

# **Penyusunan Rencana Kuliah dengan *Topological Sort* (Penerapan *Decrease and Conquer*)**

Diajukan untuk memenuhi tugas kecil matakuliah IF2211 Strategi Algoritma

Oleh:  
13519036  
Andrew



**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2020**

## A. Deskripsi Persoalan

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>.  
.  
.  
.
```

**Gambar 1. Format file teks untuk masukan daftar kuliah**

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.

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 prerequisitenya. Untuk menyederhanakan persoalan, tidak ada Batasan banyaknya kuliah yang bisa diambil pada satu semester.

Asumsi untuk persoalan ini, kuliah dan prerequisite nya pasti berupa Directed Acyclic Graph (DAG).

## B. Topological Sort dan Kaitannya dengan Algoritma Decrease and Conquer

Dalam kehidupan sehari-hari, terdapat beberapa permasalahan yang dapat dimodelkan dalam bentuk graf berarah, dimana sebuah kejadian harus terjadi lebih dahulu sebelum kejadian lain. Salah satu diantaranya adalah penyusunan rencana kuliah. Terdapat beberapa mata kuliah yang memiliki prasyarat mata kuliah, yaitu mata kuliah yang harus diambil terlebih dahulu sebelum dapat mengambil mata kuliah tersebut. Salah satu pendekatan *topological sorting* dalam penyusunan rencana kuliah memiliki langkah-langkah sebagai berikut.

1. Pilih mata kuliah yang tidak memiliki prasyarat atau yang prasyaratnya telah terpenuhi.
2. Ambil masing-masing mata kuliah tersebut dan simpan pada rencana kuliah.

3. Kurangi derajat mata kuliah yang prasyaratnya adalah mata kuliah tersebut dengan 1.
4. Ulangi langkah 1-3 sampai semua mata kuliah telah diambil.

Algoritma *Decrease and Conquer* adalah salah satu metode algoritma yang mereduksi persoalan menjadi dua upa-persoalan, namun hanya memproses satu upa-persoalan. Penyelesaian satu upa-persoalan ini diharapkan dapat menyelesaikan persoalan secara keseluruhan. Algoritma *Decrease and Conquer* memiliki dua jenis implementasi, yaitu implementasi secara rekursif dan implementasi secara iteratif. Terdapat tiga varian dari algoritma *Decrease and Conquer*, yaitu:

1. *Decrease by a constant*  
Ukuran instans persoalan direduksi sebesar konstanta yang sama setiap iterasi.
2. *Decrease by a constant factor*  
Ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi.
3. *Decrease by a variable size*  
Ukuran instans persoalan yang direduksi bervariasi pada setiap iterasi.

*Topological sort* adalah salah satu aplikasi algoritma *Decrease and Conquer*. Pada umumnya, *topological sort* adalah aplikasi algoritma *Decrease and Conquer* dengan varian *decrease by a constant*. Namun, pada persoalan ini, terdapat beberapa asumsi yang perlu diperhatikan, seperti semua mata kuliah dapat diambil di sembarang semester dan tidak adanya batasan jumlah mata kuliah yang dapat diambil dalam satu semester. Hal tersebut berakibat pada reduksi instans yang mungkin bervariasi setiap iterasinya. Dengan demikian, persoalan penyusunan rencana kuliah ini adalah penerapan *topological sort* dengan varian *decrease by a variable size*.

### C. Source Code

```
GraphAndrew.java
{ Mendefinisikan kelas GraphAndrew }

package src;

/*
  Disclaimer :
  Beberapa constructor, getter, setter, dan metode tidak
  didefinisikan karena tidak dipakai
*/

// Pengimpor sejati
import java.util.*;

public class GraphAndrew {

    /*--- Kelas Butuh Atribut ---*/
    private List<VerticeAndrew> graf; // Menyimpan simpul
    private int verticeCount; // Menyimpan jumlah simpul dalam graf
```

```

/*--- Kelas Butuh Ctor ---*/
GraphAndrew() {
    this.graf = new ArrayList<>();
    verticeCount = 0;
}

/*--- Kelas Butuh Getter ---*/
public List<VerticeAndrew> getGraf() {
    return graf;
}

public int getVerticeCount() {
    return verticeCount;
}

/*--- Kelas Butuh Metode ---*/
// Print graf
/*
 * Menuliskan graf dalam format <Nama simpul x>: <Nama simpul
yang berarah ke simpul x>.
 * "-" mengindikasikan bahwa simpul tidak memiliki derajat masuk.
 * Contoh:
 * C1: C2 C3
 * C2: -
 * C3: C2
 */
public void printGraph() {
    for (VerticeAndrew i : this.graf) {
        System.out.printf("%s: ", i.getVerticeName());
        List<String> listIn = i.getInDegreeFrom();
        if (listIn.size() != 0) {
            for (int j = 0; j < listIn.size(); j++) {
                System.out.format("%s ", listIn.get(j));
            }
        } else {
            System.out.print("-");
        }
        System.out.println();
    }
}

// Menambahkan simpul ke dalam graf
public void addVertice(VerticeAndrew werdna) {
    this.graf.add(werdna);
    verticeCount++;
}

// Menghapus simpul x dari graf dan menghapus simpul x di dalam
list simpul
// berarah masuk pada simpul lain
public void removeVertice(String vName) {
    int i = 0;

    while (i < verticeCount &&
!this.graf.get(i).getVerticeName().equals(vName)) {
        i++;
    }

    String deleted = this.graf.get(i).getVerticeName();
    this.graf.remove(i);
}

```

```

        verticeCount--;

        for (int j = 0; j < verticeCount; j++) {
            for (int k = 0; k < this.graf.get(j).getInDegreeCount();
k++) {
                if
                (this.graf.get(j).getInDegreeFrom().get(k).equals(deleted)) {
                    this.graf.get(j).deleteInDegreeFrom(k);
                    break;
                }
            }
        }

        // Membersihkan graf menjadi object baru
        public void cleanGraph() {
            this.graf.clear();
            this.verticeCount = 0;
        }
    }
}

```

## VerticeAndrew.java

{ Mendefinisikan kelas VerticeAndrew }

```

package src;

/*
    Disclaimer :
    Beberapa constructor, getter, setter, dan metode tidak
    didefinisikan karena tidak dipakai
*/

// Pengimpor sejati
import java.util.*;

public class VerticeAndrew {

    /*--- Kelas Butuh Atribut ---*/
    private String verticeName; // Menyimpan nama simpul
    private List<String> inDegreeFrom; // Menyimpan list simpul yang
berarah ke simpul tersebut
    private int inDegreeCount; // Menyimpan derajat simpul

    /*--- Kelas Butuh Ctor ---*/
    VerticeAndrew(String verticeName, List<String> inDegreeFrom) {
        this.verticeName = verticeName;
        this.inDegreeFrom = inDegreeFrom;
        this.inDegreeCount = inDegreeFrom.size();
    }

    /*--- Kelas Butuh Getter ---*/
    public String getVerticeName() {
        return this.verticeName;
    }

    public List<String> getInDegreeFrom() {
        return inDegreeFrom;
    }
}

```

```

    public int getInDegreeCount() {
        return inDegreeCount;
    }

    /*--- Kelas Butuh Metode ---*/
    // Menghapus simpul yang berarah ke simpul this
    public void deleteInDegreeFrom(int i) {
        this.inDegreeFrom.remove(i);
        inDegreeCount--;
    }
}

```

## TopoSortAndrew.java

{ Mendefinisikan kelas dan metode topological sorting }

```

package src;

// Pengimpor sejati
import java.util.*;

public class TopoSortAndrew {
    /*--- Bahkan Tempat Sorting Butuh Atribut ---*/
    private List<List<String>> hasilSorting; // Menyimpan hasil
    sorting
    public boolean invalid; // Mengecek apakah graf memiliki siklus
    atau graf tidak lengkap

    /*--- Bahkan Tempat Sorting Butuh Ctor ---*/
    TopoSortAndrew(GraphAndrew werdna) {
        hasilSorting = new ArrayList<>();
        TopologicalSorting(werdna);
    }

    /*--- Tempat Sorting Butuh Metode ---*/
    // Topological Sorting
    /*
    * Melakukan topological sorting.
    * Langkah:
    * 1. Ambil semua simpul yang tidak memiliki derajat masuk,
    simpan ke dalam sebuah list
    * 2. Tambahkan list simpul ke dalam hasil sorting.
    * 3. Hapus semua simpul tersebut dari graf.
    * 4. Ulangi langkah 1-3 sampai graf tidak memiliki simpul atau
    sebuah siklus ditemukan
    */
    public void TopologicalSorting(GraphAndrew werdna) {
        invalid = false;
        while (werdna.getVerticeCount() > 0 && !invalid) {
            List<String> listSorting = new ArrayList<>();
            int i;
            for (i = 0; i < werdna.getVerticeCount(); i++) {
                if (werdna.getGraf().get(i).getInDegreeCount() == 0)
            {
                listSorting.add(werdna.getGraf().get(i).getVerticeName());
            }
        }
    }
}

```

```

        if (listSorting.size() > 0) {
            hasilSorting.add(listSorting);

            for (i = 0; i < listSorting.size(); i++) {
                werdna.removeVertice(listSorting.get(i));
            }
        } else {
            invalid = true;
        }
    }
}

// Menampilkan output
public void printRencanaKuliah() {
    int i, j;
    int jumlahSem = hasilSorting.size();

    // Tambahan output
    if (jumlahSem > 8 && jumlahSem <= 14) {
        System.out.println("Peringatan: Anda hanya perlu
berkuliah selama 4 tahun.");
    }
    if (jumlahSem > 14) {
        System.out.println("Peringatan: Jangan menghabiskan hidup
anda dengan berkuliah saja.");
        jumlahSem = 14;
    }

    System.out.println("Berikut rencana kuliah Anda terurut
berdasarkan prerequisites-nya.");
    for (i = 0; i < jumlahSem; i++) {
        System.out.printf("%s", semester[i]);
        for (j = 0; j < hasilSorting.get(i).size(); j++) {
            System.out.print(hasilSorting.get(i).get(j));
            if (j != hasilSorting.get(i).size() - 1) {
                System.out.print(", ");
            } else {
                System.out.println();
            }
        }
    }
}

// ITB lulus paling lambat 7 tahun (14 semester)
public String[] semester = new String[] {
    "Semester I      : ",
    "Semester II     : ",
    "Semester III    : ",
    "Semester IV     : ",
    "Semester V      : ",
    "Semester VI     : ",
    "Semester VII    : ",
    "Semester VIII   : ",
    "Semester IX     : ",
    "Semester X      : ",
    "Semester XI     : ",
    "Semester XII    : ",
    "Semester XIII   : ",
    "Semester XIV    : " };
}

```

## EzLulus4Tahun.java

{ Main program }

```
package src;

// Pengimpor sejati
import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;

public class EzLulus4Tahun {
    public static void main(String[] args) {

        /*--- Kamus dengan Kearifan Lokal ---*/
        Scanner input = new Scanner(System.in); // instansiasi tukang
        baca input user
        GraphAndrew andrew = new GraphAndrew(); // instansiasi object
        graf

        int i; // index iterasi

        /*--- Algoritma tapi di OOP ---*/
        System.out.println("(input \"#\" untuk keluar)");
        System.out.print("Another test subject!: ");
        String namaFile = input.nextLine();

        // Looping program
        while (!namaFile.equals("#")) {
            try {
                namaFile = String.format("./test/%s", namaFile);
                File anotherTestSubject = new File(namaFile);

                // Mengecek apakah file kosong
                if (anotherTestSubject.length() == 0) {
                    throw new FileNotFoundException();
                }

                Scanner testSubject = new
Scanner(anotherTestSubject);

                // Memasukkan isi file ke dalam graf
                while (testSubject.hasNextLine()) {
                    String verticeName;
                    List<String> verticeIn = new ArrayList<>();
                    String[] ele =
(testSubject.nextLine()).split("[, . ]+");
                    verticeName = ele[0];

                    for (i = 1; i < ele.length; i++) {
                        verticeIn.add(ele[i]);
                    }
                    VerticeAndrew verticex = new
VerticeAndrew(verticeName, verticeIn);
                    andrew.addVertice(verticex);
                }
                testSubject.close();

                // Topological sort
                andrew.printGraph();
            }
        }
    }
}
```



```

        System.out.println();
        TopoSortAndrew topoSort = new TopoSortAndrew(andrew);

        // Mengecek 'seandainya' graf memiliki siklus atau
        graf tidak lengkap
        if (topoSort.invalid) {
            System.out.println("Graf bukan DAG (Directed
Acyclic Graph) atau graf tidak lengkap.\n");
        } else {
            topoSort.printRencanaKuliah();
            System.out.println();
        }

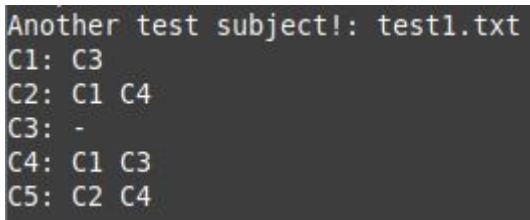
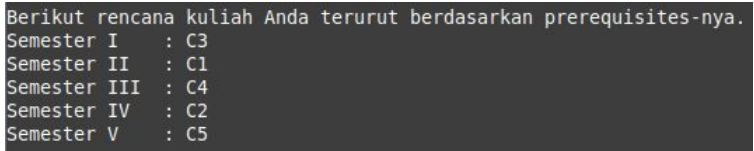
        andrew.cleanGraph(); // Reset graf menjadi semula
    } catch (FileNotFoundException e) {
        // File tidak ada di folder test atau file kosong
        System.out.println("Picked... the wrong test
subject...\n");
    }

    System.out.print("Another test subject!: ");
    namaFile = input.nextLine();
}

// Terminasi program
input.close();
System.out.println("Terima kasih telah menggunakan penyortir
ini ^_^");
}
}

```

#### D. Screenshot Penyelesaian Topological Sort

1	Input	
	Output	

2	Input	<pre> Another test subject!: test2.txt Matkul0: - Matkul1: Matkul3 Matkul2: Matkul0 Matkul9 Matkul3: Matkul0 Matkul4: Matkul1 Matkul3 Matkul7 Matkul5: Matkul4 Matkul6: Matkul0 Matkul2 Matkul10 Matkul7: Matkul6 Matkul8: Matkul4 Matkul12 Matkul9: - Matkul10: Matkul9 Matkul11: Matkul6 Matkul12: Matkul7 Matkul11 Matkul13: - </pre>
	Output	<pre> Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I   : Matkul0, Matkul9, Matkul13 Semester II  : Matkul2, Matkul3, Matkul10 Semester III : Matkul1, Matkul6 Semester IV  : Matkul7, Matkul11 Semester V   : Matkul4, Matkul12 Semester VI  : Matkul5, Matkul8 </pre>
3	Input	<pre> Another test subject!: test3.txt IF-A: - IF-B: IF-A IF-C: - IF-D: IF-A IF-E: IF-B IF-D IF-F: IF-C IF-D IF-G: IF-E IF-F IF-H: IF-E IF-I: IF-F IF-G IF-J: IF-G IF-H IF-K: - </pre>
	Output	<pre> Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I   : IF-A, IF-C, IF-K Semester II  : IF-B, IF-D Semester III : IF-E, IF-F Semester IV  : IF-G, IF-H Semester V   : IF-I, IF-J </pre>
4	Input	<pre> Another test subject!: test4.txt 1: 2 2: 3 3: 4 4: 5 5: 6 6: 7 7: 8 8: 9 9: 10 10: - </pre>

	Output	<pre> Peringatan: Anda hanya perlu berkuliah selama 4 tahun. Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I      : 10 Semester II     : 9 Semester III    : 8 Semester IV     : 7 Semester V      : 6 Semester VI     : 5 Semester VII    : 4 Semester VIII   : 3 Semester IX     : 2 Semester X      : 1 </pre>
5	Input	<pre> Another test subject!: test5.txt 1: 2 2: 3 3: 4 4: 5 5: 6 6: 7 7: 8 8: 9 9: 10 10: 11 11: 12 12: 13 13: 14 14: 15 15: 16 16: - </pre>
	Output	<pre> Peringatan: Jangan menghabiskan hidup anda dengan berkuliah saja. Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I      : 16 Semester II     : 15 Semester III    : 14 Semester IV     : 13 Semester V      : 12 Semester VI     : 11 Semester VII    : 10 Semester VIII   : 9 Semester IX     : 8 Semester X      : 7 Semester XI     : 6 Semester XII    : 5 Semester XIII   : 4 Semester XIV    : 3 </pre>
6	Input	<pre> Another test subject!: test6.txt ANDR1: ANDR6 ANDR7 ANDR9 ANDR2: - ANDR3: ANDR1 ANDR9 ANDR10 ANDR4: ANDR1 ANDR3 ANDR8 ANDR5: ANDR4 ANDR8 ANDR6: ANDR2 ANDR7: ANDR2 ANDR6 ANDR8: ANDR1 ANDR7 ANDR9: ANDR6 ANDR10: ANDR9 </pre>

	Output	Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I : ANDR2 Semester II : ANDR6 Semester III : ANDR7, ANDR9 Semester IV : ANDR1, ANDR10 Semester V : ANDR3, ANDR8 Semester VI : ANDR4 Semester VII : ANDR5
7	Input	Another test subject!: test7.txt C1: - C2: C1 C3: C2 C4: C3 C5: C1 C2 C6: C1 C2 C3 C5 C7: C2 C3 C4 C6 C8: C3 C4 C7 C9: C5 C6 C10: C5 C6 C7 C9 C11: C6 C7 C8 C10 C12: C7 C8 C11
	Output	Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I : C1 Semester II : C2 Semester III : C3, C5 Semester IV : C4, C6 Semester V : C7, C9 Semester VI : C8, C10 Semester VII : C11 Semester VIII : C12
8	Input	Another test subject!: test8.txt MA1101: - EL1200: MA1101 IF2110: - IF2130: - IF2123: MA1101 IF2210: IF2110 IF2220: MA1101 IF2230: - IF2240: - IF2250: - IF3170: IF2220 IF3110: IF2210 IF2110 IF3130: IF2230 IF3141: IF2240 IF2250 IF3150: IF2250 IF3151: IF2250 IF3210: IF2130 IF2110 IF3270: IF3170 IF2110 IF3230: IF3130 IF3250: IF3150 IF2250 IF3260: IF2130 IF2110 IF2123

	Output	Berikut rencana kuliah Anda terurut berdasarkan prerequisites-nya. Semester I : MA1101, IF2110, IF2130, IF2230, IF2240, IF2250 Semester II : EL1200, IF2123, IF2210, IF2220, IF3130, IF3141, IF3150, IF3151, IF3210 Semester III : IF3170, IF3110, IF3230, IF3250, IF3260 Semester IV : IF3270
--	--------	--

#### E. Alamat Repository

[https://github.com/andrcyes/Tucil2\\_13519036](https://github.com/andrcyes/Tucil2_13519036)

#### F. Checklist

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