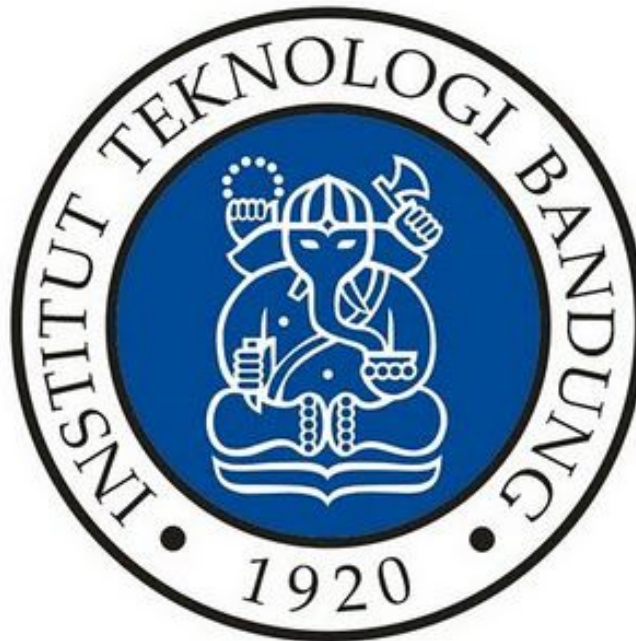


LAPORAN TUGAS KECIL 2

IF2211 - STRATEGI ALGORITMA

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



**Oleh :
Jesica 13519011**

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

Deskripsi Tugas

Membuat program yang menerima daftar mata kuliah beserta *prerequisite* dari mata kuliah tersebut. Program akan menampilkan rencana mata kuliah yang telah diurutkan dengan *topological sort*.

```
<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

Algoritma *Decrease and Conquer* & *Topological Sort*

Algoritma *decrease and conquer* adalah metode penyelesaian sebuah persoalan dengan mereduksi persoalan tersebut menjadi upa-persoalan yang lebih kecil sehingga persoalan yang diproses selanjutnya menjadi lebih kecil. *Topological sort* adalah metode pengurutan sebuah graf yang terdiri atas beberapa simpul sedemikian rupa sehingga setiap terdapat *edge* u, v (memiliki arah), simpul u muncul sebelum simpul v dalam urutannya. Metode ini tidak dapat dilakukan pada graf yang bukan DAG (*Directed Acyclic Graph*).

Topological sort sendiri merupakan salah satu pendekatan dari algoritma *decrease and conquer*. Dengan algoritma *topological sort*, akan dicari simpul yang memiliki derajat masuk 0 kemudian simpul tersebut dihapus dari graf. Begitu pun dengan simpul-simpul yang memiliki sisi berarah dari simpul yang dihapus ini, masing-masing derajat masuk simpul tersebut akan dikurangi 1. Hal ini dilakukan secara terus menerus sampai tidak ada lagi simpul yang belum diproses. Dengan dihapusnya simpul yang tidak memiliki derajat masuk, maka ini akan mereduksi persoalan, dalam hal ini jumlah simpul dalam graf. Sehingga jumlah simpul yang diproses selanjutnya akan lebih kecil.

Implementasi *topological sort* dalam penyusunan rencana kuliah

1. Program membaca file yang berisi daftar mata kuliah yang ingin diambil. Setiap mata kuliah direpresentasikan menjadi sebuah simpul graf
2. Cari simpul yang tidak memiliki derajat masuk atau dalam hal ini, mata kuliah yang tidak memiliki *prerequisite*
3. Semua simpul yang tidak memiliki derajat masuk dimasukkan ke dalam sebuah array dimana array tersebut akan berisi mata kuliah yang dapat diambil di satu semester. Derajat masuk simpul diubah menjadi -99, menandai bahwa simpul telah diproses. Hapus juga simpul dari graf
4. Kurangi semua derajat masuk simpul yang memiliki sisi berarah dari simpul yang dihapus sebelumnya dengan 1.
5. Ulangi langkah 2 sampai 4 hingga tidak ada lagi simpul yang tersisa di dalam graf

Source Code Program graph_13519011.py

```
#graph5.py
import os
acyclic = False

class Graph:

    # Constructor
    # edges itu list of tuple yang bersisian
    # contoh [(2,1),(3,1)] berarti 1 ngarah ke 2, 1 ngarah ke 3
    def __init__(self, edges, N):

        # list dari node yang bertetanggaan
        self.adjList = [[] for _ in range(N)]

        # inisialisasi derajat masuk dengan 0
        self.indegree = [0] * N

        # mengkonversi edges menjadi array daftar dari node yang ditunjuk oleh suatu node
        for (src,dest) in edges:
            self.adjList[dest].append(src)
            self.indegree[src] = self.indegree[src] + 1

# untuk mencetak angka romawi
# sumber : https://www.w3resource.com/python-exercises/class-exercises/python-class-exercise-1.php
class py_solution:
    def int_to_Roman(self, num):
        val = [
            1000, 900, 500, 400,
            100, 90, 50, 40,
            10, 9, 5, 4,
            1
        ]
        syb = [
            "M", "CM", "D", "CD",
            "C", "XC", "L", "XL",
            "X", "IX", "V", "IV",
            "I"
        ]
        roman_num = ''
        i = 0
        while num > 0:
            for _ in range(num // val[i]):
                roman_num += syb[i]
                num -= val[i]
            i += 1
        return roman_num
```

```

def noIndegree(graph):
    no = []
    N = len(graph.adjList)
    for i in range(1,N):
        if graph.indegree[i]==0:
            no.append(i)
            graph.indegree[i] = -99
    return no

# mengembalikan nested array yang berisi
# [[key matkul smt 1],[key matkul smt 2],[key matkul smt ..]]
# topological sort terjadi di sini
def pleaseWork(graph, key): # versi tidak rekursif
    global acyclic
    graphnya = graph
    arrkey = key
    level = []
    perlevel = []

    # selama arrkey tidak kosong
    while len(arrkey)>1 and not acyclic:
        # perlevel diisi dengan array of key yang simpulnya berderajat masuk 0
        perlevel = noIndegree(graphnya)

        # jika tidak ditemukan yang derajatnya 0 maka dia acyclic
        # proses berhenti
        if len(perlevel) < 1:
            acyclic = True
            level = []
            break

        # jika DAG maka dilanjutkan
        else:
            # kurangi semua derajat adjacent dari lv dengan 1
            for lv in perlevel:
                for u in graphnya.adjList[lv]:
                    graphnya.indegree[u] = graphnya.indegree[u] - 1

            # menghapus nilai lv dari arrkey
            arrkey.remove(lv)

            # memasukan array perlevel ke level (nested array)
            level.append(perlevel)
    return(level)

```

```

def pleasework2(graphnya, key, level, perlevel): # versi rekursif
    global acyclic
    ''' BASIS '''
    if(len(key)<=1 or acyclic): # ketika tidak acyclic atau graf sudah kosong
        if(acyclic):
            level = []
            return level

    ''' REKURENS '''
    if(len(key)>1):
        # perlevel diisi dengan array of key yang simpulnya berderajat masuk 0

        #print("degree mula2:",format(graphnya.indegree))
        perlevel = noIndegree(graphnya)
        #print("perlevel :",format(perlevel))

        if(len(perlevel)<1):
            acyclic = True
        # kurangi derajat simpul yang bersisian
        for lv in perlevel:
            for u in graphnya.adjList[lv]:
                graphnya.indegree[u] = graphnya.indegree[u] - 1

        # menghapus nilai lv dari arrkey
        key.remove(lv)

        #print("key :",format(key))
        # memasukan array perlevel ke level (nested array)
        level = level + [perlevel]
        #print("level :",format(level))

        return pleasework2(graphnya, key, level, perlevel)

# membaca file txt x dan membuat array of matkul y
def readWords(x,y):
    with open(x,'r') as f:
        for line in f:
            # hapus spasi disekitar koma
            cleanedline = line.replace(' ','')
            cleanedline2 = cleanedline.replace(',','')
            # hapus titik
            cleanedline3 = cleanedline2.replace('.', '')
            y += cleanedline3.rstrip().split(',')

```

```

## MEMPEROLEH KEY DARI DICT MATKUL
def getKey(val, my_dict):
    for key, value in my_dict.items():
        if (val == value):
            return key
            break

##### CREATE LIST OF EDGES #####
def readToMakeEdges(filename, edge, map):
    filepath = filename
    with open(filepath) as fp:

        # inisialisasi
        line = fp.readline()
        countcomma = 0
        pertama = 0
        kedua = 0
        while line: #selama di suatu baris

            # hapus spasi disekitar koma
            cleanedline = line.replace(' ', ',')
            cleanedline2 = cleanedline.replace(', ', ',')
            # hapus titik
            cleanedline3 = cleanedline2.replace('.', '')
            # menghitung jumlah koma
            countcomma = cleanedline.count(",")
            if(countcomma>0):
                # 1 koma berarti ada 2 matkul
                # 2 koma berarti ada 3 matkul
                # n matkul = n koma + 1

                # buat ambil matkul pertama dan kedua = [0] & [1]
                # buat ambil matkul pertama dan ketiga = [0] & [2]
                for i in range(1,countcomma+1):
                    pertama = getKey(cleanedline3.split(",")[0].strip(),map)
                    kedua = getKey(cleanedline3.split(",")[i].strip(),map)
                    edge.append((pertama,kedua))
            line = fp.readline()

def printResult(graphnya, key, map): # print result untuk yang versi bukan rekursif
    hasil = pleasework(graphnya,key)
    jumlahsemester = len(hasil)

```



```

# jika array tidak kosong
if len(hasil)!=0:
    for i in range(jumlahsemester):
        print('    Semester {:<4s} : '.format(py_solution().int_to_Roman(i+1)),end="")
        for j in range(len(hasil[i])-1):
            print('{','.format(map[hasil[i]][j])),end="")
            print('{'}.format(map[hasil[i]][len(hasil[i])-1]))
else: # array kosong (acyclic)
    print("    Tidak ada, graf yang terbentuk bukan DAG")

def printResult2(graphnya, key, map): # print result untuk yang versi rekursif
    level = []
    perlevel = []
    hasil = pleaseWork2(graphnya,key,level,perlevel)
    jumlahsemester = len(hasil)

    # jika array tidak kosong
    if len(hasil)!=0:
        for i in range(jumlahsemester):
            print('    Semester {:<4s} : '.format(py_solution().int_to_Roman(i+1)),end="")
            for j in range(len(hasil[i])-1):
                print('{','.format(map[hasil[i]][j])),end="")
                print('{'}.format(map[hasil[i]][len(hasil[i])-1]))
    else: # array kosong (acyclic)
        print("    Tidak ada, graf yang terbentuk bukan DAG")

# print ascii
def printAC3():
    width = os.get_terminal_size().columns
    s="""
          ,---,  _.._
        ,---,  /   /
       '  ' \   / / \ / .. /
      / ;   '  | :   : \ \ \ .'-
     : :       \ . | ;. / _ \ \ \ :
     : |  ^ \ \ . ; /--'   \ : |
     | : ' ;. : ; | ;       / / /
     | | ;/ \ \ \ : |       \ \ \
     ' : | \ \ ,'. | ' _ _ / : |
     | | ' '---' : ; :.' / ^ / :
     | : :       ' | '/ :/ ,/ ',-
     | | , '    | :   / \ '' \
     '---'      \ \ \ ' \ \ \
              _.._  ,---,

"""

    print('\n'.join(l.center(width-1) for l in s.splitlines()))
    print("a n t i   c h a o s".center(width-1))
    print("c h a o s   c l u b".center(width-1))

```

main_13519011.py

```
# main.py
import graph as g
import os

# directory
path_parent = os.path.dirname(os.getcwd())
os.chdir(path_parent)
os.chdir('test')

# inisialisasi boolean
acyclic = False
g.printAC3()
exit = False

# input nama file
namafile = input("\n\n    Masukkan nama file : ")

## arr adalah array yang berisi nama matkul dari file txt (masih ada duplikasi)
arr = []
g.readWords(namafile, arr)

## arrmatkul berisi semua nama matkul (sudah tidak ada duplikasi)
arrmatkul = [""] #indeks 0 diisi dengan ""
for c in arr:
    if c not in arrmatkul:
        arrmatkul.append(c)

jum = len(arrmatkul)
## MEMBUAT ARRAY DARI [0...jum]
key = [0 for i in range(jum)]
for i in range(jum):
    key[i] = i

## MEMBUAT DICTIONARY
# CONTOH : {1 : "Kalkulus", 2 : "Basdat"}
map = {} #dictionary
for nomor, matkul in zip(key, arrmatkul):
    map[nomor] = matkul
#print(map)

# inisialisasi
edges = []
g.readToMakeEdges(namafile, edges, map)

# generate graph
graphnya = g.Graph(edges, len(arrmatkul))
```

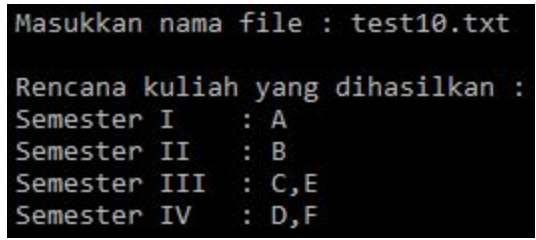


```
# mencetak hasil
print("\n    Rencana kuliah yang dihasilkan : ")
#g.printResult(graphnya,key, map)
level = []
perlevel = []
#g.printResult(graphnya, key, map) -- tidak rekursif
g.printResult2(graphnya, key, map) # menggunakan versi rekursif
```

Input Output

Input (dalam bentuk file txt)	Output
C1, C3. C2, C1, C4. C3. C4, C1, C3. C5, C2, C4.	<pre>Masukkan nama file : test1.txt Rencana kuliah yang dihasilkan : Semester I : C3 Semester II : C1 Semester III : C4 Semester IV : C2 Semester V : C5</pre>
A. B, A. C, B. D, B, G. E, C, B, D. F, E. G.	<pre>Masukkan nama file : test2.txt Rencana kuliah yang dihasilkan : Semester I : A,G Semester II : B Semester III : C,D Semester IV : E Semester V : F</pre>
C1. C2, C1. C3, C1. C4, C2. C5, C2, C6. C6, C3. C7, C4, C5, C6.	<pre>Masukkan nama file : test3.txt Rencana kuliah yang dihasilkan : Semester I : C1 Semester II : C2,C3 Semester III : C4,C6 Semester IV : C5 Semester V : C7</pre>
C2, C1. C3, C1. C4, C2. C5, C2, C3.	<pre>Masukkan nama file : test4.txt Rencana kuliah yang dihasilkan : Semester I : C1 Semester II : C2,C3 Semester III : C4,C5</pre>

C0, C7, C3. C1, C5, C7. C2, C1. C3, C7. C4, C3, C1. C5, C7. C6, C0, C1. C7.	<pre>Masukkan nama file : test5.txt Rencana kuliah yang dihasilkan : Semester I : C7 Semester II : C3,C5 Semester III : C0,C1 Semester IV : C2,C4,C6</pre>
B1. B2, B1. B3, B1, B2. B4, B2, B3. B5, B4, B3. B6, B5. B7, B1, B3. B8, B7.	<pre>Masukkan nama file : test6.txt Rencana kuliah yang dihasilkan : Semester I : B1 Semester II : B2 Semester III : B3 Semester IV : B4,B7 Semester V : B5,B8 Semester VI : B6</pre>
Algoritma Struktur Data. Basis Data, Algoritma Struktur Data. Strategi Algoritma, Algoritma Struktur Data. Matematika Diskrit, Strategi Algoritma. RPL, Strategi Algoritma, Basis Data. TBFO, RPL, Matematika Diskrit.	<pre>Masukkan nama file : test7.txt Rencana kuliah yang dihasilkan : Semester I : Algoritma Struktur Data Semester II : Basis Data,Strategi Algoritma Semester III : Matematika Diskrit,RPL Semester IV : TBFO</pre>
A1. A2. A3. A4, A1, A2, A3. A5, A4. A6, A4. A7, A4. A8, A5, A6, A7.	<pre>Masukkan nama file : test8.txt Rencana kuliah yang dihasilkan : Semester I : A1,A2,A3 Semester II : A4 Semester III : A5,A6,A7 Semester IV : A8</pre>
C0, C5, C4. C1, C4, C3. C2, C5. C3, C2.	<pre>Masukkan nama file : test9.txt Rencana kuliah yang dihasilkan : Semester I : C5,C4 Semester II : C0,C2 Semester III : C3 Semester IV : C1</pre>

A. B, A. C, B. D, C. E, B. F, E.	 <pre> Masukkan nama file : test10.txt Rencana kuliah yang dihasilkan : Semester I : A Semester II : B Semester III : C,E Semester IV : D,F </pre>
---	---

Tabel 1 Screenshot input output

Link Source Code

[jestsee/antichaoschaosclub \(github.com\)](https://github.com/jestsee/antichaoschaosclub)

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.	✓	