**Laporan Tugas Kecil 2 IF2211 Strategi Algoritma**

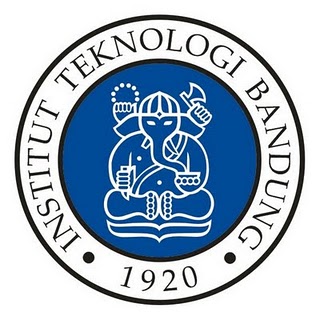
**Penyusunan Rencana Kuliah dengan *Topological Sort***

**(Penerapan *Decrease and Conquer*)**

Oleh :

Muhammad Fikri. N

13519069

****

**INSTITUT TEKNOLOGI BANDUNG**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**TEKNIK INFORMATIKA**

**2020/2021**

BAB I

DESKRIPSI MASALAH

Penyusunan rencana pengambilan mata kuliah adalahh proses pengambilan mata kuliah yang akan dilaksanakan mahasiswa pada semester yang akan datang. Mata kuliah dapat memiliki prerequisite mata kuliah yang harus diambilnya sebelum mengambil mata kuliah tersebut. Permasalahan penyusunan rencana pengambilan mata kuliah dapat dilakukan dengan memanfaatkan algoritma *Decrease and Conquer*. *Topological sort* merupakan salah satu implementasi untuk menyelesaikan permasalahan penyusunan rencana pengambilan mata kuliah

Contoh :

1. Input file teks berisi :

C1, C3.

C2, C1, C4.

C3.

C4, C1, C3.

C5, C2, C4.

Solusinya adalah :

Semester 1 : C3.

Semester 2 : C1.

Semester 3 : C4.

Semester 4 : C2 .

Semester 5 : C5.

2. Input file teks berisi :

C1, C2.

C2.

Solusinya adalah :

Semester 1 : C2.

Semester 2 : C1.

3. Input file teks berisi :

C1, C2.

Solusinya adalah :

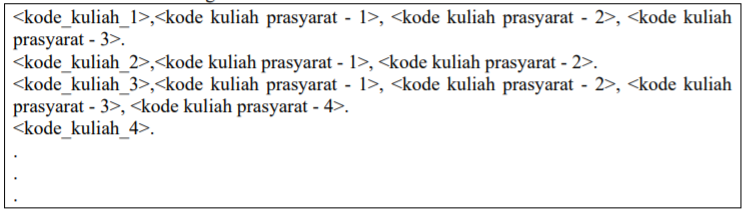
Tidak ada urutan yang mungkin.

Spesifikasi :

Membuat aplikasi sederhana yang dapat menyusun rencana pengambilan kuliah, dengan memanfaatkan algoritma Decrease and Conquer dalam bahasa C/C++/Java/Python. Penyusunan rencana kuliah diimplementasikan dengan menggunakan pendekatan Topological Sorting. Batasan untuk aplikasi ini adalah untuk setiap semester tidak terdapat batasan pengambilan mata kuliah yang bisa diambil dan pembatasan 8 semester dalam penyusunan rencana mata kuliah. Namun, untuk batasan 8 semester, aplikasi ini memberikan opsi pilihan untuk tetap menampilkan hasil penyusunan rencana mata kuliah atau tidak.

Input :

File teks dengan format di bawah ini :



Gambar 1. Format File Teks untuk Masukan Daftar Kuliah

<Sumber : File Spek Tucil 2>

Contoh isi file teks:

C1, C3.

C2, C1, C4.

C3.

C4, C1, C3.

C5, C2, C4.

Output:

Semester 1 : C3.

Semester 2 : C1.

Semester 3 : C4.

Semester 4 : C2 .

Semester 5 : C5.

BAB II

ALGORITMA TOPOLOGICAL SORT

Topological sorting merupakan suatu algoritma yang digunakan untuk memproses suatu Directed Acyclic Graph (DAG) sehingga mendapatkan keterurutan dari setiap simpul.

Bahasa pemrograman yang digunakan adalah python.

Pemecahan masalah penyusunan rencana kuliah menggunakan algoritma topological sort :

1. Baca file teks kemudian hasilnya disimpan ke array yang berisi matkul beserta *prerequisite* nya

([elm\_1,elm\_2,elm\_3, ... , elm\_N ])

dengan N merupakan jumlah baris dari file teks dan untuk setiap elm\_i dengan 1 <= i <= N

akan berbentuk [matkul,prereq\_1,\_prereq\_2, ... , prereq\_M] sebut saja array Matkul. M mungkin 0 yang berarti suatu matkul tidak memiliki prerequisite.

1. Dibentuk representasi graf menggunakan tipe data dictionary.

dict dengan key yaitu matkul dan value berbentuk dictionary juga. value memiliki dua key yaitu N\_preq dan Arr\_preq. N\_preq yaitu banyaknya *prerequisite* yang blm diambil dan Arr\_preq adalah array dari *prerequisite*.

1. Dibentuk array of array dgn nama Semester yang akan menampung mata kuliah tiap semester

Semester = [[semester1],[semester2],[semester3], ... , [semester40]]

Dalam hal ini, diinisialisasi maksimal menampung 40 semester atau dengan indeks 0 sampai 39.

Pada penulisan hasil, akan dilakukan pengecekan agar maksimal 8 semester saja. Namun, dapat memilih opsi tetap menampilkan hasil penyusunan jika ingin melihat hasil walaupun lebih dari 8 semester.

1. Setelah itu, dilakukan pengecekan apakah ada N\_preq yang bernilai 0. (Jika tidak berarti mata kuliah dan *prerequisite* nya tidak berbentuk Directed Acyclic Graph)
2. Jika langkah 4 = True, maka akan dilakukan proses selama dictionary tdk kosong ({}) dan langkah 4 = True.
3. Proses nya adalah dilakukan iterasi untuk semua elemen pada Matkul, jika terdapat dictionary yang key nya adalah elm\_i[0] dengan 1 <= i <= N atau dengan kata lain adalah elemen matkul (refer ke poin 1) dan key dari valuenya adalah N\_preq yang bernilai 0. Maka matkul tersebut akan dimasukkan ke array temporary untuk menampung matkul yang sudah diambil. Setelah itu, dilakukan penghapusan *record* dictionary yang memiliki key elm\_i[0] tadi yang telah diambil. Kemudian, juga dilakukan pengurangan N\_preq untuk setiap *record* dictionary yang *prerequisite* nya adalah elm\_i[0] tersebut dan penghapusan array Matkul yang matkul = elm\_i[0].
4. Lakukan pengecekan pada langkah 5 hingga langkah 5 = False
5. Setelah itu, diperoleh array Semester yang telah berisi matkul-matkul hasil penyusunan matkul per semester. Jika semester tidak melebihi 8 semester maka akan langsung diberikan hasilnya, jika tidak maka akan ada pilihan apakah tetap akan ditampilkan atau tidak dan kemudian memberikan saran untuk melakukan pengaturan pilihan matkul.
6. Pada langkah 8, jika semester tidak melebihi 8 maka akan dilakukan pengecekan jika dict kosong ({}) berarti semua matkul dapat terambil dan jika dictionary tdk memiliki matkul yang N\_preq sama dengan 0 berarti proses berhenti di tengah jalan (tidak terdapat lagi DAG setelah diproses).

Hubungan algoritma topological sort dengan pendekatan algoritma *decrease and conquer* adalah dengan penghapusan vertex yang telah diambil dan pengurangan N\_preq / N\_masuk jika berhubungan dengan vertex yang dihapus tadi. Varian *decrease and conquer* yang digunakan pada topological sort adalah *decrease by a constant*

BAB III

SOURCE PROGRAM

File util.py

*'''  
Author : Muhammad Fikri. N  
NIM : 13519069  
Topik : Topological Sort  
'''*# Untuk menghapus string kosong  
def delete\_arr\_empty(arr):  
 while ("" in arr):  
 arr.remove("")  
 return arr  
  
# Untuk menghapus elemen arr / matkul (berisi matkul beserta prereq) yang telah diambil (arrTakes)  
def delete\_arr(arr,arrTakes):  
 while (len(arrTakes) > 0):  
 for content in arr:  
 course = content[0]  
 for take in arrTakes:  
 if course == take:  
 arr.remove(content)  
 arrTakes.remove(take)  
 return arr  
  
# Untuk menghapus record dictionary yang memiliki kunci = key  
def delete\_dict(dict,key):  
 for course in dict.keys():  
 if course == key:  
 dict.pop(key)  
 return dict  
  
# Untuk mengurangi banyaknya prerequisite yang blm diambil  
def minus\_Npreq(dict,arrTakes):  
 for course, info\_course in dict.items():  
 for take in arrTakes:  
 if take in info\_course['Arr\_preq']:  
 info\_course["N\_preq"] -= 1  
  
# Untuk mengetahui apakah ada matkul yang banyak prerequisite yang blm diambil = 0  
def is\_Npreq\_have\_zero(dict):  
 bool = False  
 for course, info\_course in dict.items():  
 if info\_course["N\_preq"] == 0:  
 return True  
  
 return bool  
  
# Untuk convert array to string  
def arrayToString(arr):  
 string = ""  
 for i in range(len(arr)):  
 if (i < (len(arr)-1)):  
 string += arr[i]  
 string += ", "  
 else:  
 string += arr[i]  
 string += "."  
 return string

File main.py

*'''  
Author : Muhammad Fikri. N  
NIM : 13519069  
Topik : Topological Sort  
'''*import os  
import util as ut  
  
print(" \_\_\_\_\_\_\_\_ .\_\_ .\_\_\_ ")  
print(" \\_\_\_\_\_ \ | | \_\_\_\_ \_\_\_\_ \_\_| \_/\_\_\_\_ \_\_\_ \_\_\_")  
print(" / | \| | \_/ \_\_\_\/ \_ \ / \_\_ |/ \_\_ \\\ \/ /")  
print(" / | \ |\_\ \\_\_( <\_> ) /\_/ \ \_\_\_/ > < ")  
print(" \\_\_\_\_\_\_\_ /\_\_\_\_/\\_\_\_ >\_\_\_\_/\\_\_\_\_ |\\_\_\_ /\_\_/\\_ \\")  
print(" \/ \/ \/ \/ \/\n")  
  
print("-----------------------------")  
print("| Silahkan input nama file |")  
print("| (eg : tc) |")  
print("| (Tanpa ekstensi file) |")  
print("-----------------------------")  
  
print()  
  
filename = input("Input nama file : ")  
path = "../test/" + filename + ".txt"  
  
isfile = os.path.isfile(path)  
  
# Melakukan pengecekan apakah filename terdapat pada folder test atau tidak  
while (not isfile):  
 filename = input("Ulangi input nama file : ")  
 path = "../test/" + filename + ".txt"  
 isfile = os.path.isfile(path)  
  
test = open(path, 'r')  
  
temp = test.read().splitlines() # baca file teks (dengan readlines yang sekaligus hapus \n)  
  
# delete string kosong (ketika file .txt memiliki banyak enter tp tidak ada record)  
ut.delete\_arr\_empty(temp)  
  
temps = []  
  
for i in range(len(temp)):  
 elm = temp[i].replace('.',"")  
 elm = elm.replace(" ","")  
 temps.append(elm)  
  
arr = []  
  
for i in range(len(temps)):  
 # mengubah array of string menjadi array of array string dgn delimiter koma (,)  
 content = temps[i].split(',')  
  
 arr.append(content)  
  
dict = {}  
  
for i in range(len(arr)):  
 N\_preq = (len(arr[i][1:len(arr[i])])) # N\_preq adalah banyaknya prerequisite yang blm diambil  
 Arr\_preq = (arr[i][1:len(arr[i])])  
 dict[arr[i][0]] = { # Key = Matkul  
 "N\_preq": N\_preq, # Value = dictionary (key1 = N\_preq (banyak prerequisite yang blm diambil)  
 "Arr\_preq": Arr\_preq # key2 = Arr\_preq (array dari matkul prerequisite))  
 }  
  
semester = [[]for i in range(40)] # Asumsi maksimal 40 semester yang dapat ditampung  
  
temporary = []  
i = -1  
  
print()  
print("-----------------------------")  
print("| Hasilnya adalah |")  
print("| ↓ ↓ ↓ ↓ ↓ ↓ |")  
print("-----------------------------")  
print()  
  
if ut.is\_Npreq\_have\_zero(dict):  
 while (dict != {} and ut.is\_Npreq\_have\_zero(dict)):  
 i += 1  
  
 # untuk mengulang array temporary kembali kosong (berisi matkul yang diambil)  
 temporary = []  
  
 for content in arr:  
 if dict[content[0]]["N\_preq"] == 0:  
 take = content[0]  
 semester[i].append(take)  
 temporary.append(take)  
  
 # Menghapus course yang telah diambil pada dict  
 ut.delete\_dict(dict,take)  
  
 ut.minus\_Npreq(dict,temporary)  
 ut.delete\_arr(arr,temporary)  
  
else:  
 print("Tidak ada urutan yang mungkin")  
 print("Pastikan kuliah dan prerequisite nya berupa Directed Acyclic Graph (DAG)")  
 print("Silahkan atur pilihan matkul kamu lagi, FIGHTING (～￣▽￣)～")  
  
semester = [x for x in semester if x] # Untuk menghapus empty array dari sebuah array  
  
if len(semester) <= 8:  
 if (dict == {}):  
 for i in range(len(semester)):  
 print("Semester " + str(i+1) + ": ", end ="")  
 print(ut.arrayToString(semester[i]))  
 elif (not ut.is\_Npreq\_have\_zero(dict)):  
 for i in range(len(semester)):  
 print("Semester " + str(i+1) + ": ", end ="")  
 print(ut.arrayToString(semester[i]))  
 print("Tidak ada urutan yang mungkin lagi")  
 print("Silahkan atur pilihan matkul kamu lagi, FIGHTING (～￣▽￣)～")  
  
  
else:  
 print("Hasil proses melebihi 8 semester !!!")  
 print("Tidak boleh melebihi 8 semester :(")  
 print("Apakah tetap akan ditampilkan atau tidak perlu ?")  
 print("-----------------------------")  
 print("| Opsi input |")  
 print("| 1. YA |")  
 print("| 2. TIDAK |")  
 print("-----------------------------")  
  
 pil = int(input("Input pilihan : "))  
 while (pil != 1 and pil != 2):  
 pil = int(input("Ulangi input pilihan, pastikan pilihan 1 atau 2 : "))  
 if pil == 1:  
 for i in range(len(semester)):  
 print("Semester " + str(i + 1) + ": ", end="")  
 print(ut.arrayToString(semester[i]))  
 print("\nJangan lupa atur pilihan matkul kamu lagi, FIGHTING (～￣▽￣)～")  
 else:  
 print("\nSilahkan atur pilihan matkul kamu lagi, FIGHTING (～￣▽￣)～")

BAB IV

SCRENSHOOT I/O

Test Case 1 (File tc.txt)

Text

Description automatically generated with low confidence

Test Case 2 (File tc1.txt)

Text

Description automatically generated

Test Case 3 (File tc2.txt)

Text

Description automatically generated

Test Case 4 (File tc3.txt)

Text

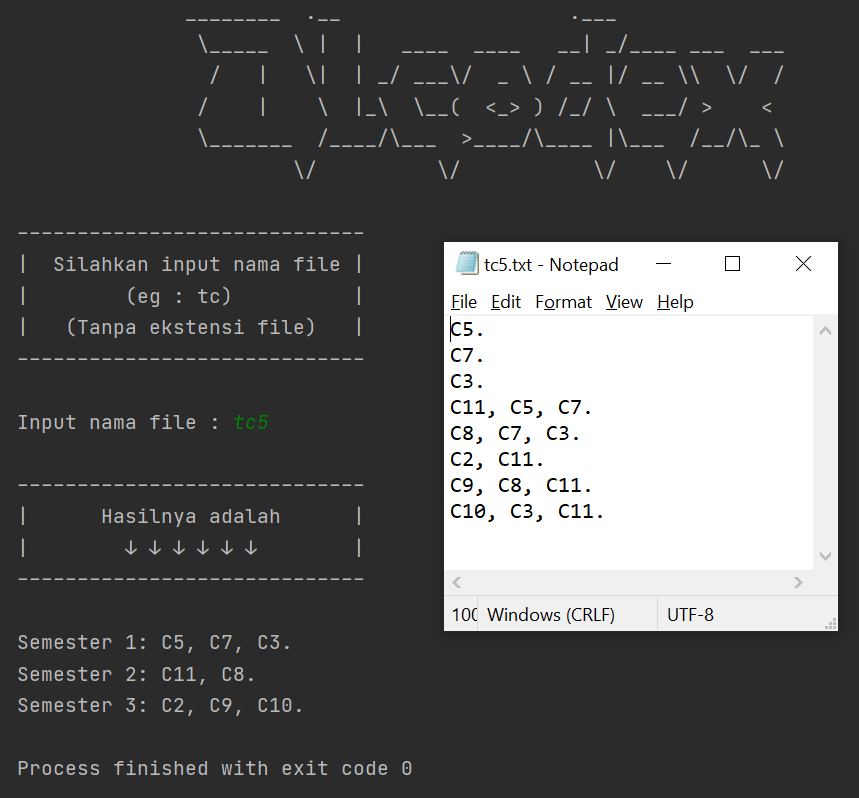
Description automatically generated

Test Case 5 (File tc4.txt)

Graphical user interface, text

Description automatically generated

Test Case 6 (File tc5.txt)



Test Case 7 (File tc6.txt)

Text

Description automatically generated with medium confidence

Test Case 8 (File tc7.txt)

Text

Description automatically generated

Test Case 9 (File tc8.txt)

Text

Description automatically generated Text

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

Test Case 10 (File tc9.txt)

Text

Description automatically generated

BAB V

CEK LIST

|  |  |  |
| --- | --- | --- |
| 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 |  | - |

Link Github :