

**Laporan Tugas Kecil**  
**IF2211 Strategi Algoritma**  
**Decrease and Conquer**



Rexy Gamaliel Rumahorbo (13519010)

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

**2021**

### 1. Algoritma

Pada persoalan ini setiap matkul memiliki dependensi prerequisite mata kuliah lain, sehingga hubungan dependensi ini dapat digambarkan dengan Directed Acyclic Graph/DAG. Pada *graph* ini, *node* adalah kode mata kuliah, dan *edge* dari *node* i ke *node* j berarti mata kuliah j memiliki *prerequisite* mata kuliah i. DAG digambarkan menggunakan struktur data *dictionary* Python di mana *key* adalah suatu kode mata kuliah dan *value* adalah list kode mata kuliah yang menjadi *prerequisite* mata kuliah tersebut.

Algoritma Topological Sorting yang digunakan adalah sebagai berikut:

- Pada semester ke-i, setiap mata kuliah yang tidak memiliki prerequisite dapat diambil. Tambahkan mata kuliah tersebut ke hasil, untuk semester i.
- Hapus mata kuliah tersebut dari *dictionary*.
- Hapus kemunculan mata kuliah tersebut pada *list of prerequisite* mata kuliah lain
- Selama *dictionary* tidak kosong (masih ada mata kuliah yang belum diambil), ulangi langkah a-c untuk semester ke-i+1

Algoritma Topological Sorting ini menerapkan strategi Decrease and Conquer karena untuk setiap semesternya, persoalan dapat dibagi menjadi 2 upa persoalan: mata kuliah yang dapat diambil pada semester tersebut dan yang tidak. Untuk mata kuliah yang dapat diambil pada suatu semester, dilakukan proses b dan c pada langkah di atas, sehingga menyisakan masalah yang lebih kecil, yakni mata kuliah yang dapat diambil pada semester berikutnya.

Menurut jenisnya, algoritma ini menerapkan jenis strategi Decrease and Conquer yang mengurangi persoalan sebesar suatu variabel, di mana variabel bergantung pada banyaknya mata kuliah yang dapat diambil pada suatu semester.

### 2. Link

[bit.ly/Tucil2-13519010](https://bit.ly/Tucil2-13519010)

### 3. Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi	v	
2. Program berhasil running	v	
3. Program dapat membaca berkas input dan menghasilkan output	v	
4. Luaran sudah benar untuk semua kasus input	v	

### 4. Source Code

```
# ##### Tugas Kecil 2: Penyusunan Rencana Kuliah dengan Topological Sorting #####
# NIM      : 13519010
# Nama     : Remy Gamaliel Rumahorbo
```

```

# Kelas      : K01

# Untuk mengimplementasikan DAG pada persoalan ini, digunakan dictionary in_kode_
matkul:
#   key berupa suatu kode mata kuliah, dan value berupa list of kode mata kuliah
yang menjadi prerequisite-nya
#   Dengan kata lain, in_kode_matkul merupakan "adjacency list" untuk edge "hubun
gan prerequisite" yang *masuk* ke setiap node "kode mata kuliah"

def parseTxt(in_kode_matkul):
    # Melakukan parsing dari file .txt dengan format kode mata kuliah diikuti
    # prerequisite-nya ke dalam bentuk dictionary dengan key adalah kode mata kuliah dan
    # value berupa list of kode mata kuliah lainnya yang menjadi prerequisite bagi
    mata kuliah tersebut.
    with open("input.txt") as file:
        nth_row = 0
        for row in file:
            i = 0
            parent_kode_matkul = True
            current_parent_kode = ''
            temp_kode_list = []
            while (True):
                # akuisisi baris kode mata kuliah dan prerequisite-nya
                current_kode = ''
                while (row[i] != ','):
                    if (row[i] == '.'): # end of line
                        break
                    current_kode += row[i]
                    i += 1
                # selesai mengakuisisi 1 kode mata kuliah
                if (parent_kode_matkul): # jika kode matkul yang dibaca adalah kode pert
                    ama di baris
                        current_parent_kode = current_kode
                        in_kode_matkul.update({current_parent_kode: []})
                        parent_kode_matkul = False
                else:
                    temp_kode_list.append(current_kode)
                if (row[i] == '.'): # end of line
                    break
                i += 1
            in_kode_matkul.update({current_parent_kode: temp_kode_list})
            nth_row += 1
        file.close()

```

```

def deleteDependency(in_kode_matkul, list_deleted_kode_matkul):
    dictionary = in_kode_matkul.copy()
    # menghapus kode_matkul dari dictionary
    for kode_matkul in list_deleted_kode_matkul:
        if kode_matkul in dictionary:
            dictionary.pop(kode_matkul)

    # menghapus kode_matkul yang menjadi prerequisite matkul lain
    for kode_matkul in list_deleted_kode_matkul:
        list_deleted_prerequisite = []
        for key in dictionary:
            temp_list = dictionary.get(key)
            if kode_matkul in temp_list:
                temp_list.remove(kode_matkul)
                dictionary.update({key: temp_list})

    return dictionary

def TopologicalSort(kode_matkul_in, result, current_semester):
    in_kode_matkul = kode_matkul_in.copy()

    if in_kode_matkul: # masih ada mata kuliah yang belum dijadwalkan pada result
        result.update({current_semester: []})
        deleted_matkul = []
        for key in in_kode_matkul:
            if len(in_kode_matkul.get(key)) == 0:
                # menambahkan kode mata kuliah ke hasil
                temp_list = result.get(current_semester)
                temp_list.append(key)
                result.update({current_semester: temp_list})
                deleted_matkul.append(key)
        in_kode_matkul = deleteDependency(in_kode_matkul, deleted_matkul)
        result = TopologicalSort(in_kode_matkul, result, current_semester+1)

    # semua mata kuliah telah di-assign pada semester tertentu
    return result

def printHasil(result):
    print("Solusi urutan pengambilan mata kuliah:")
    for key in result:
        print("Semester", key, ":", result.get(key))

# Menampilkan hasil parsing file .txt
in_kode_matkul = {}

```

```

parseTxt(in_kode_matkul)
print(in_kode_matkul)

print("# of prerequisites setiap matkul: ")
for key in in_kode_matkul:
    print(key, end=': ')
    print(len(in_kode_matkul.get(key)))

result = {}
result = TopologicalSort(in_kode_matkul, {}, 1)
print("Result: ")

printHasil(result)

```

## 5. Contoh kasus

### a. input.txt:

```

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

```

Hasil:

```

Result:
Solusi urutan pengambilan mata kuliah:
Semester 1 : ['C3']
Semester 2 : ['C1']
Semester 3 : ['C4']
Semester 4 : ['C2']
Semester 5 : ['C5']

```

### b. input.txt:

```

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

```

Hasil:

```

Result:
Solusi urutan pengambilan mata kuliah:
Semester 1 : ['C1', 'C3']
Semester 2 : ['C4']
Semester 3 : ['C2']
Semester 4 : ['C5']

```

### c. input.txt:

C2.  
C3.  
C4.  
C5.

Hasil:

```
Result:  
Solusi urutan pengambilan mata kuliah:  
Semester 1 : ['C2', 'C3', 'C4', 'C5']
```

- d. input.txt:  
IF2211,IF1201,MA1101,MA1201.  
MA1101.  
MA1201.  
IF1201,MA1101.  
IF3120,IF2211.

Hasil:

```
Result:  
Solusi urutan pengambilan mata kuliah:  
Semester 1 : ['MA1101', 'MA1201']  
Semester 2 : ['IF1201']  
Semester 3 : ['IF2211']  
Semester 4 : ['IF3120']
```

- e. input.txt  
IF2211,IF1201,MA1101,MA1201.  
MA1101.  
MA1201.  
IF1201.  
IF3120,IF2211.

Hasil:

```
Result:  
Solusi urutan pengambilan mata kuliah:  
Semester 1 : ['MA1101', 'MA1201', 'IF1201']  
Semester 2 : ['IF2211']  
Semester 3 : ['IF3120']
```

- f. input.txt:  
A1.  
A2,A1.  
B1,A1,A2.  
B2,B1.  
C1,B2.

Hasil:

```
Result:
Solusi urutan pengambilan mata kuliah:
Semester 1 : ['A1']
Semester 2 : ['A2']
Semester 3 : ['B1']
Semester 4 : ['B2']
Semester 5 : ['C1']
```

g. input.txt:

```
A1.
A2,A1.
B1,A1.
B2,B1.
C1,B2.|
```

Hasil:

```
Result:
Solusi urutan pengambilan mata kuliah:
Semester 1 : ['A1']
Semester 2 : ['A2', 'B1']
Semester 3 : ['B2']
Semester 4 : ['C1']
```

h. input.txt

```
A1.
A2,A1.
B1,A1.
B2,B1.
C1,B1,B2.|
```

Hasil:

```
Result:
Solusi urutan pengambilan mata kuliah:
Semester 1 : ['A1']
Semester 2 : ['A2', 'B1']
Semester 3 : ['B2']
Semester 4 : ['C1']
```