Naufal Yahya Kurnianto 13519141
IF2211 Strategi Algoritma K-03

LAPORAN TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

1. Algoritma Topological Sort

Topological sort merupakan suatu algoritma sorting untuk menyusun Directed Acyclic Graph. Directed Acyclic Graph (DAG) merupakan graf yang menandakan bahwa tiap simpulnya memiliki setidaknya satu sisi yang mengarah ke simpul tersebut atau pergi dari simpul tersebut. Menggunakan topological sort, kita dapat mengetahui arah gerak dari simpul-simpul tersebut dari yang tidak memiliki simpul lain sebagai prasyarat hingga simpul terakhir yaitu simpul yang tidak berperan sebagai prasyarat simpul lain.

Masalah yang dapat direkayasa menjadi DAG contohnya adalah pengambilan mata kuliah. Diketahui beberapa mata kuliah yang memiliki prasyarat dan juga merupakan prasyarat bagi mata kuliah lain. Hal ini menyebabkan terbentuknya DAG yang kemudian dapat dicari solusinya menggunakan topological sort.

Menggunakan topological sort, langkah pertama yang dilakukan adalah mencari mata kuliah yang tidak memiliki prasyarat sama sekali. Setelah mengeluarkan atau mencatat mata kuliah tersebut, cari mata kuliah lain yang memiliki mata kuliah yang tercatat tersebut sebagai prasyaratnya. Karena mata kuliah prasyarat telah diambil yang berarti mata kuliah baru ini tidak lagi memiliki prasyarat sehingga bisa dikeluarkan atau dicatat. Langkah tersebut kemudian akan terus diulang hingga seluruh mata kuliah dapat dikeluarkan atau dicatat. Maksud dari dikeluarkan adalah dikeluarkan dari list mata kuliah yang belum diambil. Perhatikan bahwa list mata kuliah yang belum diambil akan terus berkurang. Dengan menerapkan variasi decrease and conquer yaitu mengurangi ukuran instansi (di sini mata kuliah) dengan nilai konstan yang sama. Nilai konstannya adalah satu karena mata kuliah yang diambil (dikeluarkan dari instansi) satu-persatu hingga semuanya sudah diambil. Kita dapat mendapatkan solusi dari permasalahan pengambilan mata kuliah.

2. Source Code Program dalam Bahasa Python

```
#open file
print("Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya")
name = input("Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt\n")
file = open("../test/"+name,"r")

# listall merupakan list yang berisi string per line dari file yang masuk
# contoh: c1, c2.
#         c2, c3.
#         c3.
# listall = ["c1 c2", "c2 c3", "c3"]
listall = []
isi = file.readlines()
for i in isi:
    temp = ""
```

```
    for j in i:
        if(j==','):
            continue
        elif(j=='.'or j=="\n"):
            break
        else:
            temp+=j
    listall.append(temp)

# matkul adalah jumlah matkul yang ada
matkul = len(listall)
# data adalah matriks yang menandakan relasi matkul dengan pre-requisitenya dengan komposisi
# baris = matkul, kolom = pre-requisite matkul tersebut
data = [[0 for x in range(matkul)] for y in range(matkul)]

# leter_count adalah panjang string dari nama matkul, contoh c1 = 2; if2211 = 6
letter_count = 0
for i in listall[0]:
    if (i==' '):
        break
    letter_count += 1

# individual adalah list matkul yang ada tanpa pre-requisitenya (olahan dari listall)
individual = []
for i in listall:
    temp = ""
    for j in range (letter_count):
        temp+= i[j]
    individual.append(temp)

# membentuk data dengan kondisi keadaan yang real (sinkronisasi dengan info dari file masukan)
for i in range (len(data)):
    # jika panjang string matkul + pre-req lebih dari panjang string matkul (berarti ada pre-req)
    if (len(listall[i])>letter_count):
        # menghitung jumlah pre-req dari suatu matkul
        prec_amount = 0
        for j in listall[i]:
            if(j==' '):
                prec_amount += 1
        # perulangan pre-req untuk mengisi matriks
        # cara kerjanya sebagai berikut
        prec_proc = 0
        while (prec_proc<prec_amount):
            for j in range (len(data[i])):
                correct = True
                # loop dari indeks awal string pre-req
                # perhatikan bahwa acuan yang digunakan untuk mendata pre-req adalah listall
                # pada contoh awal, listall = ["c1 c2", "c2 c3", "c3"], maka start = 3
                start = (letter_count+1)*(prec_proc+1)
                for k in range (letter_count):
                    # listall[i] dengan indeks start = 3
                    # maka "c2" akan dicek dengan masing-masing elemen dari list individual
                    if (listall[i][start]!=individual[j][k]):
                        correct = False
                        break
                    start += 1
                # jika cocok (posisi pre-req sama pada matriks untuk menandakan ada pre-req)
                # matriks akan diisi dengan 1
                if (correct):
                    data[i][j] = 1
                    prec_proc += 1
                    break
                # hasil akhir matriks dengan contoh listall = ["c1 c2", "c2 c3", "c3"],
                # individual = ["c1", "c2", "c3"]
```

```
                # 0 1 0
                # 0 0 1
                # 0 0 0
                # yang berarti c1 [indeks 0 - baris 1] memiliki pre-req c2 [indeks 1 - kolom 2]
                # c2 [indeks 1 - baris 2] memiliki pre-req c3 [indeks 2 - kolom 3]
                # c3 tidak memiliki pre-req
        # loop akan berjalan hingga pre-req yang terproses (terisi di matriks) sudah sesuai dengan
        # jumlah pre-req yang telah dikalkukasikan di awal (prec_amount)

# list sorted untuk topological sort
sorted = []
# current index untuk menandakan indeks sorted yang sedang digunakan
curr_index = 0
# count semester untuk keluaran
semester = 1

# akan dilakukan perulangan hingga matkul di list sorted berjumlah sama dengan jumlah matkul yang ada
while (len(sorted)<matkul):
    # gone menandakan matkul yang akan disalin dari list individual ke list sorted
    gone = []
    # proses sort dilakukan dengan memroses matriks yang telah dibuat
    for i in range (len(data)):
        clear = True
        visited = False
        # clear di sini menandakan bahwa seluruh kolom bernilai 0 (tidak memiliki pre-requisite)
        # perhatikan perulangan terus dilakukan pada baris matriks
        for j in range (len(data[i])):
            if (data[i][j]!=0):
                clear = False
        # setelah menemukan baris yang memiliki kolom dengan seluruh nilainya 0
        # akan dicek apakah sudah tercatat atau belum di list sorted
        if(clear):
            for k in sorted:
                if(k==individual[i]):
                    visited = True

        # jika clear dan belum ada di sorted maka matkul akan dimasukkan ke list sorted
        if(clear and not visited):
            sorted.append(individual[i])
            gone.append(i)

    # proses keluaran agar sesuai dengan spek
    while(curr_index<len(sorted)):
        # jika pada 1 semester dapat mengambil lebih dari 1 sks
        if(len(sorted)-curr_index>1):
            print("semester",semester,": ",end='')
            while(len(sorted)-curr_index>=1):
                print(sorted[curr_index],end='')
                curr_index += 1
                if(len(sorted)-curr_index>=1):
                    print(", ",end='')
            print()
        # jika hanya 1 sks yang dapat diambil pada semester tersebut
        else:
            print("semester",semester,":",sorted[curr_index])
            curr_index += 1

    # update count semester untuk keluaran
    semester += 1

    # karena matkul pada indeks gone sudah disalin ke list sorted
    # menandakan bahwa matkul telah diambil sehingga
    # dapat dianggap jumlah pre-req matkul lain yang bersangkutan ada yang berkurang
    for i in range (len(data)):
```

```
        for j in range (len(data[i])):
            for k in gone:
                if (j==k):
                    data[i][j] = 0

    # ulangi hingga list sorted berjumlah sama dengan jumlah matkul yang ada
    # yang berarti tidak ada lagi pre-requisite yang diproses (matkul beserta pre-req sudah terambil
semua)
    # karena jumlah pre-req terus berkurang tiap semesternya (decrease and conquer)
```

3. Input dan Output

Format yang saya gunakan di sini adalah output di sebelah kiri dan file input di sebelah kanan.

3.1. Test-Case 1

```
Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya
Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt
1.txt
semester 1 : c3
semester 2 : c1
semester 3 : c4
semester 4 : c2
semester 5 : c5
>>>
```

1 - Notepad

```
File  Edit  Format  View  Help
c1, c3.
c2, c1, c4.
c3.
c4, c1, c3.
c5, c2, c4.
```

3.2. Test-Case 2

```
Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya
Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt
2.txt
semester 1 : c3
semester 2 : c2
semester 3 : c1
>>>
```

2 - Notepad

```
File  Edit  Format  View  Help
c1, c2.
c2, c3.
c3.
```

3.3. Test-Case 3

```
Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya
Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt
3.txt
semester 1 : c4
semester 2 : c1, c2
semester 3 : c3
semester 4 : c5
>>>
```

3 - Notepad

```
File  Edit  Format  View  Help
c1, c4.
c2, c4.
c3, c1.
c4.
c5, c3.
```

3.4. Test-Case 4

```
Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya
Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt
4.txt
semester 1 : c05
semester 2 : c08
semester 3 : c06
semester 4 : c07
semester 5 : c03
semester 6 : c01
semester 7 : c02
semester 8 : c04
>>>
```

4 - Notepad

```
File  Edit  Format  View  Help
c01, c03, c08.
c02, c01, c07
c03, c07, c05.
c04, c02.
c05.
c06, c08.
c07, c06
c08, c05.
```

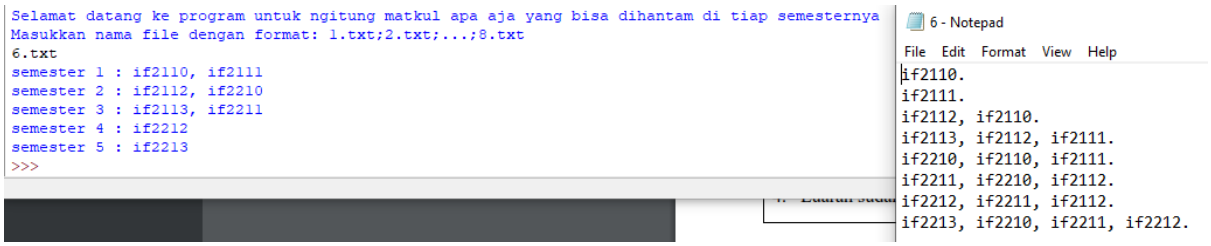
3.5. Test-Case 5

```
Selamat datang ke program untuk ngitung matkul apa aja yang bisa dihantam di tiap semesternya
Masukkan nama file dengan format: 1.txt;2.txt;...;8.txt
5.txt
semester 1 : bajep001
semester 2 : bajep101, bajep102
semester 3 : bajep103
semester 4 : bajep111
semester 5 : bajep112
semester 6 : bajep113
semester 7 : bajep333
>>>
```

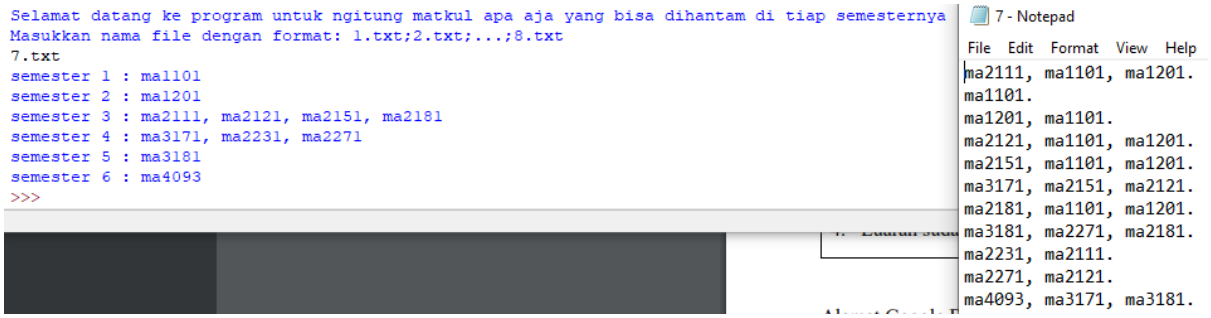
5 - Notepad

```
File  Edit  Format  View  Help
bajep001.
bajep101, bajep001.
bajep102, bajep001.
bajep103, bajep101, bajep102.
bajep111, bajep103.
bajep112, bajep111.
bajep113, bajep112.
bajep333, bajep101, bajep102, bajep103, bajep111, bajep112, bajep113.
```

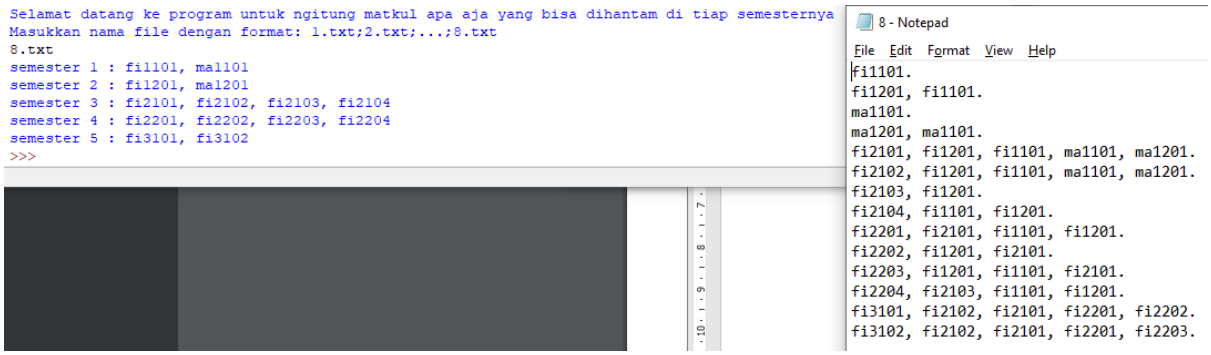
3.6. Test-Case 6



3.7. Test-Case 7



3.8. Test-Case 8



4. Repo Github

Berikut repository github yang dapat diakses sebagai alternatif tempat pengaksesan source code, disertakan pula readme. https://github.com/ayahyaaa/IF2211-Stima-Tucil2

5. Tabel Pencapaian Target

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.	✓	