TOPOLOGICAL SORT DENGAN ALGORITMA DECREASE AND CONQUER

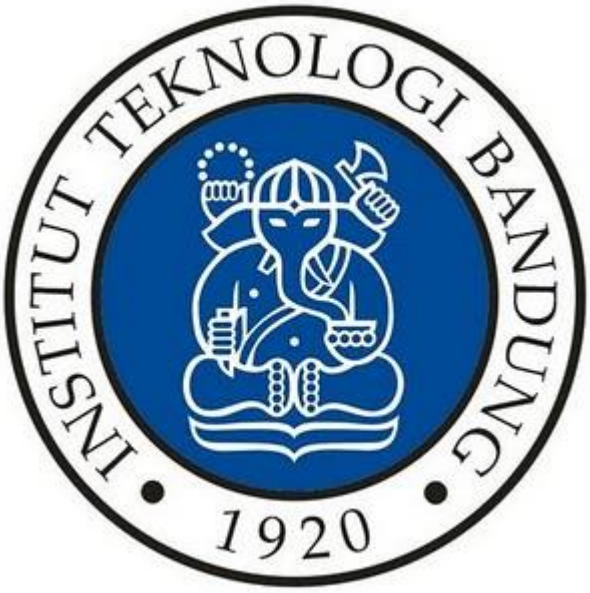
TUGAS KECIL

Diajukan sebagai tugas dari mata kuliah Strategi Algoritma di jurusan Informatika Institut Teknologi Bandung

Oleh :

Fakhri Nail Wibowo

135190



PROGRAM STUDI INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

A. Algoritma Decrease and Conquer

Decrease and conquer adalah metode perancangan algoritma dengan mereduksi persoalan menjadi dua *sub-problem*, decrease and conquer memiliki kemiripan dengan divide and conquer. Divide and conquer membagi persoalan menjadi dua *sub-problem* dan memproses keduanya lalu digabungkan, sedangkan decrease and conquer hanya memproses salah satu saja.

Biasanya persoalan akan diproses secara rekursif, namun dalam tugas ini penulis menyelesaikan persoalan dengan menggunakan *looping*.

Terdapat tiga macam dari decrease dan conquer. Pertama, decrease by a constant, ukuran persoalan direduksi sebesar konstanta yang tetap di setiap iterasi. Kedua, decrease by a constant factor, ukuran persoalan direduksi sebesar faktor yang telah ditetapkan di awal iterasi. Ketiga, decrease by a variable size, ukuran persoalan direduksi dengan jumlah yang tidak tetap pada setiap iterasi.

B. Algoritma Topological Sort

Topological sort adalah metode untuk mengurutkan graf, khususnya graf bertipe DAG atau Directed Acyclic Graph. Topological sort akan mengurutkan graf mulai dari *node* yang tidak memiliki derajat masuk sama sekali. Hasil dari sort tidak unik.

Representasi graf berupa List dengan anggota indeks terkecil sebagai node dan elemen setelahnya node yang masuk, jumlah derajat masuk adalah panjang List dikurangi satu.

Algoritma topological sort telah disesuaikan untuk persoalan ini:

1. Cari semua node yang memiliki derajat masuk bernilai nol, jika tidak ada maka bukan DAG
2. Node-node berderajat nol akan disimpan dalam satu struktur data untuk keperluan output, banyaknya node tersebut juga akan disimpan
3. Ambil satu node berderajat nol dan hapus node tersebut dari semua node yang berkaitan termasuk dirinya sendiri
4. Masukkan node yang dihapus ke dalam kelompok solusi
5. Jika jumlah solusi belum sama dengan jumlah node berderajat nol yang sudah dihitung pada langkah 2 maka ulangi proses dari langkah 3
6. Jika jumlah solusi sudah sama, ulangi dari langkah 1
7. Jika semua node sudah selesai diproses dan graf kosong keluarkan hasil yang ada

Algoritma disesuaikan untuk tidak menghapus node yang berderajat nol kecuali jika tersimpan juga di struktur data yang dibuat pada langkah ke-2. Hal ini

untuk menghindari kesalahan urutan dalam output

C. Source Code

```
# print screen awal
def printStartScreen():
    print('                    _                      _   ')
    print('                   | |                    | |  ')
    print(' _ __  _   _  ___ | |__    ___  ___  _ __| |_ ')
    print("| '_ \| | | |/ _ \| '_ \  / __|/ _ \| '__| __|")
    print('| | | | |_| | (_) | | | | \__ \ (_) | |  | |_ ')
    print('|_| |_|\__, |\___/|_| |_| |___/\___/|_|   \__|')
    print('        __/ |                                 ')
    print('       |___/                                  ')
    print()

# cek graph siklik
def isGraphCyclic(graph):
    for node in graph:
        # cek ada in degree 0
        if len(node) == 1:
            return False

    return True

# cek prereq dari course
def courseHaveNoPrereqs(course):
    return len(course) == 1

# hitung jumlah course tanpa prereq dalam satu semester
def countDegreeZeroInSemester():
    degreeZeroInSameSemester = 0
    for course in listOfCoursesAndPrereqs:
        if courseHaveNoPrereqs(course):
                degreeZeroInSameSemester += 1
    return degreeZeroInSameSemester

# simpan course tanpa prereq dalam satu semester
def storeDegreeZeroInSemester():
    sameSemester = []
    for course in listOfCoursesAndPrereqs:
        if courseHaveNoPrereqs(course):
                sameSemester.append(course)

    return sameSemester
```

```
# hapus course tanpa prereq
def removeDegreeZeroCourse():
    tempCourse = ''.join(courseNode)
    solution.append(tempCourse)
    for courses in listOfCoursesAndPrereqs:
        for course in courses:
            if course == tempCourse:
                courses.remove(tempCourse)

# print solusi
def printSolution(solution):
    courseIndex = 0
    semester = 1
    for num in courseCountPerSemester:
        print('Semester ' + str(semester) + ': ')
        for i in range(num):
            if i != num-1:
                print(solution[courseIndex] + ', ', end='')
            else:
                print(solution[courseIndex])
            courseIndex += 1
        semester += 1


filename = input('Masukkan nama file yang ingin diproses (tanpa .txt) :
 ')
listOfCoursesAndPrereqs = []

# setup
with open('testfiles/' + filename + '.txt') as file:
    removeChars = ',.'
    for line in file:
        for char in removeChars:
            line = line.replace(char, '')

        listOfCoursesAndPrereqs.append((line.strip()).split())

file.close()

totalCourses = len(listOfCoursesAndPrereqs)
solution = []
courseCountPerSemester = []
coursesPerSemester = []
```

```
for i in range(len(listOfCoursesAndPrereqs)):
    if isGraphCyclic(listOfCoursesAndPrereqs):
        break
    for courseNode in listOfCoursesAndPrereqs:
        if isGraphCyclic(listOfCoursesAndPrereqs):
            break

        if courseCountPerSemester == [] or sum(courseCountPerSemester)
== len(solution):
            degreeZeroCountInSameSemester = countDegreeZeroInSemester()
            degreeZeroCoursesInSameSemester = storeDegreeZeroInSemester
()
            courseCountPerSemester.append(degreeZeroCountInSameSemester
)

        if courseHaveNoPrereqs(courseNode) and courseNode in degreeZero
CoursesInSameSemester:
            removeDegreeZeroCourse()

printSolution(solution)
```

D. Screenshot

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test1
Semester 1:
C3
Semester 2:
C1
Semester 3:
C4
Semester 4:
C2
Semester 5:
C5
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test2
Semester 1:
bentar
Semester 2:
lagi
Semester 3:
ujian
Semester 4:
tengah
Semester 5:
semester
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test3
Semester 1:
Untuk
Semester 2:
Tuhan, Bangsa, dan, Almamater
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test4
Semester 1:
MA1101, FI1101, KU1001, KU1102, KU1011, KU1024, IF1210, KU1202, IF2121, IF2110, IF2120, IF2124, IF2130, IF2211, IF2230, IF2240, IF2250, IF3140, IF3280, IF4091, KU2071, KU206X, A
S2005
Semester 2:
MA1201, FI1201, EL1200, IF2123, IF2210, IF3130, IF3141, IF3150, IF3151, IF3210, IF4090, IF4092
Semester 3:
IF2220, IF3110, IF3230, IF3250, IF3260
Semester 4:
IF3170
Semester 5:
IF3270
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test5
Semester 1:
MA1101, FI1101, KU1001, KU1102, KU1011, KU1024
Semester 2:
MA1201, FI1201, IF1210, KU1202, KI1002, EL1200
Semester 3:
IF2121, IF2110, IF2120, IF2124, IF2123, IF2130
Semester 4:
IF2210, IF2211, IF2220, IF2230, IF2240, IF2250
Semester 5:
IF3170, IF3110, IF3130, IF3141, IF3150, IF3140, IF3151
Semester 6:
IF3210, IF3270, IF3230, IF3250, IF3260, IF3280
Semester 7:
IF4090, IF4091
Semester 8:
IF4092
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test6
Semester 1:
1
Semester 2:
2
Semester 3:
3
Semester 4:
4
Semester 5:
5
Semester 6:
6
Semester 7:
7
Semester 8:
8
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test7
Semester 1:
Honor, is, dead
Semester 2:
But
Semester 3:
I'll, see, what
Semester 4:
I, can, do
```

```
$ python tucil2_13519035.py
Masukkan nama file yang ingin diproses (tanpa .txt) : test8
Semester 1:
Kalkulus, Fisika
Semester 2:
Matdis
Semester 3:
Kriptografi, TBFO, Stima
```

E. Alamat Kode Program

fakhrinail/TucilStima2 (github.com)

F. Cek List

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	