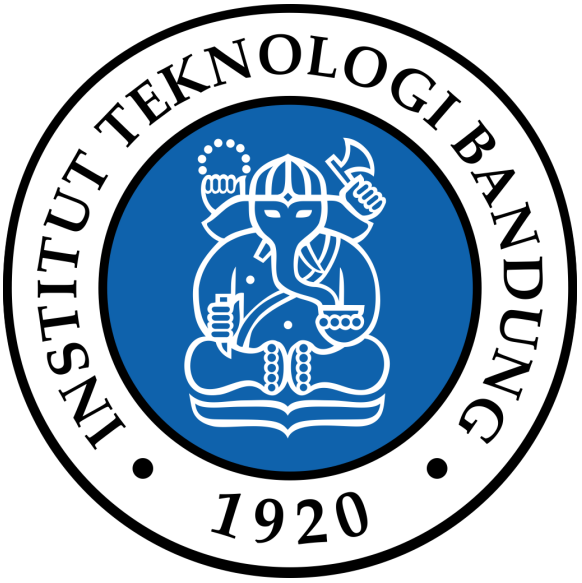
Penyusunan Rencana Kuliah dengan Topological Sort

LAPORAN TUGAS KECIL 2
Diajukan sebagai salah satu Tugas Kecil 2
IF2211 Strategi Algoritma Semester II tahun 2020/2021

Disusun Oleh:
Muhammad Fahkry Malta (13519032)



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

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.		✓

Bab 1. Algoritma Topological Sort

Decrease and conquer adalah metode desain algoritma dengan mereduksi persoalan menjadi beberapa sub persoalan yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja. Topological sorting atau topological ordering adalah algoritma untuk melakukan pengurutan secara linier terhadap semua simpul dari sebuah graf berarah, di mana setiap sisi uv, simpul u berada sebelum v pada hasil pengurutan. Setiap simpul dari graf tersebut dapat saja dimisalkan sebagai suatu pekerjaan yang harus dilakukan. Sisi yang ada menandakan urutan pengerjaan, bahwa suatu pekerjaan harus dilakukan terlebih dahulu sebelum pekerjaan yang lain. Dalam hal ini, topological sorting atau topological ordering adalah keterurutan yang sahih dari penjadwalan pekerjaan ini. Topological sorting dapat dilakukan jika dan hanya jika graf yang bersangkutan tidak mempunyai siklus berarah, dengan kata lain graf itu adalah Directed Acyclic Graph (DAG) atau graf berarah yang tidak mempunyai siklus. Setiap DAG pasti mempunyai minimal satu keterurutan secara topologi dan algoritma topological sorting digunakan untuk menemukan keterurutan topologi dari DAG tersebut secara linier. Algoritma Topological Sort sangat erat kaitannya dengan Decrease and conquer, hal ini karena Topological Sort menggunakan metode Decrease and Conquer dalam implementasinya seperti backtracking dan lain-lain.

```python
# Kode program untuk menampilkan ke layar (print) topological sorting dari DAG

# Class untuk merepresentasikan graf

class Graf:
    def __init__(self, simpul):
        self.graf = defaultdict(list)
        self.S = simpul

    # fungsi untuk memasukkan sisi-sisi kedalam graf berdasarkan simpulnya
    def addSisi(self, u, v):
        self.graf[u].append(v)

    # Fungsi rekursif Topological Sort
    def topologicalSortUtil(self, v, dikunjungi, stack):

        dikunjungi[v] = True

        for i in self.graf[v]:
            if dikunjungi[i] == False:
                self.topologicalSortUtil(i, dikunjungi, stack)

        stack.append(v)
```

```python
    # Fungsi untuk melakukan Topological Sort
    # topologicalSortUtil()
    def topologicalSort(self):

        dikunjungi = [False]*self.S
        stack = []

        for i in range(self.S):
            if dikunjungi[i] == False:
                self.topologicalSortUtil(i, dikunjungi, stack)

        return(stack[::-1])
```

Bab 2. Source program dalam bahasa pemrograman yang dipilih

```
from collections import defaultdict

# Membaca file teks sebagai file masukkan (input)
berkas = input("Masukkan berkas: ")
baca = open(berkas, "r")
arr_baca = baca.readlines()
arr_matkul = ["*" for i in range(len(arr_baca))]

# Menghilangkan char "." dan "\n" pada array arr_baca
```

```
for i in range(len(arr_baca)-1):
        arr_matkul[i] = arr_baca[i].replace(".\n", "")
arr_matkul[len(arr_baca)-1] = arr_baca[len(arr_baca)-1].replace(".", "")

# Memisahkan setiap mata kuliah yang terdapat dalam satu baris (satu indeks list) dengan tanda koma
matriks_matkul = ["*" for i in range(len(arr_matkul))]
for i in range(len(arr_matkul)):
        matriks_matkul[i] = arr_matkul[i].split(", ")

# Kode program untuk menampilkan ke layar (print) topological sorting dari DAG

# Class untuk merepresentasikan graf

class Graf:
        def __init__(self, simpul):
        self.graf = defaultdict(list)
        self.S = simpul

        # fungsi untuk memasukkan sisi-sisi kedalam graf berdasarkan simpulnya
        def addSisi(self, u, v):
        self.graf[u].append(v)

        # Fungsi rekursif Topological Sort
        def topologicalSortUtil(self, v, dikunjungi, stack):

        dikunjungi[v] = True

        for i in self.graf[v]:
        if dikunjungi[i] == False:
                self.topologicalSortUtil(i, dikunjungi, stack)

        stack.append(v)

        # Fungsi untuk melakukan Topological Sort
        # topologicalSortUtil()
        def topologicalSort(self):

        dikunjungi = [False]*self.S
        stack = []

        for i in range(self.S):
        if dikunjungi[i] == False:
                self.topologicalSortUtil(i, dikunjungi, stack)

        return(stack[::-1])
```

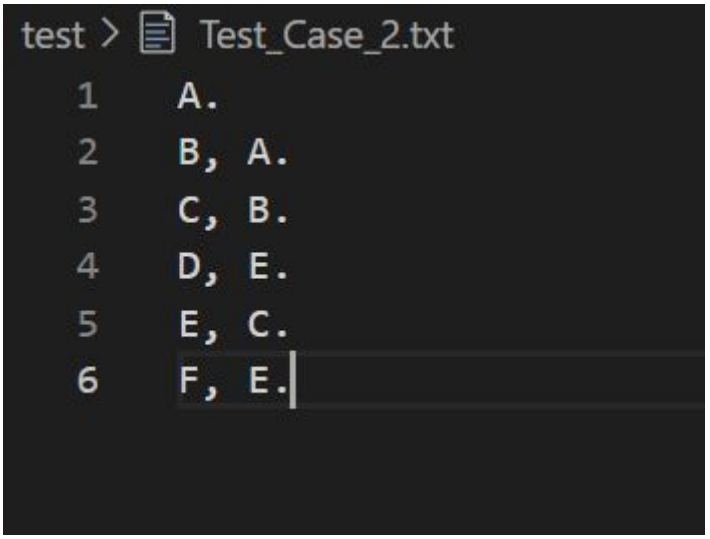
```
# Array matkul_unik untuk menyimpan mata kuliah dari Array matriks_matkul tetapi secara unik untuk masing-masing matkul
matkul_unik = []
for i in matriks_matkul:
        for j in i:
        if j not in matkul_unik:
        matkul_unik.append(j)

# Array nil_matkul untuk menyimpan integer sebagai nilai unik untuk masing-masing matkul
nil_matkul = [0 for i in range(len(matkul_unik))]
for i in range(len(matkul_unik)):
        nil_matkul[i] = i

# Matriks mat_nil_matkul sebagai matriks yang akan menyimpan matkul pada matriks matriks_matkul dengan nilai integer unik masing-masing
mat_nil_matkul = matriks_matkul
for i in range(len(matriks_matkul)):
        for j in range(len(matriks_matkul[i])):
        for k in range(len(matkul_unik)):
        if matkul_unik[k] == mat_nil_matkul[i][j] :
                mat_nil_matkul[i][j] = nil_matkul[k]

# Implementasi addSisi untuk menyimpan hubungan antara matkul dan prerequisite nya
graf_matkul = Graf(len(matkul_unik))
for i in range(len(mat_nil_matkul)):
        if len(mat_nil_matkul[i]) > 1:
        for j in range(1,len(mat_nil_matkul[i])):
        graf_matkul.addSisi(mat_nil_matkul[i][j], mat_nil_matkul[i][0])

# Array nil_hasil sebagai array yang menyimpan hasil topological sorting akan tetapi masih dalam bentuk integer
nil_hasil = []
nil_hasil = graf_matkul.topologicalSort()

# Array hasil sebagai array yang menyimpan hasil topological searching dan sudah dalam bentuk susunan matkul
hasil = ["*" for i in range(len(nil_hasil))]
for i in range (len(nil_matkul)):
        for j in range (len(nil_hasil)):
        if nil_hasil[j] == nil_matkul[i]:
        hasil[j] = matkul_unik[i]

for i in range(len(hasil)):
        print("Semester ", i+1, "   : ", hasil[i])
```

3. Tangkapan layer dari input dan output minimal 8 contoh

Test Case 1


```
Masukkan berkas: Test_Case_1.txt
Semester  1    :  A
Semester  2    :  G
Semester  3    :  C
Semester  4    :  H
Semester  5    :  D
Semester  6    :  E
Semester  7    :  F
Semester  8    :  B
PS D:\Semester 4 IP 4!!!\Strategi Algoritma (Stima)\Tubes & Tucil\Tucil 2 - Penyusunan Rencana Kuliah dengan Topological Sort\test>
```

Test Case 2



```
Masukkan berkas: Test_Case_2.txt
Semester  1    :  A
Semester  2    :  B
Semester  3    :  C
Semester  4    :  E
Semester  5    :  F
Semester  6    :  D
```

Test Case 3

test > Test_Case_3.txt

```
1   A.
2   B, A.
3   C, B.
4   D, B, G.
5   E, C, B, D.
6   F, E.
7   G.
```

```
Masukkan berkas: Test_Case_3.txt
Semester  1    :  G
Semester  2    :  A
Semester  3    :  B
Semester  4    :  D
Semester  5    :  C
Semester  6    :  E
Semester  7    :  F
```

Test Case 4

test > Test_Case_4.txt

```
1   A.
2   B, A.
3   C, A.
4   D, A, B, C.
5   E, A, C, D.
```

```
Masukkan berkas: Test_Case_4.txt
Semester  1    :  A
Semester  2    :  C
Semester  3    :  B
Semester  4    :  D
Semester  5    :  E
```

Test Case 5

test > Test_Case_5.txt

```
1   A.
2   B, A.
3   C, A.
4   D, C.
5   E, B, C, D.
```

```
Masukkan berkas: Test_Case_5.txt
Semester  1    :  A
Semester  2    :  C
Semester  3    :  D
Semester  4    :  B
Semester  5    :  E
```

Test Case 6

test > Test_Case_6.txt

```
1   A.
2   B, A.
3   C, B.
4   D, B.
5   E, C, D.
6   F, C.
7   G, F.
```

```
Semester  1    :  A
Semester  2    :  B
Semester  3    :  D
Semester  4    :  C
Semester  5    :  F
Semester  6    :  G
Semester  7    :  E
```

Test Case 7

test > Test_Case_7.txt

```
1   A.
2   B, A.
3   C, A.
4   D, B, C.
5   E, B.
6   F, C.
7   G, D.
8   H, D.
```

```
Masukkan berkas: Test_Case_7.txt
Semester  2    :  C
Semester  3    :  F
Semester  4    :  B
Semester  5    :  E
Semester  6    :  D
Semester  7    :  H
Semester  8    :  G
```

Test Case 8

test > Test_Case_8.txt

```
1   A.
2   B.
3   C.
4   D, A, B, C.
5   E, D.
6   F, D.
7   G, D.
8   H, E, F, G.
```

```
Masukkan berkas: Test_Case_8.txt
Semester  1    :  C
Semester  2    :  B
Semester  3    :  A
Semester  4    :  D
Semester  5    :  G
Semester  6    :  F
Semester  7    :  E
Semester  8    :  H
```

4. Alamat drive yang berisi kode program

https://github.com/fahkrymalta21/Tucil-2-Stima