF2230, IF2123, IF2130 ----------
5	src > test5.txt 1 KU1102,MA1101 2 IF1210 3 MA1101 4	----------GRAPH---------- KU1102 <- MA1101. IF1210. MA1101. ----------RENCANA KULIAH---------- Semester 1 : IF1210, MA1101 Semester 2 : KU1102 ----------
6	src > test6.txt 1 IF2130,MA1101 2 MA1101	----------GRAPH---------- IF2130 <- MA1101. MA1101. ----------RENCANA KULIAH---------- Semester 1 : MA1101 Semester 2 : IF2130 ----------

7	src > test7.txt 1 KU1102,MA1101 2 IF1210,KU1102 3 KU1024,KU1102 4 IF2110,IF1210 5 IF2230,IF2110 6 IF2123,IF2110 7 IF2130,MA1101 8 MA1101 9	----------GRAPH---------- KU1102 <- MA1101. IF1210 <- KU1102. KU1024 <- KU1102. IF2110 <- IF1210. IF2230 <- IF2110. IF2123 <- IF2110. IF2130 <- MA1101. MA1101. ----------RENCANA KULIAH---------- Semester 1 : MA1101 Semester 2 : KU1102 Semester 3 : IF1210, KU1024, IF2110 Semester 4 : IF2230, IF2123, IF2130 ----------
8	src > test8.txt 1 MA1101	----------GRAPH---------- MA1101. ----------RENCANA KULIAH---------- Semester 1 : MA1101 ----------

Poin	Ya	Tidak
Program berhasil dikompilasi		
Program berhasil running		
Program dapat menerima berkas input dan menuliskan output		
Luaran sudah benar untuk semua kasus input		