

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

**LAPORAN**

**Sebagai Bagian dari Tugas Kecil 2 mata kuliah Strategi Algoritma IF2211 pada  
semester II Tahun Akademik 2020/2021**



**Oleh**

**Isabella Handayani Sumantri**

**13519081**

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

**2021**

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	<b>4</b>
1.1. Topological Sort	4
1.2. Algoritma Decrease and Conquer	4
1.3. Penerapan Algoritma Decrease and Conquer dalam Topological Sort	5
<b>BAB II</b>	<b>6</b>
2.1. Variabel Global	6
2.2. Prosedur start	6
2.3. Prosedur add	7
2.4. Prosedur deleteClass	8
2.5. Prosedur output	8
2.6. Prosedur toposort	9
2.7. Fungsi isAcyclic	10
2.8. Program Utama	10
<b>BAB III</b>	<b>11</b>
<b>Screenshot Uji</b>	<b>11</b>
3.1. Uji 1	11
3.1.1. Input	11
3.1.2. Output	11
3.2. Uji 2	11
3.2.1. Input	11
3.2.2. Output	12
3.3. Uji 3	13
3.3.1. Input	13
3.3.2. Output	13
3.4. Uji 4	14
3.4.2. Output	14
3.5. Uji 5	15
3.5.1. Input	15
3.5.2. Output	15
3.6. Uji 6	16
3.6.1. Input	16
3.6.2. Output	16
3.7. Uji 7	17

3.7.1. Input	17
3.7.2. Output	17
3.8. Uji 8	17
3.8.1. Input	17
3.8.2. Output	18
<b>BAB IV</b>	<b>19</b>
<b>4.1. Alamat Repository</b>	<b>19</b>
<b>4.2. Checklist</b>	<b>19</b>

# BAB I

## Teori Dasar

### 1.1. *Topological Sort*

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge  $u \rightarrow v$ , vertex  $u$  comes before  $v$  in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

*Topological sorting* adalah pengurutan linear simpul dalam graf sehingga untuk setiap busur berarah dari  $u$  ke  $v$ , simpul  $u$  akan berada di urutan yang lebih awal dari  $v$  dalam pengurutan. *Topological sorting* untuk graf tidak memungkinkan jika graf bukanlah *directed acyclic graph*.

Langkah-langkah pendekatan *topological sorting*

1. Dari graf (DAG) yang terbentuk, hitung semua derajat-masuk (in-degree) setiap simpul yaitu banyaknya busur yang masuk pada simpul tersebut.
2. Pilih sembarang simpul yang memiliki derajat-masuk 0.
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.
4. Ulangi langkah 2 dan 3 hingga semua simpul pada DAG terpilih.

### 1.2. *Algoritma Decrease and Conquer*

Algoritma *decrease and conquer* adalah metode perancangan algoritma dengan mereduksi persoalan menjadi dua upa-persoalan (sub-problem) yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja.

Algoritma *decrease and conquer* 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.

Terdapat tiga varian dari algoritma *decrease and conquer*:

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

### **1.3. Penerapan Algoritma *Decrease and Conquer* dalam *Topological Sort***

Penerapan algoritma *decrease and conquer* dalam *topological sorting* merupakan varian algoritma *decrease and conquer* versi *decrease by a variable size*. Ukuran instans persoalan akan direduksi bervariasi pada setiap iterasi algoritma berdasarkan banyaknya mata kuliah yang memiliki derajat masuk 0.

Adapun tahap algoritma *decrease and conquer* dalam tahapan *topological sort*

#### **1. Tahap *Decrease***

Algoritma akan mereduksi persoalan dengan menghilangkan simpul beserta semua busur yang keluar dari simpul tersebut pada graf, dan kurangi derajat simpul yang berhubungan dengan simpul tersebut dengan 1.

#### **2. Tahap *Conquer***

Memproses upa-persoalan berupa *directed acyclic graph* yang telah melalui tahap *decrease*. Kemudian, secara rekursif akan memproses *directed acyclic graph* sampai semua simpul pada *graph* diurutkan.

## BAB II

### Source Program

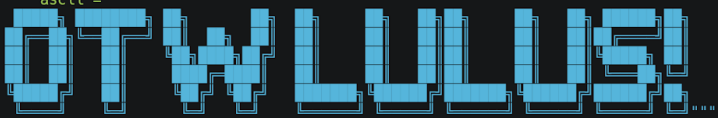
#### 2.1. Variabel Global

```
import roman  
  
text = []  
result = []
```

Pada bagian program ini dilakukan inisialisasi terhadap list berikut,

1. *text* : menyimpan data dari *file input*
2. *result* : menyimpan data yang sudah diproses

#### 2.2. Prosedur *start*

```
def start():  
    # Mencetak Logo Program  
    # KAMUS LOKAL  
    # ascii = string  
  
    # ALGORITMA  
  
    ascii = """  
      
    """  
    print(ascii)  
    print("SOLUSI 144 SKS ANDA")
```

Prosedur *start* digunakan untuk mencetak nama program ketika program pertama kali dijalankan.

### 2.3. Prosedur *add*

```
def add(file):
    # Menyimpan masukan dari file ke dalam list text

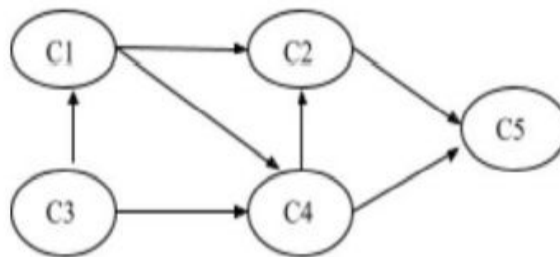
    # KAMUS LOKAL
    # string : string
    # row : array of string

    # ALGORITMA
    for kelas in (file):
        string = ""
        row = []
        for i in range(len(kelas)):
            if(kelas[i] == "," or kelas[i] == "."):
                row.append(string)
                string = ""
            elif(kelas[i] != "," and kelas[i] != "." and kelas[i] != " " and kelas[i] != "\n"):
                string = string + kelas[i]
        text.append(row)
```

Prosedur *add* digunakan untuk menyimpan masukan dari *file* ke dalam *list text*. Adapun masukan akan disimpan dalam bentuk *list of list* seperti berikut.

```
[['C1', 'C3'], ['C2', 'C1', 'C4'], ['C3'], ['C4', 'C1', 'C3'], ['C5', 'C2', 'C4']]
```

Struktur data *list of list* tersebut digunakan untuk mengimplementasikan *directed acyclic graph*. *Head* dari element *list* di dalam *list* adalah simpul dari *graph*, dan *tail* dari *list* adalah *list* mata kuliah *prerequisite* yang harus dipenuhi sebelum bisa mengambil mata kuliah *head*. Oleh karena itu, ilustrasi bentuk *list of list* tersebut menjadi berikut,



## 2.4. Prosedur *deleteClass*

```
def deleteClass(deleted):
    # Menghapus mata kuliah deleted yang tidak memiliki prerequisite
    # (busur menuju mata kuliah deleted = 0)
    # dari busur mata kuliah lain

    # ALGORITMA
    for i in range(len(text)):
        for j in range(len(text[i])):
            if(text[i][j] == deleted):
                text[i].pop(j)
                break
```

Prosedur *deleteClass* digunakan untuk menghapus busur dari mata kuliah *deleted* yaitu, mata kuliah dengan derajat masuk = 0 atau mata kuliah yang tidak memiliki *prerequisite* ke semua mata kuliah yang memiliki *prerequisite* mata kuliah *deleted*.

## 2.5. Prosedur *output*

```
def output():
    # Mencetak hasil topological sort ke layar

    # KAMUS LOKAL
    # count : int

    # ALGORITMA
    count = 1
    for semester in result:
        print("Semester", roman.toRoman(count), ":", end="")
        print(semester[0], end="")
        for i in range(1, len(semester)):
            print(",", semester[i], end="")
        print("\n")
        count += 1
```

Prosedur *output* digunakan untuk mencetak hasil *topological sort* perencanaan kuliah ke layar.



## 2.6. Prosedur *toposort*

```
def toposort():
    # Melakukan topological sorting secara rekursif

    # KAMUS LOKAL
    # temp : array of string
    # idx : array of int

    # ALGORITMA
    if(len(text) == 0): # Basis
        pass
    else: # Rekursi
        temp = []
        idx = []
        for i in range(len(text)):
            if(len(text[i]) == 1): # Derajat masuk = 0
                temp.append(text[i][0])
                idx.append(i)

        # Menghapus semua mata kuliah dengan derajat masuk = 0
        for i in range(len(idx)-1, -1, -1):
            text.pop(idx[i])

        # Menghapus mata kuliah derajat 0 dari prerequisite
        for i in range(len(temp)):
            deleteClass(temp[i])
        result.append(temp)
        toposort()
```

Prosedur *toposort* digunakan untuk melakukan *topological sorting*. Prosedur *toposort* adalah prosedur rekursif dengan basis *list text* kosong yang artinya semua mata kuliah sudah diurutkan. Selanjutnya untuk rekursinya pertama-tama, akan diperiksa elemen dari *list text* yang memiliki panjang 1. Panjang *list* 1 menandakan elemen *list* memiliki derajat masuk 0. Kemudian, akan disimpan *list* yang memiliki derajat 0 pada *list temp* dan indeksnya pada *list idx*. Selanjutnya, program akan menghapus semua elemen *list text* yang memiliki panjang 1 tersebut. Berikutnya, mata kuliah yang memiliki derajat masuk 0 tersebut dihapus dari elemen *list text* yang memiliki mata kuliah tersebut sebagai *prerequisite*-nya. Terakhir, *list temp* akan dimasukkan ke *list result*. Oleh karena itu, format penyimpanan pada *list result* sebagai berikut di akhir,

```
[['C3'], ['C1'], ['C4'], ['C2'], ['C5']]
```

Dengan elemen *list* pertama menandakan semester 1 dan list kedua menandakan semester 2 dan seterusnya. Kemudian prosedur *toposort* akan dipanggil kembali dengan kondisi *list text* yang sudah berkurang.

## 2.7. Fungsi *isAcyclic*

```
def isCyclic():  
    # Memeriksa apakah graf siklik  
  
    # KAMUS LOKAL  
    # found : boolean  
  
    # ALGORITMA  
    found = False  
    for i in range(len(text)):  
        if len(text[i]) == 1 :  
            found = True  
    return found
```

Fungsi *isAcyclic* digunakan untuk memeriksa apakah masukan pada *list text* merupakan graf siklik. Pengecekan hanya dilakukan dengan memeriksa apakah terdapat derajat masuk 0. Pengecekan didasari asumsi bahwa jika terdapat simpul yang memiliki derajat masuk 0 maka masukan pasti sebuah *directed acyclic graph*. Jika tidak terdapat simpul yang memiliki derajat masuk 0, masukan adalah *cyclic graph*.

## 2.8. Program Utama

```
def main():  
    # Program utama  
  
    # KAMUS  
    # filename : string  
  
    #ALGORITMA  
    start()  
    filename = input("Masukkan file kuliah :")  
    try:  
        file = open(f"../test/{filename}", "r")  
        add(file)  
        if isCyclic():  
            toposort()  
            output()  
        else:  
            print("Format masukan salah, graf siklik!")  
    except:  
        print("No File Found")  
        print("Semoga cepat lulus~~")  
        print("Enter to close..")  
        input()
```

Pada program utama, pertama akan dicetak nama program melalui pemanggilan prosedur *start*. Pengguna akan diminta memasukkan *file* berisi data uji. Data masukan dari *file* akan disimpan ke dalam variabel global *text* melalui prosedur *add*. Kemudian, dilakukan pemeriksaan terhadap *list text* untuk menentukan graf adalah *list* yang asiklik atau tidak. Jika benar, dilakukan pemanggilan prosedur *toposort* yang akan melakukan *topological sorting* terhadap data mata kuliah secara rekursif. Setelah itu, program akan mencetak hasil *topological sorting* ke layar melalui pemanggilan prosedur *output*. Jika bukan asiklik, maka akan ditampilkan pesan kesalahan.

## BAB III

### Screenshot Uji

#### 3.1. Uji 1

##### 3.1.1. *Input*

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

##### 3.1.2. *Output*

```
OTW LULUS!  
SOLUSI 144 SKS ANDA  
Masukkan file kuliah :C.txt  
Semester I :C3  
  
Semester II :C1  
  
Semester III :C4  
  
Semester IV :C2  
  
Semester V :C5  
  
Semoga cepat lulus~~  
Enter to close..
```

#### 3.2. Uji 2

##### 3.2.1. *Input*

```
EB2102.  
EB2103.  
EB2101.  
MA2072.  
EB2206, EB2102.  
EB2207, EB2103.  
EB3101, EB2207.
```

### 3.2.2. Output

```
Command Prompt - python: toposort.py

OTW LULUS!

SOLUSI 144 SKS ANDA
Masukkan file kuliah :EB.txt
Semester I :EB2102, EB2103, EB2101, MA2072

Semester II :EB2206, EB2207

Semester III :EB3101

Semoga cepat lulus~~~
Enter to close..
```

### 3.3. Uji 3

#### 3.3.1. *Input*

```
FI1101.  
MA1101.  
KU1001.  
KU1102.  
KU1011.  
KU1024.  
MA1201, MA1101.  
FI1201, FI1101.  
IF1210, KU1102.  
EL1200, MA1101.  
EL2001, EL1200.  
EL2101, EL1200.  
EL2002, EL1200.  
EL2102, EL1200.  
EL2003, MA1201.  
EL2004, MA1201.  
MA2072, MA1101, MA1201.  
EL2005, EL2001.  
EL2205, EL2001.  
EL2006, MA2072, FI1201.  
EL2007, EL1200, EL2001.  
EL2008, IF1210, MA2072.  
EL2208, IF1210, MA2072.  
MA2074, MA2072.  
EL3009, EL2005.  
EL3109, EL2205.  
EL3010, EL2007.  
EL3011, EL2002, EL2007.  
EL3111, EL2007.  
EL3012, EL2005.  
EL3013, EL2005.  
EL3014, EL2005, EL2008, EL2208, EL2205, EL3010.  
EL3214, EL2208, EL2205, EL3010.  
EL3015, EL2007, EL3010.  
EL3216, EL2007, EL3010.  
EL3016, EL2004, EL2007, EL3010.  
EL3216, EL2004, EL2007, EL3010.  
EL3017, EL2001, EL3009.  
EL3217, EL2101, EL3009.  
EL4018, EL3009.  
EL4092, EL4018.  
EL4090, EL4018.  
EL4091, EL4090.
```

#### 3.3.2. *Output*

```
Command Prompt - python toposort.py
C:\Users\USER\Documents\Git\topological-sort\src>python toposort.py

OTW LULUS!

SOLUSI 144 SKS ANDA
Masukkan file kuliah :EL.txt
Semester I :FI1101, MA1101, KU1001, KU1102, KU1011, KU1024

Semester II :MA1201, FI1201, IF1210, EL1200

Semester III :EL2001, EL2101, EL2002, EL2102, EL2003, EL2004, MA2072

Semester IV :EL2005, EL2205, EL2006, EL2007, EL2008, EL2208, MA2074

Semester V :EL3009, EL3109, EL3010, EL3011, EL3111, EL3012, EL3013

Semester VI :EL3014, EL3214, EL3015, EL3216, EL3016, EL3216, EL3017, EL3217, EL4018

Semester VII :EL4092, EL4090

Semester VIII :EL4091

Semoga cepat lulus~~
Enter to close..
```

### 3.4. Uji 4

#### 3.4.1. Input

```
MA1101.
FI1101.
EL1200, FI1101.
MA1201, MA1101.
EP2091, MA1101, MA1201.
EL2001, EL1200, EP2091.
```

#### 3.4.2. Output

```
Command Prompt - python toposort.py
C:\Users\USER\Documents\Git\topological-sort\src>python toposort.py

OTW LULUS!

SOLUSI 144 SKS ANDA
Masukkan file kuliah :EP.txt
Semester I :MA1101, FI1101

Semester II :EL1200, MA1201

Semester III :EP2091

Semester IV :EL2001

Semoga cepat lulus~~
Enter to close..
```

## 3.5. Uji 5

### 3.5.1. Input

```
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.
```

### 3.5.2. Output

```
Command Prompt - python 13519081.py  
C:\Users\USER\Documents\Git\topological-sort\src>python 13519081.py  
  
OTW LULUS!  
  
SOLUSI 144 SKS ANDA  
Masukkan file kuliah :IF.txt  
Semester I :MA1101, FI1101, KU1001, KU1102, KU1011, KU1024  
  
Semester II :MA1201, FI1201, IF1210, KU1202, KI1002, EL1200  
  
Semester III :IF2121, IF2110, IF2120, IF2124, IF2123, IF2130  
  
Semester IV :IF2210, IF2211, IF2220, IF2230, IF2240, IF2250  
  
Semester V :IF3170, IF3110, IF3130, IF3141, IF3150, IF3140, IF3151  
  
Semester VI :IF3210, IF3270, IF3230, IF3250, IF3260, IF3280  
  
Semester VII :IF4090, IF4091  
  
Semester VIII :IF4092  
  
Semoga cepat lulus~~  
Enter to close..
```

## 3.6. Uji 6

### 3.6.1. Input

```
MA1101.  
FI1101.  
KI1101.  
KU1001.  
KU1102.  
KU1011.  
KU1024.  
MA1201, MA1101.  
FI1201, FI1101.  
TI2105, MA1201, MA1101.  
TI2103, MA1101, MA1201.  
TI2106, KU1102.  
TI2201, TI2105, TI2106.  
TI2005, TI2103.  
MR2003, FI1201.  
TI3103, TI2201.  
TI3001, TI2201.  
TI3104, TI2201.  
TI3002, TI2201.  
TI3005, MR2003.  
TI3201, TI3002.  
TI3202, TI3002.  
TI3203, TI3103.  
TI3007, TI3001, TI3104, TI3005.  
TI3006, TI2106.  
TI4102, TI3001, TI3202.  
TI4103, TI3001, TI3002, TI2005, TI3202.  
TI4160, TI3006.
```

### 3.6.2. Output

```
Command Prompt - python 13519081.py  
  
C:\Users\USER\Documents\Git\topological-sort\src>python 13519081.py  
  
OTW LULUS!  
  
SOLUSI 144 SKS ANDA  
Masukkan file kuliah :TI.txt  
Semester I :MA1101, FI1101, KI1101, KU1001, KU1102, KU1011, KU1024  
  
Semester II :MA1201, FI1201, TI2106  
  
Semester III :TI2105, TI2103, MR2003, TI3006  
  
Semester IV :TI2201, TI2005, TI3005, TI4160  
  
Semester V :TI3103, TI3001, TI3104, TI3002  
  
Semester VI :TI3201, TI3202, TI3203, TI3007  
  
Semester VII :TI4102, TI4103  
  
Semoga cepat lulus~~  
Enter to close..
```



## 3.7. Uji 7

### 3.7.1. Input

```
C1, C2.  
C2, C1.  
C3, C2.
```

### 3.7.2. Output

```
cs Command Prompt - python 13519081.py  
C:\Users\USER\Documents\Git\topological-sort\src>python 13519081.py  
  
OTW LULUS!  
  
SOLUSI 144 SKS ANDA  
Masukkan file kuliah :Cyclic.txt  
Format masukan salah, graf siklik!  
Semoga cepat lulus~~  
Enter to close..
```

## 3.8. Uji 8

### 3.8.1. Input

```
MA1101.  
FI1101.  
KU1001.  
KU1102.  
KU1011.  
KU1024.  
MA1201, MA1101.  
FI1201, FI1101.  
IF1210, KU1102.  
KU1202, KU1011.  
KI1002, MA1101.  
EL1200, FI1101.  
II2130, IF1210.  
II2110, MA1201.  
II2111, MA1201.  
TI3005, KI1002.  
IF2140, IF1210.  
IF2111, IF1210.  
II2250, IF2140.  
II2260, II2130, IF1210.  
II2230, II2130.  
II2220, TI3005.  
II2240, TI3005.  
IF2212, IF2111.  
II3150, II2110, II2230.  
II3160, II2240.  
II3120, TI3005, II2240.  
II3131, IF2212.  
II3121, II2240.  
IF3152, IF2212.  
II3260, IF3152.
```

### 3.8.2. Output

```
Command Prompt - python 13519081.py
C:\Users\USER\Documents\Git\topological-sort\src>python 13519081.py

OTW LULUS!

SOLUSI 144 SKS ANDA
Masukkan file kuliah :STI.txt
Semester I :MA1101, FI1101, KU1001, KU1102, KU1011, KU1024

Semester II :MA1201, FI1201, IF1210, KU1202, KI1002, EL1200

Semester III :II2130, II2110, II2111, TI3005, IF2140, IF2111

Semester IV :II2250, II2260, II2230, II2220, II2240, IF2212

Semester V :II3150, II3160, II3120, II3131, II3121, IF3152

Semester VI :II3260

Semoga cepat lulus~~
Enter to close..
```

## BAB IV

### LAMPIRAN

#### 4.1. Alamat *Repository*

<a href="https://github.com/isabellahandayani/topological-sort">https://github.com/isabellahandayani/topological-sort</a>
---

#### 4.2. *Checklist*

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