**TOPOLOGICAL SORT DENGAN ALGORITMA DECREASE AND CONQUER**

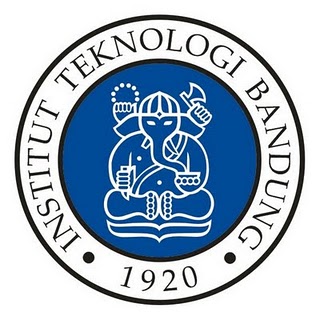
TUGAS KECIL

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**

1. **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.

1. **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

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

1. **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()

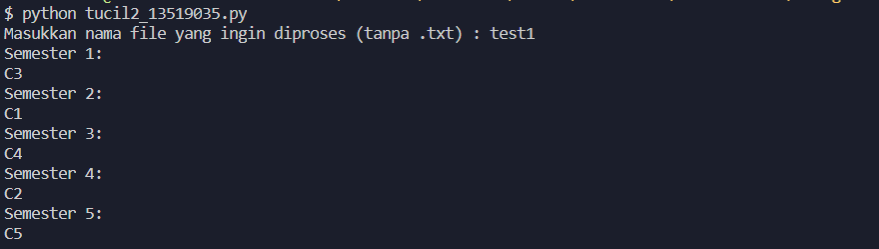
            courseCountPerSemester.append(degreeZeroCountInSameSemester)

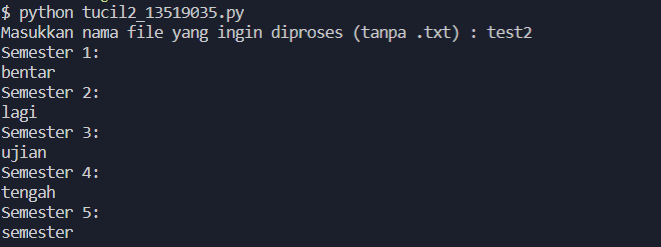
*if* courseHaveNoPrereqs(courseNode) and courseNode in degreeZeroCoursesInSameSemester:

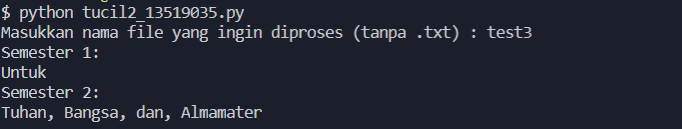
            removeDegreeZeroCourse()

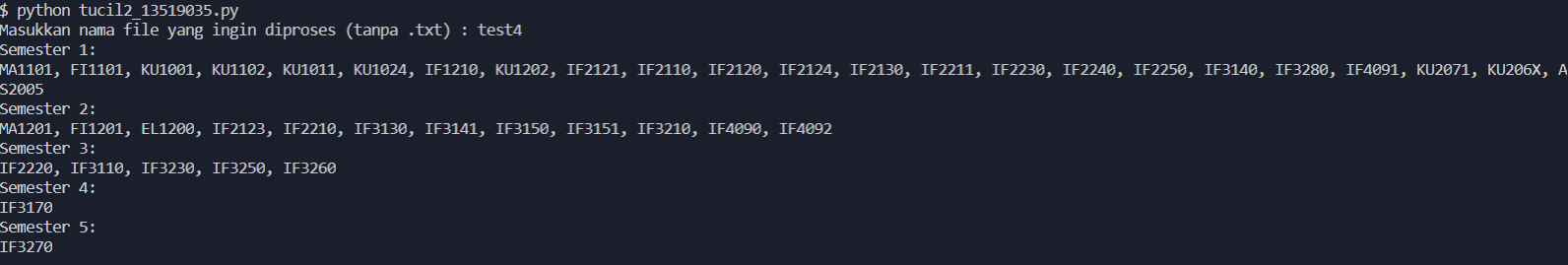
printSolution(solution)

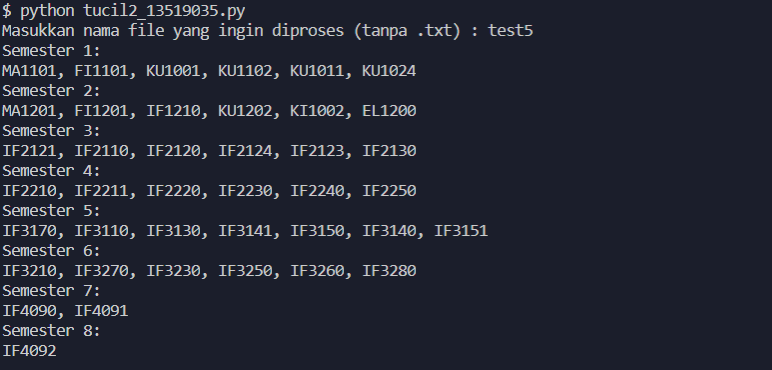
1. **Screenshot**

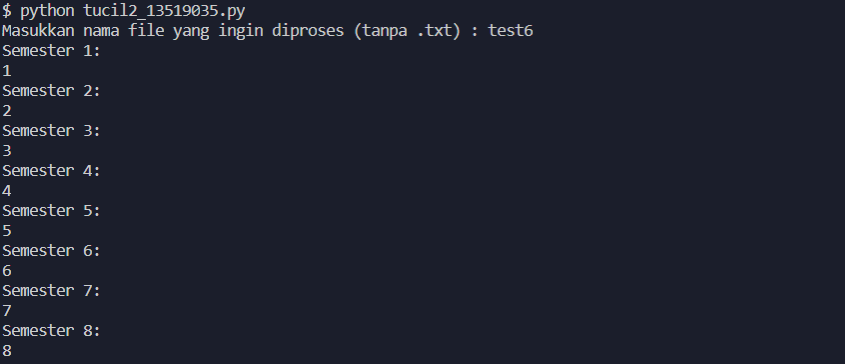


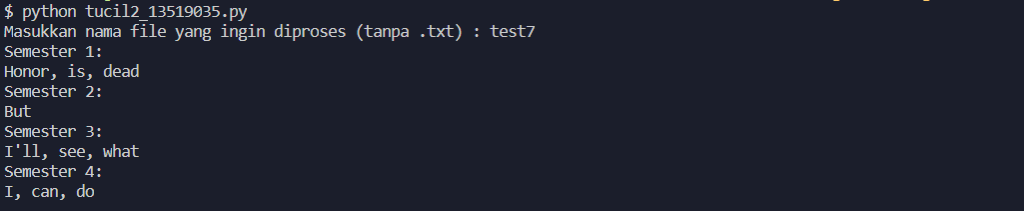


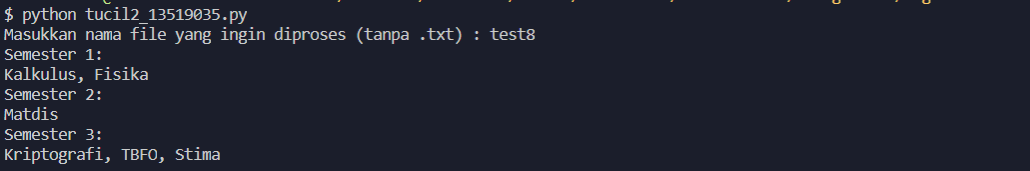












1. **Alamat Kode Program**
2. **Cek List**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat menerima berkas input dan menuliskan output | V |  |
| 1. Luaran sudah benar untuk semua kasus input | V |  |