Penyusunan Rencana Kuliah dengan Topological Sort (Penerapan Decrease and Conquer) dalam Bahasa JAVA

Laporan Tugas Kecil II
Mata kuliah IF2211 Strategi Algoritma

Disusun Oleh :

Juan Louis Rombetasik (13519075)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Semester 2 Tahun 2020/2021

ALGORTIMA TOPO-SORT (public static void solver)

Merekursif hingga list kosong, List yang berisi Object Graph dengan dan men pop indeks 0 dari list

Memasukkan ke himpunan solusi mereka yang tidak memiliki prerequisite

Memasukkan ke himpunan solusi mereka yang memliki prerequisite yang sudah di dalam himpunan solusi

Jika tidak dimasukkan kedalam himpunan solusi maka isi list pada saat itu akan dimasukkan ke dalam List Graph ke posisi paling akhir

Menoutput (log) himpunan list solusi (satu mata kuliah per semester) jika List sudah kosong (semua Mata kuliah sudah masuk kedalam himpunan solusi)

https://github.com/mizuday/Tucil2_13519075

KODE PROGRAM

```
class Main {
    public static void main(String[] args) {
        while (true) {
            System.out.println();
            Scanner sc= new Scanner(System.in);
            System.out.print("nama file (tanpa txt): ");
            String namafile = sc.nextLine();

            List<Matkul> graph = Matkul.fileToListDAG("../test/" + namafile + ".txt");
            List<Matkul> graphCopy = Matkul.copyListMatkul(graph);
            List<String> solusi = new ArrayList<>();
            Matkul.solver(graph, solusi);

            Matkul.printSolusi(graphCopy,solusi);

            System.out.print("solve lagi? y/n >> ");
            String in = sc.nextLine().toLowerCase();
            if (in.equals("n")|| in.equals("no")){
                break;
            }
        }
    }
}

class Matkul{
    private String nama;
    private List<String> prereq;
```

```java
public Matkul(String nama){
    this.nama = nama;
    prereq = new ArrayList<>();
}

public Matkul(Matkul nama){
    this.nama = nama.getNama();
    this.prereq = new ArrayList<>();
    this.prereq.addAll(nama.getPrereq());
}

public String getNama() {
    return nama;
}
public List<String> getPrereq(){
    return prereq;
}

public void tambahPrereq(String nama){
    prereq.add(nama);
}

public void kurangPrereq(String nama){
    int counter = 0;
    for(String pre : prereq ){
        if (pre.equals(nama)){
            prereq.remove(counter);
            break;
        }
        counter++;
    }
}

public boolean hasPrereq(){
    return prereq.size() > 0;
}

public static List<Matkul> copyListMatkul(List<Matkul> src){
    List<Matkul> copy = new ArrayList<>();
    for (Matkul matkul : src){
        Matkul temp = new Matkul(matkul);
        copy.add(temp);
    }
    return copy;
```

```
}

public static void menghapusPrereq(List<Matkul> dag, String nama){
    for (Matkul matkul : dag){
        matkul.kurangPrereq(nama);
    }
}

public static boolean punyaPreqSama(List<Matkul> listDAG, String namaMatkul, List<String> matkul) {
    for (Matkul dag : listDAG){
        if (dag.getNama().equals(namaMatkul)){
            for (String str : matkul){
                for (String lagi: dag.getPrereq()){
                    if (str.equals(lagi))
                        return true;
                }
            }
        }
    }
    return false;
}

public static List<Matkul> fileToListDAG(String namafile){
    List<Matkul> listDAG = new ArrayList<>();
    List<String[]> data = new ArrayList<>();
    try {
        BufferedReader reader = new BufferedReader(new FileReader(namafile));
        String line = reader.readLine();
        while (line != null) {
            line = line.replace(".", "");
            String[] temp = line.split(", ");
            data.add(temp);
            line = reader.readLine();
        }
        reader.close();

        for (int i = 0; i < data.size(); i++) {
            listDAG.add(new Matkul(data.get(i)[0]));
            for (String string : data.get(i)){
                if (!string.equals(data.get(i)[0]))
                listDAG.get(i).tambahPrereq(string);
            }
        }
    } catch (IOException e) {
        System.out.println("Nama file salah");
```

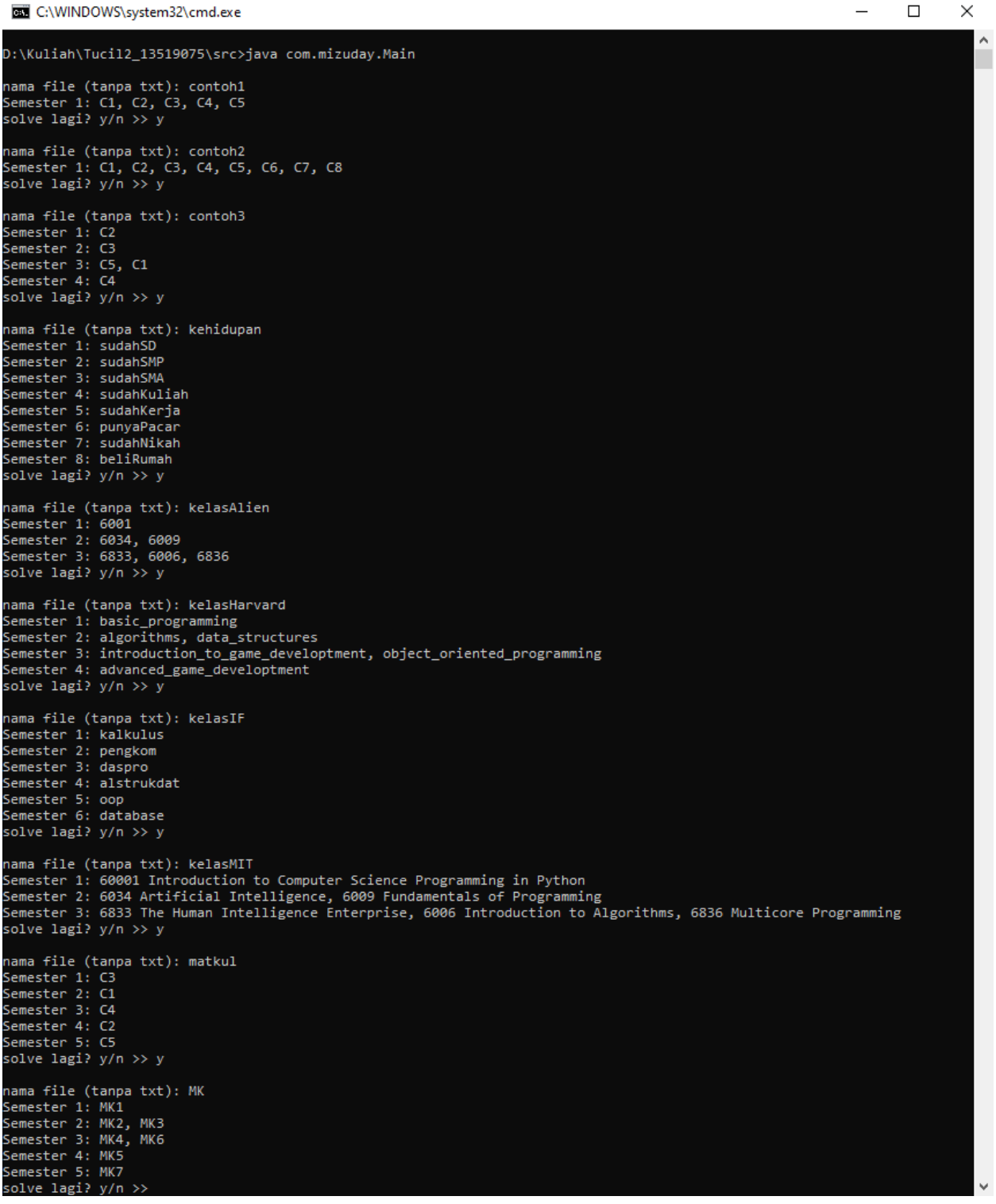
```
        }
        return listDAG;
    }

    public static void solver(List<Matkul> listDAG, List<String> solusi){
        if (!listDAG.isEmpty()){
            try {
                // Jika punya prereq tambahkan ke paling akhir
                if (listDAG.get(0).hasPrereq()) {
                    listDAG.add(listDAG.get(0));
                }
                // Jika tidak punya prereq, tambahkan ke solusi dan hapus prereq
                else {
                    solusi.add(listDAG.get(0).getNama());
                    menghapusPrereq(listDAG, listDAG.get(0).getNama());
                }
                // Buang paling awal rekursif lagi
                listDAG.remove(0);
                solver(listDAG, solusi);
            } catch (NullPointerException e) {
                System.out.println("NullPointerException Caught");
            }
        }
    }

    public static void printSolusi(List<Matkul> listDAGCUI, List<String> solusi){
        int smt = 1;
        for (int i = 0; i < solusi.size(); i++) {
            System.out.print("Semester "+ (smt) + ": " + solusi.get(i));
            List<String> printSebelumnya = new ArrayList<>();
            printSebelumnya.add(solusi.get(i));

            while (i < solusi.size()-1 && !punyaPreqSama(listDAGCUI,solusi.get(i+1), printSebelumnya)){
                System.out.print(", "+solusi.get(i+1));
                i++;
            }
            smt++;
            System.out.println();
        }
    }
}
```

SKRINSHUT OUTPUT PROGRAM



PENGUJIAN

test > contoh1.txt

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

test > contoh2.txt

```
C1.
C2.
C3.
C4.
C5.
C6
C7.
C8.
```

test > contoh3.txt

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

test > kehidupan.txt

```
punyaPacar, sudahKerja.
sudahKerja, sudahKuliah.
sudahSMP, sudahSD.
sudahSD.
beliRumah, sudahNikah, sudahKerja.
sudahKuliah, sudahSMA.
sudahSMA, sudahSMP.
sudahNikah, punyaPacar.
```

test > kelasAlien.txt

```
6034, 6001.
6006, 6009.
6009, 6001.
6001.
6833, 6034.
6836, 6006.
```

test > kelasHarvard.txt

```
algorithms, basic_programming.
data_structures, basic_programming.
introduction_to_game_developtment, data_structures, algorithms.
advanced_game_developtment, introduction_to_game_developtment, ob
basic_programming.
object_oriented_programming, data_structures, algorithms.
```

test > kelasIF.txt

```
database, oop.
oop, alstrukdat.
alstrukdat, daspro.
pengkom, kalkulus.
daspro, pengkom.
kalkulus.
```

test > kelasMIT.txt

```
6034 Artificial Intelligence, 60001 Introduction to Computer Science Programming in Python.
6006 Introduction to Algorithms, 6009 Fundamentals of Programming.
6009 Fundamentals of Programming, 60001 Introduction to Computer Science Programming in Python.
60001 Introduction to Computer Science Programming in Python.
6833 The Human Intelligence Enterprise, 6034 Artificial Intelligence.
6836 Multicore Programming, 6006 Introduction to Algorithms.
```

test > matkul.txt

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

test > MK.txt

```
MK1.
MK2, MK1.
MK3, MK1.
MK4, MK1, MK2.
MK5, MK1, MK2, MK4, MK6.
MK6, MK1, MK3.
MK7, MK5, MK6.
```

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