Tugas Kecil 2 IF2211 Strategi Algoritma

Aplikasi Algoritma Decrease And Conquer untuk Menyelesaikan Permasalahan Pengambilan Mata Kuliah dengan Prerequisite dengan Pendekatan Topological Sorting

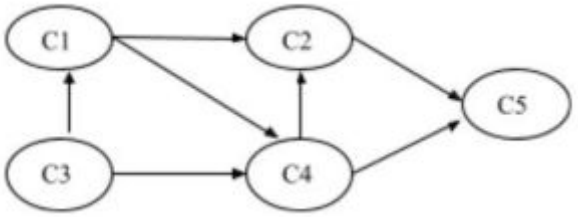
Nama: Rezda Abdullah Fachrezzi
NIM : 13519194

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

1. Deskripsi Singkat

Pada perkuliahan, mahasiswa wajib mengambil mata kuliah dengan jumlah tertentu sebagai syarat kelulusan, baik itu kuliah wajib dari program studi maupun kuliah pilihan dari dalam maupun luar program studi. Dari mata kuliah tersebut, ada yang terdapat *prerequisite* atau prasyarat untuk mengambilnya, yaitu berupa mata kuliah lain.

Mata kuliah beserta prasyarat pengambilannya dapat dimodelkan dengan *Directed Acrylic Graph* (DAG). DAG adalah graf yang tidak memiliki *cycle* atau siklus di dalamnya. Contoh dari model tersebut adalah sebagai berikut.



Gambar 1.1. Contoh mata kuliah dengan prasyaratnya yang dimodelkan dalam bentuk DAG. (diambil dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf)

Pada gambar 1.1, tiap simpul menggambarkan mata kuliah dan simpul yang berdekatan yang mengarah ke simpul tadi menggambarkan prasyarat pengambilannya. Sebagai contoh, mata kuliah **C1** memiliki prasyarat pengambilannya yaitu mata kuliah **C3**.

Walaupun sebenarnya dalam dunia nyata pengaturan pengambilan mata kuliah tersebut sudah diatur oleh kurikulum pada kampus terkait, tetapi pada laporan ini penulis akan membuat program sederhana yang dapat melakukan pengurutan pengambilan mata kuliah tersebut dengan prasyarat pengambilannya dengan menggunakan algoritma *Decrease and Conquer* dengan pendekatan *Topological Sorting* serta penjelasannya.

2. Penjelasan Pendekatan Topological Sorting yang digunakan.

Topological Sort merupakan sebuah proses pengurutan sedemikian rupa untuk suatu graf dengan simpul terhubung V dan E, V ada sebelum E. Topological Sorting yang digunakan pada program yang dibuat merupakan pendekatan dengan menggunakan algoritma decrease and conquer. Topological sorting ini hanya berlaku pada *Directed Acrylic Graph* (DAG), sehingga pada program ini diasumsikan semua input merupakan Graf berjenis DAG.

Secara umum, penerapan topological sorting tersebut dengan algoritma decrease dan conquer adalah sebagai berikut.

1. Program membaca fail yang di-input oleh pengguna.
2. Fail yang dibaca kemudian diubah menjadi struktur data sederhana simpul dengan bentuk berikut.

```
{ simpul: string, # nama simpul (misal A)
  masuk: array of string # simpul yang bersebelahan dengan A yang masuk ke simpul A }
```

3. Struktur data tersebut mewakili satu simpul yang kemudian akan dimasukkan ke dalam array of simpul dan kemudian disebut **Graf**.
4. Graf tersebut kemudian diurutkan simpulnya dengan pendekatan topological sorting dengan algoritma decrease and conquer. Proses ini menggunakan fungsi rekursi.
5. Pertama-tama, akan dicari simpul yang tidak memiliki simpul lain yang masuk ke simpul tersebut yang berarti **simpul.masuk** memiliki jumlah nol.
6. Simpul yang sesuai kemudian dipisahkan dari graf dan dimasukkan ke array kosong.
7. Array yang baru terisi tadi akan dikonkatenasi dengan array baru yang merupakan hasil rekursi dari graf baru yang sudah dihapus salah satu simpulnya pada proses enam. Array baru ini kemudian akan menjadi graf baru.
8. Proses lima sampai tujuh akan terus diulang sampai simpul pada graf hanya bersisa satu.
9. Proses pengurangan/penghapusan simpul dari graf pada proses enam merupakan proses ***decrease,*** sedangkan konkatenasinya merupakan proses ***conquer***.
10. Hasil akhir dari proses ini akan menghasilkan graf baru yang berisi simpul yang sudah diurutkan dengan pendekatan topological sorting.
11. Graf tadi kemudian akan di-*map* pada semester yang sesuai untuk dibuat urutan pengambilannya untuk tiap semester.

3. Source Code Program.
   Program yang dibuat menggunakan bahasa pemrograman Python. Pada program ini, terdapat 3 fail, yaitu **main.py**, **program.py**, dan **util.py**.

   a. Fail main.py
   Program utama untuk dijalankan.

```
import program

'''
Rezda Abdullah Fachrezzi
13519194

Tucil 2 IF2211 - Algoritma Decrease and Conquer

program.py berisi algoritma
util.py berisi fungsi-fungsi bantuan

Parameter pada fungsi start, yaitu
start(nointro: boolean, showsorted: boolean)
```

```
Apabila nointro bernilai True, program tidak akan menuliskan intro, vice versa.
(default: False)
Apabila showsorted bernilai True, program akan menuliskan graf yang sudah diurutkan.
(default: False)
'''

program.start()
```

b. Fail program.py

Fail berisi algoritma topological sorting serta program yang akan dijalankan.

```
from util import concat, removefromgraf, updategraf, matching, mappingsemester,
printgraf

def toso(graf):
 # implementasi pendekatan topological sort
 # dengan decrease and conquer (rekursi)

 # basis
 if(len(graf)==1):
   return graf
 # rekuren
 else:
   # akan dipilih simpul tanpa sisi masuk
   for data in graf:
     if len(data["masuk"])==0:
       simpul = data["simpul"]
       # decreasing
       decrease = updategraf(removefromgraf(graf, simpul), simpul)
       # conquering + recursion
       return concat([data], toso(decrease))

def parse(file):
 '''
 parsing file

 contoh isi file = A,B,C. dengan A adalah simpul dan B,C adalah simpul adjacent yang
menuju A.
 A adalah mata kuliah dengan B dan C adalah prerequisite-nya.

 Akan di-parse menjadi tipe data berikut, misal Graf
```

```
 {
   simpul: nama simpul (misal A),
   masuk: array of simpul bersebelahan dengan A yang masuk ke simpul A
 }

 Seluruh baris harus diakhiri dengan titik (spek soal) dan enter atau new line.

 Asumsi input adalah Directed Acrylic Graph

 Output berupa Graf.
 '''
  copy = file.readlines()

 # menghapus spasi, new line, dan titik
 # return 1 jika tidak ada titik di akhir baris
 for i in range(len(copy)):
   if copy[i][len(copy[i])-2] != ".":
     return 1
   copy[i] = copy[i].replace(" ", "").replace("\n","").replace(".","")

 out = []
 # mengolah baris-baris fail tadi lalu dipindahkan ke struktur data
 for simpul in copy:
   split = simpul.split(",")
   data = {
     "simpul": split[0],
     "masuk": []
   }
   for i, masuk in enumerate(split):
     if i>0:
       data["masuk"].append(masuk)
   out.append(data)

 return out

def intro():
 print("Selamat datang di Penentu Pengambilan Mata Kuliah!\n")
 print("Anda bingung mau ngambil kuliah apa dulu karena kebanyakan
prerequisite?\nGampang! Masukin aja daftar kuliahnya beserta prerequisite-nya ke
program ini, nanti bakal dikasih urutan pengambilan dalam bentuk tiap-tiap semester!")
 print("========================")
 print("Contoh isi file:\n[nama kuliah] [, prerequisite 1] [, prerequisite 2].")
 print("A.\nB, C, A.\nC, A.")
 print("========================")
```

```
 print("Program dikerjakan oleh Rezda A.F 13519194\n")

def start(nointro = False, showsorted = False):
 '''
 Menjalankan program dengan flow sbb.

 1. Meminta nama fail dan direktori jika ada
 2. Mem-parsing fail jika tidak ada error
 3. Memprosesnya jika tidak ada error
 4. Mengurutkan graf dengan pendekatan topological sort
 5. Mapping graf yang sudah diurutkan ke semester yang sesuai
 6. Mengeluarkannya ke layar
 '''

 if not nointro: intro()

 nama_file = input("Masukkan nama fail dengan ekstensinya (dan foldernya jika ada)\n>
")
 if nama_file=="exit": return

 try:
   file = open(nama_file, "r")
 except:
   print ("Error, terjadi kesalahan saat membaca fail.\n")
   start(True)
   return
  # parsing fail
 parsed = parse(file)

 # error handling
 if type(parsed) is int and parsed == 1:
   print ("Error, format isi fail salah!\nSetiap baris harus diakhiri titik.\n")
   start(True)
   return

 # membuka kembali fail untuk ditampilkan ke layar
 # dan untuk mencocokkan data graf yang sudah diurutkan
 file = open(nama_file, "r")
 parsed2 = parse(file)

 print("\nMata Kuliah beserta prerequisite-nya: ")
 printgraf(parsed2)

 sortedgraf = matching(parsed2, toso(parsed))
```

```
 if showsorted:
   print("\nMata Kuliah setelah diurutkan: ")
   printgraf(sortedgraf)

 # mapping semester
 semester = mappingsemester(sortedgraf)

 # menampilkan hasilnya ke layar
 i = 0
 print("\nUrutan pengambilan dalam semester:")
 for i, listmatkul in enumerate(semester):
   print("Semester", str(i+1), ":", end=" ")
   for i, matkul in enumerate(listmatkul):
     print (matkul, end=", " if i < len(listmatkul)-1 else "\n")
```

c. Fail util.py

Fail berisi fungsi-fungsi bantuan untuk program utama.

```
def printgraf(graf):
 # print graf
 for simpul1 in graf:
   print(simpul1["simpul"] + ": ", end="")
   if len(simpul1["masuk"]) == 0:
     print("-")
   else:
     for i, simpul2 in enumerate(simpul1["masuk"]):
       print (simpul2, end=", " if i < len(simpul1["masuk"])-1 else "\n")

def copyarr(a):
 # copy array
 b = []
 for data in a:
   b.append(data)
 return b

def concat(a,b):
 # konkatenasi array
 c = copyarr(a)
 for data in b:
   c.append(data)
 return c
```

```
def removefromgraf(arr, dat):
 # menghapus simpul dat dari graf arr
 out = []
 for data in arr:
   if data["simpul"]!=dat:
     out.append(data)
 return out

def removefrom(arr, dat):
 # menghapus string dat dari array arr
 out = []
 for data in arr:
   if data!=dat:
     out.append(data)
 return out

def updategraf(arr, dat):
 # mengupdate simpul masuk tiap-tiap simpul dengan
 # menghapus simpul dat
 for i in range(len(arr)):
   arr[i]["masuk"] = removefrom(arr[i]["masuk"], dat)
 return arr

def matching(inn, out):
 # merestore data pada graf out dari graf inn
 # karena graf yang diurutkan menjadi graf dengan simpul masuk kosong
 for data in inn:
   for i in range (len(out)):
     simpul = out[i]["simpul"]
     if(simpul == data["simpul"]):
       out[i]["masuk"] = data["masuk"]
 return out

def isinmasuk(a, simpul):
 # mengecek apakah simpul merupakan simpul yang masuk ke suatu simpul a
 for simp in a["masuk"]:
   if (simp==simpul):
     return True
 return False

def isprereqtaken(kuliah, semester):
 # mengecek prerequisite untuk suatu simpul (kuliah) pada array semester
 c = 0
```

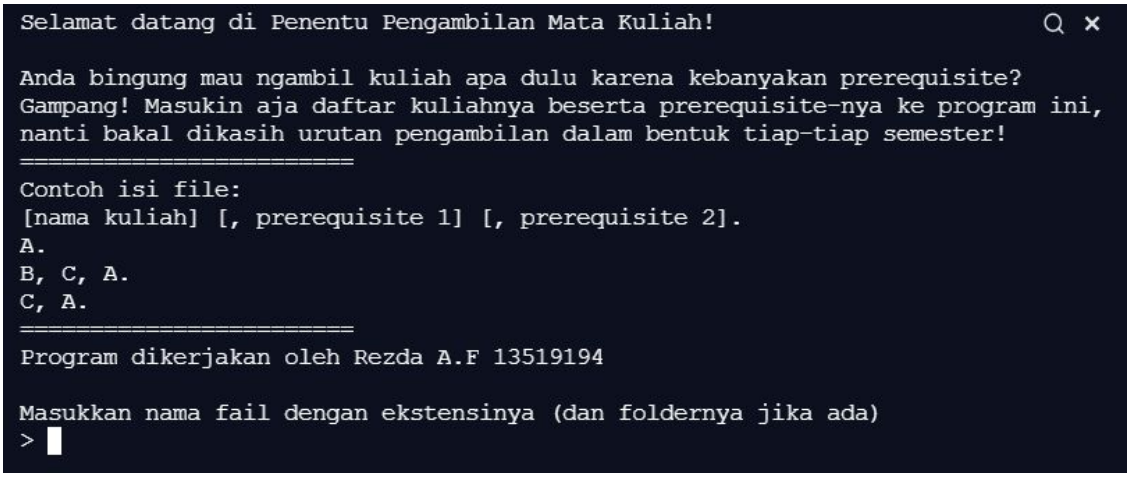
```
 for i, listmatkul in enumerate(semester):
   if i < len(semester)-1:
     for matkul in listmatkul:
       #print(matkul)
       if isinmasuk(kuliah, matkul):
         c += 1
 return len(kuliah["masuk"])==c

def mappingsemester(kuliah):
 # me-map graf kuliah yang sudah diurutkan ke tiap-tiap semester
 # asumsi tidak ada batas bawah dan atas untuk jumlah kuliah
 # asumsi tiap input selalu cukup untuk 8 semester (tidak lebih)

 '''
 cara kerja mapping semester
 1. Jika semester 1, maka seluruh kuliah (simpul) tanpa prerequisite akan dimasukkan, simpul tsb dihapus dari graf.
 2. Jika bukan semester 1, kuliah pertama yang didapat pada graf yang prereqnya sudah diambil akan dimasukkan, kuliah tersebut dihapus dari graf.
 3. Masih lanjutan nomor 2, kemudian akan dicek apakah kuliah lainnya memenuhi syarat untuk dimasukkan ke semester sekarang, yaitu jika tidak bersamaan dengan prereqnya dan prereqnya sudah diambil di semester sebelumnya, proses ini dilanjutkan sampai seluruh kuliah (simpul) pada graf sudah semuanya di-map.
 '''
  semester = []
 i = 0
 while len(kuliah)>0:
   semester.append([])
   for data in kuliah:
     if i == 0:
       if len(data["masuk"])==0:
         if isprereqtaken(data, semester):
           semester[i].append(data["simpul"])
           kuliah = removefromgraf(kuliah, data["simpul"])
     else:
       if len(semester[i])==0:
         semester[i].append(data["simpul"])
         kuliah = removefromgraf(kuliah, data["simpul"])
       else:
         for matkul in semester[i]:
           if not isinmasuk(data, matkul) and isprereqtaken(data, semester):
             semester[i].append(data["simpul"])
             kuliah = removefromgraf(kuliah, data["simpul"])
             break
```

```
  i += 1
return semester
```

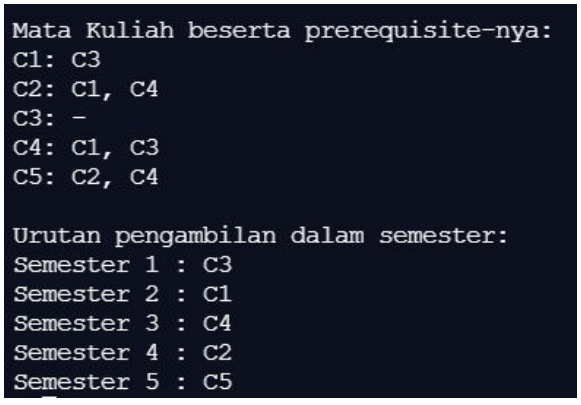
4. Beberapa contoh dan tangkapan layar pada saat program dijalankan.
    a. Ketika program pertama kali dijalankan.



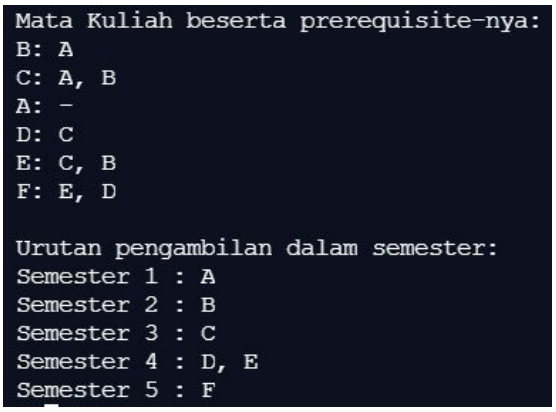
Gambar 4.1. Program saat pertama kali dijalankan.

    b. Beberapa contoh hasil dari masukan.

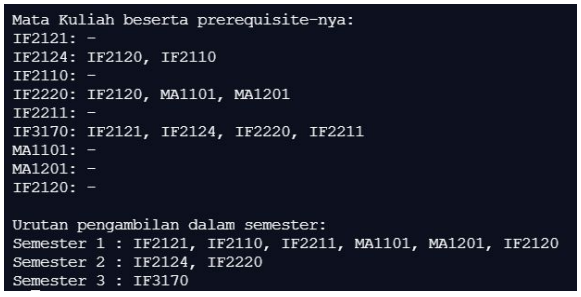
1.



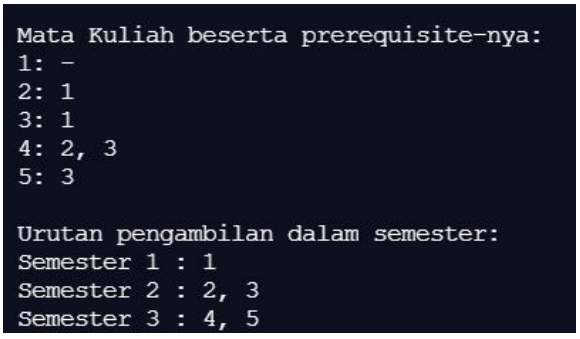
2.



3.



4.



5.

```
Mata Kuliah beserta prerequisite-nya
0: 4, 5
4: -
5: -
2: 5
1: 4, 3
3: 2

Urutan pengambilan dalam semester:
Semester 1 : 4, 5
Semester 2 : 0, 2
Semester 3 : 3
Semester 4 : 1
```

6.

```
Mata Kuliah beserta prerequisite-nya:
7: -
5: -
3: -
11: 7, 5
8: 3, 7
2: 11
9: 8, 11
10: 11, 3

Urutan pengambilan dalam semester:
Semester 1 : 7, 5, 3
Semester 2 : 11, 8
Semester 3 : 2, 9, 10
```

7.

```
Mata Kuliah beserta prerequisite-nya:
6: 5
12: 10
4: 2, 3
7: 4, 6
15: 11
5: 1
11: 10
2: 1
9: 8, 5
1: -
13: 10
3: 1
10: 1
8: 1
14: 11

Urutan pengambilan dalam semester:
Semester 1 : 1
Semester 2 : 5, 2, 3, 10, 8
Semester 3 : 6, 4, 12, 11, 13, 9
Semester 4 : 7, 15, 14
```

8.

```
Mata Kuliah beserta prerequisite-nya:
1: -
2: 1
3: 1
4: 2
5: 2, 6
6: 3
7: 4, 5, 6

Urutan pengambilan dalam semester:
Semester 1 : 1
Semester 2 : 2, 3
Semester 3 : 4, 6
Semester 4 : 5
Semester 5 : 7
```

5. Alamat unduh program.
   Program dapat diunduh melalui alamat berikut.
   https://github.com/raf555/course-prerequisite-sorter

6. Tabel Keberjalanan Program

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.	✓	

7. Kesimpulan

   Program dapat dijalankan tanpa ada kesalahan compile (syntax error / gagal compile), program juga dapat membaca fail dengan isi fail berupa format tertentu tanpa ada masalah. Program juga dapat menyelesaikan persoalan pengurutan pengambilan mata kuliah dengan prasyarat pengambilan tertentu yang dimodelkan dengan graf DAG yang diurutkan dengan pendekatan Topological Sort dengan Algoritma Decrease and Conquer. Luaran program sudah benar untuk semua kasus input.