LAPORAN TUGAS KECIL 2

IF2211 – STRATEGI ALGORITMA

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

Oleh :

Raihan Astrada Fathurrahman 13519113

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

1. Algoritma *Topological Sort*

Algoritma *Topological Sort* merupakan suatu algoritma pengurutan simpul (*node*) dalam graf berarah (*directed graph*) yang mana untuk setiap sisi berarah dari A ke B, simpul A akan diurutkan lebih dahulu jika dibandingkan dengan B. Pertama, program akan membaca input file yang telah dimasukkan dan memetakannya ke dalam representasi graf. Pada program yang saya buat representasi graf terdiri atas dua bagian yaitu simpul (kode suatu mata kuliah) dan busur berarah yang menuju ke simpul tersebut (mata kuliah yang perlu diambil terlebih dahulu, selanjutnya disebut sebagai *list prec node*). Setelah graf-graf tersebut disimpan, maka algoritma *Topological Sort* akan mencari simpul pada pada *Directed Acylic Graph* yang memiliki derajat masuk (*in-degree*) bernilai 0. Karena terdapat kemungkinan bahwa terdapat banyak simpul yang memiliki nilai 0, maka seluruh simpul yang memiliki *in-degree* bernilai 0 tersebut akan disimpan dalam suatu *list*. Selanjutnya, program akan mengurangi isi *list prec node* dari suatu simpul jika simpul tersebut memiliki mata kuliah dengan *in-degree* bernilai 0 sebagai prerequisite. Setelah mengurangi isi *list prec node*, program akan menghapus simpul yang memiliki *in-degree* 0 dan menampilkan kode matkul pada suatu layar sebagai matkul yang diambil pada suatu semester. Proses ini terjadi untuk setiap elemen dari simpul yang memiliki *in-degree* bernilai 0. Jika, seluruh elemen *list in-degree* bernilai 0 sudah selesai di proses tetapi masih terdapat graf yang tersisa maka program akan kembali lagi ke pengecekan awal untuk mencari simpul-simpul yang memiliki *in-degree* bernilai 0 dan mengulang lagi proses-proses berikutnya. Pengulangan ini akan terjadi secara menerus hingga seluruh graf sudah dihapuskan dan tidak terdapat lagi graf yang tersisa untuk diproses.

Kaitannya dengan pendekatan *Decrease and Conquer* yaitu, pada algoritma *topological sort*, seluruh proses didalamnya mengurangi permasalahan secara terus menerus menjadi permasalahan yang lebih kecil dari awalnya. Dalam kasus ini, hal yang dikurangi adalah simpul mata kuliahnya. Pada awalnya, dalam permasalahan kita tidak dapat mengambil suatu mata kuliah yang memiliki mata kuliah prerequisite. Namun, dengan cara kita mengurangi mata kuliah yang tidak memiliki prerequisite secara terus menerus, pada akhirnya tidak akan terdapat mata kuliah yang memiliki prerequisite lagi dan kita dapat menyelesaikan permasalahan menyusun rencana kuliah tersebut.

2. Source Code Program

```
# Modul yang digunakan
import os

def get_files(filename):
    # Membaca file dan mengembalikan list graph
    cur_path = os.path.dirname(__file__)
    fpath = os.path.join(cur_path, '..\\test\\'+filename)
    graph_list = []
    try:
        f = open(fpath, "r")
        node = "" # Kode matkul
        prerequisites = "" # Kode matkul prerequisite
        preq_node = [] # List kode matkul prerequisite
        char = f.read(1)
```

```
        while (char): # Selama bukan akhir file
            # Ambil kode matkul
            while(char != "," and char != "."):
                node += char
                char = f.read(1)
                # {EOP : char = "," or char = "." selesai mengambil kode matkul}

            if (char != "."):
                # Jika terdapat matkul prerequisites, char = ","
                char = f.read(1)
                # Ambil kode matkul prerequisite
                while(char != "."):
                    if (char == ","):
                        preq_node.append(prerequisites)
                        prerequisites = ""
                        char = f.read(1)
                    else:
                        prerequisites += char
                        char = f.read(1)
                    # {EOP : selesai mengambil kode matkul prerequisites}
                preq_node.append(prerequisites)
                prerequisites = ""

            # Masukkan ke list graph
            graph_list.append([node,preq_node])
            node = ""
            preq_node = []

            # Ke next line
            char = f.read(1)
            char = f.read(1)
        # {EOP : not char, EOF}

    except:
        # Error message
        print("Tidak ditemukan file dengan nama tersebut\n")

    return graph_list

def find_zero_indegree(graph_list):
    # Mengembalikan list index matkul pada graph yang memiliki in-degree = 0
    zero_indegree = []
```

```
    for i in range(len(graph_list)):
        if (len(graph_list[i][1]) == 0):
            zero_indegree.append(i)
    return zero_indegree

def delete_node(idx_list,graph_list):
    for i in range(len(idx_list)):
        matkul = graph_list[idx_list[i]][0] # Mengambil kode matkul
        # Menghapus kode matkul pada setiap preq node pada graph_list
        neff = len(graph_list)
        j = 0
        while (j < neff):
            try:
                graph_list[j][1].remove(matkul)
                j += 1
            except:
                j += 1

    # Menghapus simpul pada graf
    while (len(idx_list) != 0):
        del graph_list[idx_list[0]]
        idx_list.pop(0)
        try:
            for i in range(len(idx_list)):
                idx_list[i] -= 1
        except:
            break

def topo_sort(graph_list,queue):
    idx_list = find_zero_indegree(graph_list) # index matkul dgn in-degree = 0
    for i in range(len(idx_list)):
        queue.append(graph_list[idx_list[i]][0]) # Masukkan matkul ke queue
    delete_node(idx_list,graph_list) # Mengurangi & mendelete node pada graf

def to_rome(number):
    # Mengembalikan angka romawi
    switcher = {
        1: "I    ",
        2: "II   ",
        3: "III  ",
        4: "IV   ",
        5: "V    ",
```

```
        6: "VI   ",
        7: "VII  ",
        8: "VIII ",
    }
    return switcher.get(number,"nothing")

def print_solusi(queue,sem):
    # Mencetak format solusi
    print("Semester",to_rome(sem),":",end="")
    for i in range(len(queue)):
        print(queue[i],end="")
        if (i != len(queue)-1):
            print(",",end="")

    while(len(queue) != 0):
        # Hapus elemen queue, karena sudah dicetak
        queue.pop(0)

# Main Program
if __name__ == "__main__":
    filename = str(input("Masukkan nama file: "))
    graph_list = get_files(filename)
    queue = [] # Matkul akan ditaruh di queue
    sem = 1
    while (len(graph_list) != 0):
        # Selama graf belum selesai diproses
        topo_sort(graph_list,queue)
        print_solusi(queue,sem)
        if(len(graph_list) != 0):
            print("")
        sem += 1
    # {EOP : len(graph_list) == 0, graf selesai diproses}
    print(".")
```

3. Testing Program

Tabel 1. Screenshot Program

No	Input	Output

1	C1,C3. C2,C1,C4. C3. C4,C1,C3. C5,C2,C4. Gambar 1.1 Test Case ‘basic.txt’	Masukkan nama file: basic.txt Semester I :C3 Semester II :C1 Semester III :C4 Semester IV :C2 Semester V :C5. Gambar 1.2 Hasil ‘basic.txt’
2	IF2121. IF2110. IF2120. IF2124. IF2123. IF2130. Gambar 2.1 Test Case ‘1.txt’	Masukkan nama file: 1.txt Semester I :IF2121,IF2110,IF2120,IF2124,IF2123,IF2130. Gambar 2.2 Hasil ‘1.txt’
3	IF2220,MA1101,MA1201,IF2120. MA1101. MA1201. IF2120. IF2123. Gambar 3.1 Test Case ‘2.txt’	Masukkan nama file: 2.txt Semester I :MA1101,MA1201,IF2120,IF2123 Semester II :IF2220. Gambar 3.2 Hasil ‘2.txt’
4	IF4061,IF3151,IF2220. IF2220,MA1101,MA1201,IF2120. MA1101. MA1201. IF2120. IF3151,IF2250. IF2250. Gambar 4.1 Test Case ‘3.txt’	Masukkan nama file: 3.txt Semester I :MA1101,MA1201,IF2120,IF2250 Semester II :IF2220,IF3151 Semester III :IF4061. Gambar 4.2 Hasil ‘3.txt’
5	ET4040,ET2208,ET3101. ET2208,ET2101. ET2101,MA1201. MA1201. ET3101,ET2109,ET2204. ET2109,MA1201. ET2204,ET2103. ET2103. Gambar 5.1 Test Case ‘4.txt’	Masukkan nama file: 4.txt Semester I :MA1201,ET2103 Semester II :ET2101,ET2109,ET2204 Semester III :ET2208,ET3101 Semester IV :ET4040. Gambar 5.2 Hasil ‘4.txt’
6	IF4073,IF3270,IF3260. IF3260,IF2130,IF2110,IF2123. IF2130. IF2110. IF2123,MA1101. MA1101. IF3270,IF3170,IF2110.	Masukkan nama file: 5.txt Semester I :IF2130,IF2110,MA1101,IF2121,IF2211,MA1201,IF2120 Semester II :IF2123,IF2124,IF2220 Semester III :IF3260,IF3170 Semester IV :IF3270 Semester V :IF4073. Gambar 6.2 Hasil ‘5.txt’

	IF3170,IF2121,IF2124,IF2220,IF2211. IF2121. IF2124,IF2120,IF2110. IF2220,MA1101,MA1201,IF2120. IF2211. MA1201. IF2120. Gambar 6.1 Test Case ‘5.txt’	
7	KU1102. MA1101. IF1210,KU1102. IF2120,MA1101,MA1201. IF2220,MA1101,MA1201,IF2120. IF2123,MA1101,MA1201. IF3170,IF2220. IF3270,IF3170. MA1201,MA1101. Gambar 7.1 Test Case ‘6.txt’	Masukkan nama file: 6.txt Semester I :KU1102,MA1101 Semester II :IF1210,MA1201 Semester III :IF2120,IF2123 Semester IV :IF2220 Semester V :IF3170 Semester VI :IF3270. Gambar 7.2 Hasil ‘6.txt’
8	EL4243,EL3009,EL3010. EL3009,EL2005. EL3010,EL2007,EL3110. EL3109,EL3009,EL2205. EL2005,EL2001. EL2205,EL2005. EL2007,EP2094,EL1200. EL1200,MA1101,MA1201,FI1201. MA1101. MA1201. FI1201. EP2094,MA2072,EL2001. MA2072. EL2001,EB2102,EL1200,EL2101. EL3110. EB2102. EL2101. Gambar 8.1 Test Case ‘7.txt’	Masukkan nama file: 7.txt Semester I :MA1101,MA1201,FI1201,MA2072,EL3110,EB2102,EL2101 Semester II :EL1200 Semester III :EL2001 Semester IV :EL2005,EP2094 Semester V :EL3009,EL2205,EL2007 Semester VI :EL3010,EL3109 Semester VII :EL4243. Gambar 8.2 Hasil ‘7.txt’

9	ET4048,ET3206. ET3206,ET2202,ET3100. ET2202,ET2103. ET2103,EL1200. EL1200,MA1101,MA1201,FI1201. MA1101. MA1201. FI1201,FI1101. FI1101. ET3100,ET2200,MA2074. MA2074,MA2024. MA2024,MA2101. MA2101,MA1101. ET2200,ET2103,MA2072. MA2072,MA2021. MA2021. Gambar 9.1 Test Case ‘8.txt’	Masukkan nama file: 8.txt Semester I :MA1101,MA1201,FI1101,MA2021 Semester II :FI1201,MA2101,MA2072 Semester III :EL1200,MA2024 Semester IV :ET2103,MA2074 Semester V :ET2202,ET2200 Semester VI :ET3100 Semester VII :ET3206 Semester VIII :ET4048. Gambar 9.2 Hasil ‘8.txt’

4. Alamat Kode Program

Link Github : https://github.com/raihanastrada/stima-topological-sort

5. Checklist Program

Tabel 2. Checklist Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi	V	
2. Program berhasil *running*	V	
3. Program dapat menerima berkas input dan menuliskan output	V	
4. Luaran sudah benar untuk semua kasus input	V	