LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

Penyusunan Rencana Kuliah dengan Topological Sort

(Penerapan Decrease and Conquer)

Nama : Afifah Fathimah Qur'ani

NIM : 13519183

Kelas : K-04

Dosen : Dr. Ir. Rinaldi, M.T.

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

A. Algoritma Topological Sort dan Kaitannya dengan Pendekatan Decrease and Conquer

Decrease and Conquer adalah metode perancangan algoritma dengan mereduksi persoalan menjadi beberapa upa-persoalan yang lebih kecil sehingga lebih mudah diproses, dan selanjutnya hanya memproses satu upa-persoalan saja. Dengan kata lain metode ini akan mengambil kesimpulan general dari potongan kecil persoalan, berbeda dengan metode *Divide and Conquer* yang menggabungkan seluruh kesimpulan dari persoalan kecil dan mengombinasikannya menjadi kesimpulan general.
Algoritma ini terdiri dari dua tahapan , yaitu

1. *Decrease*:
   Tahap pereduksian persoalan menjadi upa-persoalan yang lebih kecil. Banyaknya upa-persoalan yang dihasilkan bergantung kepada varian *Decrease and Conquer* itu sendiri, diantaranya
   a. *Decrease by a constant:*
      Ukuran instans persoalan direduksi menjadi sebesar konstanta yang sama pada setiap iterasi. Contoh algoritma yang menggunakan metode ini adalah *Selection Sort* dan *Insertion Sort.*
   b. *Decrease by a constant factor:*
      Ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi algoritma. Salahsatunya contoh algoritma dengan metode ini adalah *Binary Search.*
   c. *Decrease by a variable size:*
      Ukuran instans persoalan direduksi bervariasi pada setiap iterasi algoritma. Contoh algoritmanya diantara lain *Interpolation Search* dan *Finding Median.*
2. *Conquer*:
   Tahap pemrosesan salahsatu upa-persoalan secara rekursif.

Sedangkan *Topological Sort* adalah salahsatu algoritma yang menggunakan metode *Decrease and Conquer*, tepatnya *decrease by a constant 1*.

Persoalan yang dapat diselesaikan dengan *Topological Sort* adalah persoalan dengan model DAG (*Directed Acyclic Graph*) yaitu persoalan berupa grafik tanpa siklus tertutup. DAG ini banyak digunakan pada contoh-contoh persoalan di dunia nyata yang melibatkan batasan-batasan berupa prasyarat, misalnya pada compilers, pengambilan matakuliah yang memiliki perquisites, pengontrolan versi-versi dokumen atau arsip, dan lain-lain.

Topological Sort terdiri lagi beberapa jenis algoritma, diantaranya

1. *Kahn's Algorithm*
   Algoritma ini akan mencari terlebih dahulu simpul (*nodes*) pada *graph* yang tidak memiliki sisi masuk (*incoming edges*) dan memasukkan simpul-simpul tersebut kedalam sebuah himpunan. Kemudian dari simpul awal tadi akan dipilih satu simpul, dimasukkan kedalam list solusi, dan dilakukan penghapusan sisi keluar dari simpul tersebut, sehingga akan ditemukan simpul baru tanpa sisi masuk. Seterusnya dilakukan sehingga didapatkan list solusi lengkap.

2. *Depth-First Search*
   Algoritma ini akan melakukan looping pada setiap simpul di grafik, tanpa urutan yang ditentukan, dan akan berhenti ketika mencapai simpul yang telah dikunjungi sebelumnya atau ketika simpul tidak memiliki sisi keluar. Oleh karena itu, setiap simpul yang dimasukkan ke list solusi telah mempertimbangkan semua simpul lain yang bergantung pada simpul solusi.
   Karena setiap sisi dan simpul akan dikunjungi sekali, maka kompleksitas waktunya linear.
3. Dan lain-lain

B. Source Code

Berikut ini *source code* yang ditulis menggunakan bahasa pemrograman python. Kode dibawah ini menggunakan *Topological Sort* dengan *Depth-First Search.*

```
# Nama : Afifah Fathimah Qur'ani
# NIM : 13519183
# Kelas : K-04


from collections import defaultdict
import os
import sys


# KELAS
class Graph :
    def __init__ (self, countV): #constructor
        self.names = []
        self.graph = defaultdict(list)
        self.vertices = countV #jumlah simpul


    def addName(self, x, y): #re-assign nama node
        names[x] = y


    def addEdge(self, u, v):
        self.graph[u].append(v)


    def delEdge(self, x): #hapus edge antara node x node y
        for item in self.graph:
```

```
            if (x in self.graph[item]):
                self.graph[item].remove(x)

    def addNode(self, x): #tambah node
        self.names.append(x)
        self.vertices += 1

    def delNode(self):#hapus node dan seluruh edge yg terhubung
        count = 0
        for i in printed:
            if i in self.graph:
                del self.graph[i]
                count += 1
            self.delEdge(i)
        self.vertices -= count

    def topologicalSort(self):
        result = [] #inisialisasi stack
    keys = [*g.graph]
        values = [*g.graph.values()]

        # Key tanpa values (matakuliah tanpa prereq)
        for list in values:
            for item in list:
                if (item not in g.graph.keys() and item not in
result):
                    result.append(item)
                    self.vertices -= 1

        checked = [0 for i in range(self.vertices)] #set nodes
false (belum dilalui)

        for i in range(len(keys)):
            if checked[i] == 0:
```

```
                self.recursiveTopological(keys, values, i, chec
ked, result)


        return (result)


    def recursiveTopological(self, keys, values, i, checked, re
sult):
        checked[i] = 1


        for j in [*g.graph.values()][i]:
            if (j not in keys): # jika mata kuliah tidak ada pr
ereq
                continue
            if checked[keys.index(j)] == 0:
                self.recursiveTopological(keys, values, keys.in
dex(j), checked, result)


        result.append(keys[i])


#PROGRAM UTAMA


# Membuka file txt
filename = input("Masukkan nama file: ")
file = open(os.path.join(sys.path[0], filename), "r")


# Membaca file per line
lines = file.read().splitlines()


# Inisialisasi graph
countV = 0
for i in lines: # Hitung jumlah vertices (asumsi jmlh lines = j
ml vertices)
    if i:
        countV += 1
g = Graph(countV)
```

```
# Meng-assign isi line menjadi atribut graph
for i in range(len(lines)):
    lines[i] = lines[i].split(', ') # Membagi isi line menjadi
list of string
    lines[i][-1] = lines[i][-1].replace('.', '')


# Buat edges
for i in range(len(lines)):
    for j in range(1, len(lines[i])):
        g.addEdge(lines[i][0], lines[i][j])


# Panggil topological sort
result = g.topologicalSort()


# Print hasil
printed = []
for i in range(8):
    print("Semester", i+1, ":")
    for j in result:
        if (j not in g.graph): # matakuliah tanpa prereq
            printed.append(j)
            print(j)
        elif (i > 0):
            # cek apakah matakuliah masih ada di keys
            check = (any(item in g.graph for item in g.graph[j]
))
            if (check==False):
                printed.append(j)
                print(j)
            else:
                continue


    result = [x for x in result if x not in printed]
    g.delNode() # hapus key yang sudah di print
```

C. Tangkapan Layar

Input	Output
C1, C3. C2, C1, C4. C3. C4, C1, C3. C5, C2, C4.	Masukkan nama file: graph1.txt Semester 1 : C3 Semester 2 : C1 Semester 3 : C4 Semester 4 : C2 Semester 5 : C5 Semester 6 : Semester 7 : Semester 8 : PS C:\Users\Lenovo\Desktop\C>
MA1101. FI1101. KI1102. MA1201, MA1101. FI1201, FI1101. KI1202, KI1102. EL1200, FI1101, MA1101. EL2001, EL1200. KU1072. EL2002, EL1200, EL2003. EL2005, EL2001. EL2008, KU1072, EL2003. EL2006, FI1201. EL3012, KI1202, EL2006. EL3014, EL2008, EL2005. EL3017, EL2001. EL3011, EL2002. EL3013, EL2005. EL3009, EL2005. EL2003, MA1201. EL2004, MA1201. EL3016, EL2004, EL2007. EL3010, EL2007. EL3015, EL2007. MA2072, MA1201. MA2074, MA2072. EL2007, MA2074.	Masukkan nama file: matkul.txt Semester 1 : MA1101 FI1101 KI1102 KU1072 Semester 2 : MA1201 FI1201 KI1202 EL1200 Semester 3 : EL2001 EL2003 EL2006 EL2004 MA2072 Semester 4 : EL2002 EL2005 EL2008 EL3012 EL3017 MA2074 Semester 5 : EL3014 EL3011 EL3013 EL3009 EL2007 Semester 6 : EL3016 EL3010 EL3015 Semester 7 : Semester 8 : PS C:\Users\Lenovo\Desktop\C>

```
MA1101.
FI1101.
KI1102.
MA1201, MA1101.
FI1201, FI1101.
KI1202, KI1102.
KU1102.
AS2111.
AS2102.
AS2103.
AS2104, MA1201.
AS2112, KU1102.
FI2102, MA1201, FI1201.
AS2211.
AS2213, AS2111.
AS2202, AS2102, AS2111, AS2103.
AS2204, AS2104.
AS2212.
FI2202, FI1201.
AS3112, FI2202.
AS3111, FI2102.
AS3101, AS2202.
AS3113, AS2213.
AS3105, AS2213, AS2211, AS2111.
AS3211, FI2102, AS2204.
AS3201, AS2213.
AS3202.
AS3204, AS2204, AS3112.
```

```
Masukkan nama file: graph3.txt
Semester 1 :
MA1101
FI1101
KI1102
KU1102
AS2111
AS2102
AS2103
AS2211
Semester 2 :
MA1201
FI1201
KI1202
AS2112
AS2213
AS2202
Semester 3 :
AS2104
FI2102
FI2202
AS3101
AS3113
AS3105
AS3201
Semester 4 :
AS2204
AS3112
AS3111
Semester 5 :
AS3211
AS3204
Semester 6 :
Semester 7 :
Semester 8 :
PS C:\Users\Lenovo\Desktop\C>
```

```
MA1101.
FI1101.
KI1101.
MA1201, MA1101.
FI1201, FI1101.
KI1201, KI1101.
KU1102.
TI2101.
TI2105, MA1201.
TI2103, MA1201, TI2101.
TI2106, KU1102.
MAS2140.
MS2050.
MA2021.
TI2201, TI2106, TI2105.
TI2005, TI2101, TI2103.
TI2202, TI2101.
TI2204, MS2050.
TI2002, TI2204.
MA2031, MA1201.
MR2003, FI1201.
TI2001, MA2021.
TI3004, MA1201.
TI3102, TI2001.
TI3103, TI2201, TI2001.
TI3001, TI2204, TI2201, TI2001.
TI3104, TI2204, TI2201.
TI3201, TI3002, TI2202.
TI3202, TI3002, TI2001.
TI3203, TI3103, TI2001.
TI3204, TI2101, TI2202.
```

```
Masukkan nama file: graph4.txt
Semester 1 :
MA1101
FI1101
KI1101
TI2101
TI2106
MS2050
MA2021
TI3002
Semester 2 :
MA1201
FI1201
KI1201
TI2201
TI2202
TI2204
TI2001
Semester 3 :
TI2103
TI2002
MA2031
MR2003
TI3004
TI3102
TI3103
TI3001
TI3104
TI3201
TI3202
TI3204
Semester 4 :
TI2005
TI3203
Semester 5 :
Semester 6 :
Semester 7 :
```

```
A.
B, A.
C, A, B.
D, A, B.
E, C, D.
```

```
Masukkan nama file: graph5.txt
Semester 1 :
A
Semester 2 :
B
Semester 3 :
C
D
Semester 4 :
E
Semester 5 :
Semester 6 :
Semester 7 :
Semester 8 :
PS C:\Users\Lenovo\Desktop\C>
```

P. Q, P, R, S. R, P, S. S.	Semester 1 : P S Semester 2 : R Semester 3 : Q Semester 4 : Semester 5 : Semester 6 : Semester 7 : Semester 8 : PS C:\Users\Lenovo\Desktop\C>
A. B, A. C, B. D, A. E, D. F, C, E.	Semester 1 : A Semester 2 : B D Semester 3 : C E Semester 4 : F Semester 5 : Semester 6 : Semester 7 : Semester 8 : PS C:\Users\Lenovo\Desktop\C>
142. 143, 142. 370, 143. 378, 143. 341, 143. 321, 143. 326, 321, 143. 322, 321, 143. 401, 378, 341, 322, 326. 421, 326, 322.	Masukkan nama file: grap Semester 1 : 142 Semester 2 : 143 Semester 3 : 370 378 341 321 Semester 4 : 326 322 Semester 5 : 401 421 Semester 6 : Semester 7 : Semester 8 : PS C:\Users\Lenovo\Deskt

D. Alamat Kode Sumber Program

Alamat Github : https://github.com/afifahfq/Tucil2-Stima-13519183

E. Poin Penilaian

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	✓	

F. Referensi

1. Algoritma Decrease and Conquer (Versi baru 2021) diakses dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm
2. Breadth First Search (BFS) dan Depth First Search (DFS) diakses dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm
3. Kahn, 1962. Topological sorting of large networks. Communications of the ACM, Washington D.C. Diakses dari https://dl.acm.org/doi/10.1145/368996.369025
4. NUS Course Material, diakses dari https://www.comp.nus.edu.sg/~ooiwt/tp/cs1102-0203-s1/lecture/13-algo.pdf