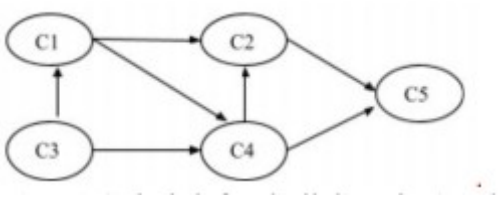
Laporan Tugas Kecli 2 IF2211 Strategi Algoritma

NIM : 13519019
Nama : Jason Stanley Yoman
Kelas : K-01

1. Algoritma *Topological Sort* dengan pendekatan *Decrease and Conquer*



Penjelasan di laporan akan mengacu pada penyelesaian masalah topological sort graph di atas

Kamus
Struktur data yang digunakan adalah sebuah graph yang direpresentasikan dengan list ketetanggaan. Jika graph pada masalah yang akan diselesaikan diubah menjadi list ketetanggaan, maka listnya akan menjadi seperti ini :

C1 → C4, C2
C2 → C5
C3 → C1, C4
C4 → C2, C5
C5 →

Selain itu, digunakan juga sebuah map yang menunjukkan derajat masuk tiap node. Yang menjadi key dari map adalah string yang menunjukkan kode kuliah sedangkan valuenya adalah integer yang menunjukkan jumlah derajat masuk. Jika graph pada masalah yang akan diselesaikan dibuat mapnya, maka map akan menjadi seperti ini :

C1 → 1
C2 → 2
C3 → 0
C4 → 2
C5 → 2

Untuk membuat program menjadi lebih modular, maka dibuatlah sebuah class graph. Graph memiliki atribut dan method sebagai berikut :

Attribute :
- adjList : list ketetanggaan
- inDegree : map derajat masuk

Method :
- construtctor yang menginisialisasi adjList kosong dan inDegree kosong
- addNode(a) : menambah node a pada graph
- addInDegree(a) : menambah derajat masuk dari node a
- addEdge(a,b) : menambah sisi yang berasal dari a dan berakhir di b
- topologicalSort : method topological sort yang akan dijelaskan di bagian setelah ini

- topologicalSortUtil : method pembantu topological sort yang akan dijelaskan di bagian setelah ini

Algoritma

Program akan dibagi menjadi beberapa bagian yang setiap bagiannya mengerjakan sebuah sub-task. Sub-task tersebut adalah:

A. Memproses file sehingga terbentuk graph

Langkah – langkah :
1. Program akan mengiterasi setiap baris dalam file txt. Setelah itu baris tersebut akan diolah. Karena format setiap baris dalam spesifikasi sudah pasti, kita tinggal mengambil dari karakter pertama dalam baris sampai karakter kedua terakhir dari baris. Dilakukan hal demikian karena karakter terakhir merupakan sebuah titik dan kita tidak memerlukan titik tersebut dalam pemrosesan baris. Setelah itu, potong baris tersebut menjadi sebuah array dengan delimiternya adalah koma. Berikut adalah contohnya :

Baris yang semula seperti ini : C5,C2,C4.
Akan diubah menjadi array : [“C5”, “C2”, “C4”]

2. Elemen pertama pada array pasti merupakan node awal dan elemen berikutnya adalah target dari node awal. Pada graph yang akan dikonstruksi, kita tinggal memanggil fungsi addNode dan addEdge saja.
Berikut adalah contohnya :
Array yang akan dijadikan graph : [“C5”, “C2”, “C4”]
Pemanggilan fungsi pada graph :
I. addNode(“C5”)
II. addEdge(“C5”, “C2”)
III. addEdge(“C5”, “C4”)

B. Topological Sort
topologicalSort()
topologicalSortUtil(adjList, inDegree, depth)

Dua fungsi di atas adalah 2 fungsi yang berperan dalam proses mencari mata kuliah apa yang bisa diambil di setiap semester. Fungsi topologicalSort akan memanggil fungsi topologicalSortUtil dengan memasukkan paramter adjList dan inDegree dengan attribute dalam objek graph masing-masing. Sedangkan parameter depth akan diisi dengan 1 yang menandakan semester dimulai dari 1.

Langkah – langkah :
1. Fungsi akan mengiterasi setiap inDegree dan mengecek apakah ada inDegree yang bernilai 0, jika ada masukkan ke sebuah array (sebut saja toDelete)
Pada masalah yang dibahas, array toDelete akan menjadi : [“C3”]
2. Jika panjang dari toDelete <= 0, maka tidak ada kuliah yang bisa diambil di semester itu. Namun, jika panjang toDelete > 0, maka kuliah yang terdapat pada toDelete adalah kuliah yang bisa diambil di semester itu dan cetak semuanya ke layar.
3. Setelah itu fungsi akan menghapus semua node pada graph yang nodenya berada pada toDelete dan mengurangi inDegree jika node pada toDelete bertetanggaan dengan node lainnya. Disini lah konsep decrease and conquer diterapkan. Awal mulanya terdapat banyak node dan kemudian dilakukan pengurangan node jika node sudah memenuhi kriteria untuk diambil pada semester tersebut.
AdjList setelah dilakukan penghapusan adalah sebagai berikut :

C1 → C4, C2
C2 → C5
C4 → C2, C5
C5 →

inDegree setelah dilakuka penghapusan adalah sebagai berikut :
C1 → 0
C2 → 2
C4 → 1
C5 → 2

4. Setelah itu, fungsi akan melakukan rekursi dengan paramter adjList dan inDegree yang sudah dikurangi dan parameter depth adalah depth sebelumnya ditambah 1

2. Source Code

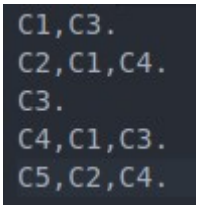
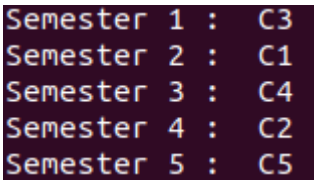
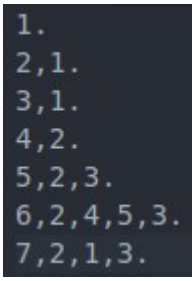
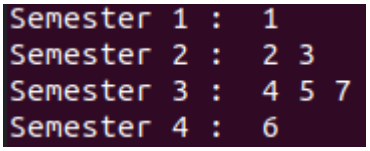
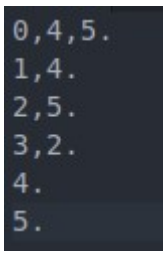
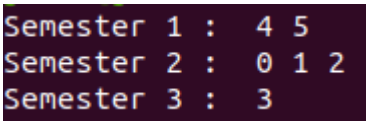
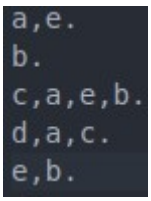
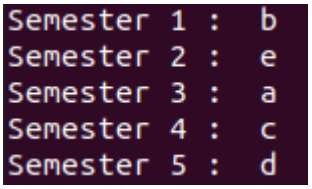
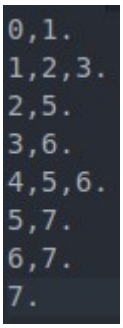
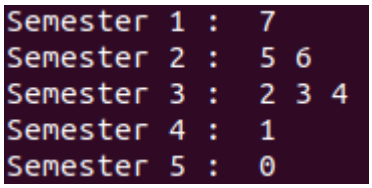
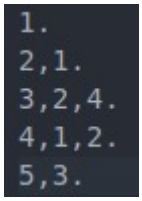
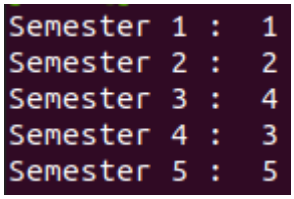
```
import os
import sys

class Graph () :
    def __init__ (self) :
        self.adjList = dict()
        self.inDegree = dict()
    def addEdge (self, start, end) :
        self.addNode(start)
        self.addNode(end)
        self.adjList[start].append(end)
        self.addInDegree(end)
    def addNode (self, node) :
        if node not in self.adjList:
            self.adjList[node] = []
            self.inDegree[node] = 0
    def addInDegree(self, node):
        if node not in self.inDegree:
            self.inDegree[node] = 0
        self.inDegree[node] += 1
    def printGraph(self):
        for key in self.adjList.keys():
            print(key + ": ")
            print(self.adjList[key])
    def printInDegree(self):
        for key in self.inDegree.keys():
            print(key + ": ")
            print(self.inDegree[key])
```

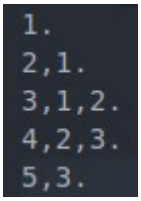
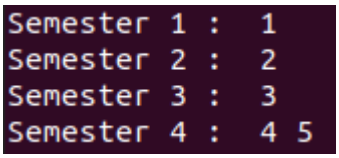
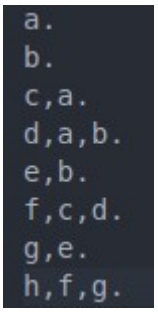
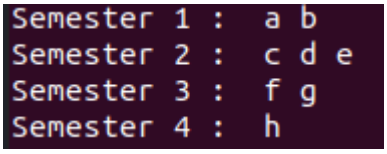
```
    def topologicalSort(self) :
        adjList = self.adjList.copy()
        inDegree = self.inDegree.copy()

        self.topologicalSortUtils(adjList, inDegree, 1)

    def topologicalSortUtils (self, adjList : dict, inDegree : dict, depth : int):
        toDelete = []
        for course in inDegree.keys():
            if inDegree[course] == 0:
                toDelete.append(course)
        if len(toDelete) > 0:
            print("Semester {} : ".format(depth), end=" ")
            for course in toDelete:
                print(course, end=" ")
                neighbourList = adjList[course]
                del inDegree[course]
                del adjList[course]

                for neighbour in neighbourList:
                    inDegree[neighbour] -= 1
            print()
            self.topologicalSortUtils(adjList, inDegree, depth + 1)
```

```
    def constructGraphFromFile(self, filename):
        with open(filename, "r") as file:
            for line in file:
                line = line.strip()
                line = line[:len(line) - 1]
                courseList = line.split(",")
                self.addNode(courseList[0])
                for course in courseList[1:]:
                    self.addEdge(course, courseList[0])
```

```
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage : python topological.py FILE_NAME")
    else:
        testFilePath = os.path.join(os.getcwd(), "../test", sys.argv[1] + ".txt")
        if os.path.exists(testFilePath) :
            graph = Graph()
            graph.constructGraphFromFile(testFilePath)
            graph.topologicalSort()

        else:
            print("Not a valid test file")
```

3. Screenshot Hasil Program

Input	Output
	
	
	
	
	
	

4. Alamat Drive

https://github.com/jasonstanleyyoman/Tucil2Stima

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	