LAPORAN TUGAS KECIL II

ALGORITMA *TOPOLOGICAL SORT* UNTUK MASALAH PENGAMBILAN KELAS DENGAN PENERAPAN *DECREASE AND CONQUER*

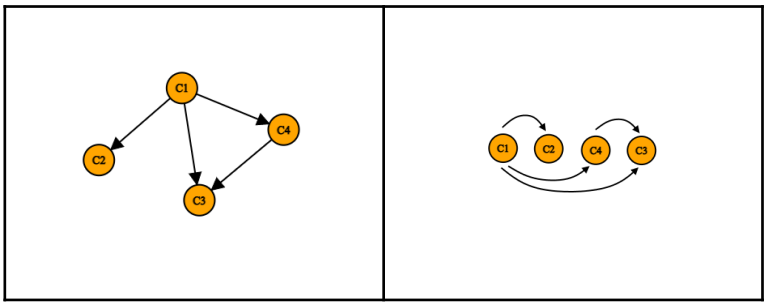
IF2211 STRATEGI ALGORITMA

Disusun Oleh:
James Chandra 13519078

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

1. ALGORITMA *TOPOLOGICAL SORT* DAN KAITANNYA DENGAN *DECREASE AND CONQUER*

Topological sorting, juga dikenal sebagai *top sort*, adalah algoritma pengurutan topologis terhadap suatu graf agar semua simpul dari graf tersebut terurut secara linier. Perhatikan bahwa permasalahan yang dapat diselesaikan algoritma ini harus bisa direpresentasikan sebagai graf berarah asiklik (pengurutan pengambilan kelas dengan prerekuisit berantai, dependency program berurut, instruksi assembly). Sebagai ilustrasi dari definisi tersebut, secara *layman*, topological sort dapat dijelaskan sebagai pengurutan simpul graf dalam garis lurus, dan semua garis berarah ke kanan dan tidak ada satupun yang sebaliknya.



Ilustrasi 1. Contoh Sebelum dan Setelah Pengurutan Topologis

Kaitan pendekatan *topological sort* ini dengan konsep *decrease and conquer* adalah faktanya bahwa pengurutan dilakukan dengan mengambil simpul dengan derajat masuk 0 (bisa lebih dari satu) kemudian menghilangkan simpul tersebut (pada *decrease and conquer* ini disebut *decrease by variable size* karena bisa satu atau lebih dari satu simpul yang dihilangkan) lalu setelah itu akan dimasukan array hasil, lalu dicetak (*conquer*).

Pada tugas kecil II IF2211 mata kuliah Strategi Algoritma ini, akan digunakan sebuah algoritma *topologiical sorting* untuk mencari penyelesaian permasalahan urutan pengambilan kelas paling optimal dengan menerapkan *decrease and conquer*, dengan langkah-langkah sebagai berikut.

1. Lakukan inisialisasi terhadap dictionary yang akan digunakan sebagai *adjacency list* untuk menampung representasi graf berarah asiklik.
2. <opsional> Nyatakan sebuah senarai berisikan 8 angka romawi pertama untuk nanti digunakan untuk substitusi untuk pencetakan semester ke sekian.
3. Panggil fungsi openFile yang berfungsi membaca file baris per baris, menghilangkan koma dan titik, kemudian menambahkan mata kuliah pertama sebagai key dictionary, kemudian mata kuliah setelahnya sebagai value di dalam senarai.
4. <asumsi: sesuai dengan jawaban pada qna, maksimal semester yang menjadi output adalah 8 semester> Iterasi sebanyak 8 kali dengan iterator i.
    a. Inisialisasi variabel boolean yang mengecek apakah semua mata kuliah sudah diambil
    b. Apabila masih ada mata kuliah yang belum diambil, maka
        i. Inisialisasi array hasil kosong, dan print template output semester ke sekian
        ii. Jalankan fungsi takeClass yang melakukan append key dictionary terhadap array hasil kemudian menjadikan value dari key tersebut menjadi array berisikan elemen None

iii. Panggil pula fungsi deleteKeyValue untuk menghapus value dari key lain yang terkandung dalam array hasil

iv. Cetak hasil iterasi kini, tambahkan tanda titik (‘.’) apabila semua mata kuliah sudah diambil.

Prinsip utama adalah a. nyatakan persoalan dalam DAG dengan representasi senarai ketetanggaan, *indegree* sebagai prerekuisit mata kuliah, b. pilih simpul dengan *indegree* 0 lalu hilangkan dan masukan ke senarai hasil semester tersebut, c. hilangkan busur yang keluar dari simpul yang dihapus pada b, d. Ulangi langkah b dan c hingga tak ada lagi busur.

2. SOURCE CODE PROGRAM PYTHON

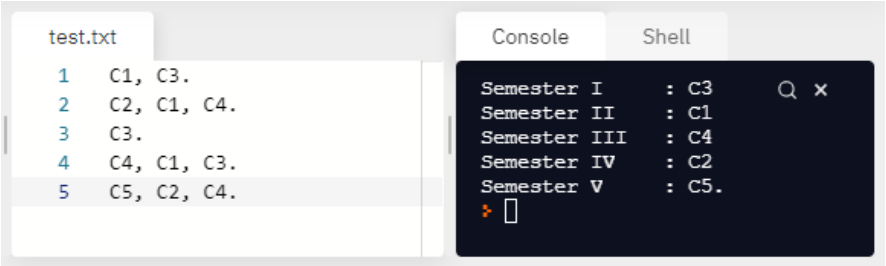
```
# TUGAS KECIL 2 STIMA
# JAMES CHANDRA / 13519078 / KELAS 02
# assumptions:
#     - penjadwalan akhir tidak melebihi 8 semester
#     - masukan berupa valid/directed acyclic graph


# DEFINISI FUNGSI

def openFile(inputFile, outputDict):
# read file line by line, calls function to add read data as graph vertex
  with open(inputFile) as f:
    for line in f:
      lineArray = line.strip(" .\n").split(", ")
      addVertex(lineArray, outputDict)

def addVertex(lineArray, outputDict):
# initializes key-value pair if first in one line, adds vertex if succeeding first class
  for i in range(len(lineArray)):
    if (i == 0):
      outputDict[lineArray[i]] = []
    else:
      outputDict[lineArray[0]].append(lineArray[i])

def takeClass(arrayResult, myDict):
# appends key to result array and sets dict key to empty / [None]
  for key in myDict:
    if (myDict[key] == []):
      arrayResult.append(key)
      myDict[key] = [None]

def deleteKeyValue(arrayResult, myDict):
# removes appropriate value of key (found in result array)
  for key in myDict:
    for keyDelete in arrayResult:
      if keyDelete in myDict[key]:
        myDict[key].remove(keyDelete)

def printConditional(arrayResult, myDict):
# prints according to dictionary/adjacency list condition
  if (list(myDict.values()).count([None]) == len(myDict)):
    print(*arrayResult, sep=", ", end=".\n")
  elif (len(arrayResult) == 0):
    return "error"
  else:
    print(*arrayResult, sep=", ")

# CONTINUE NEXT PAGE... >>
```

```
# MAIN PROGRAM

# initialize dictionary
myDict = {}

# roman numbers substitution
romanNumbers = ["I     : ", "II    : ", "III   : ", "IV    : ", "V     : ", "VI    : ", "VII   : ", "VIII  : "]

# call openfile function
openFile("test.txt", myDict)

# assumptions taken from qna: iterate for 8 semesters
for i in range(8):

  # determine if all values of each dictionary element is [none] or
  # all classes have been taken
  allNone = list(myDict.values()).count([None]) == len(myDict)

  if (not(allNone)):

    # initialize print array
    arrayResult = []

    print("Semester " + romanNumbers[i], end="")

    # call function to take class (append class to result array and set key's
    # value to [None] and call another function to remove the key's value from array)
    takeClass(arrayResult, myDict)
    deleteKeyValue(arrayResult, myDict)

    # print conditional, if returns error, then the graph is cyclic & invalid
    if (printConditional(arrayResult, myDict) == "error"):
      print("\n\nInput error: graph given is cyclic and hence is invalid")
      break
```

3. TANGKAPAN LAYAR INPUT DAN OUTPUT 8 CONTOH

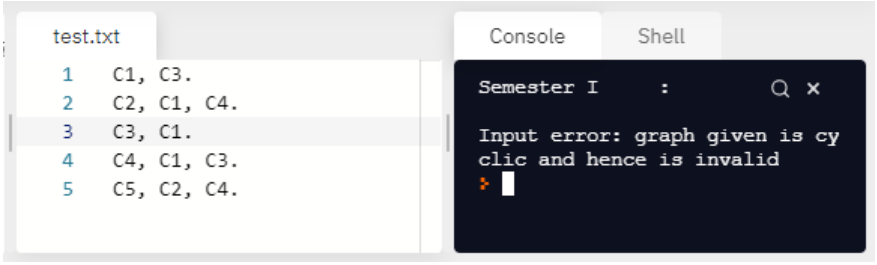
Perhatikan bahwa sang penguji HARUS mengganti test.txt secara manual dengan isi konten test1-test8.txt.

- Contoh dari spesifikasi



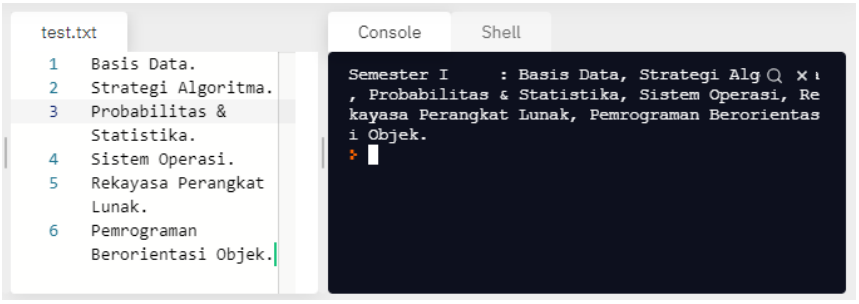
Gambar 1. Input dan output test case 1

- Contoh kedua (kasus graf siklik)



Gambar 2. Input dan output test case 2

- Contoh ketiga

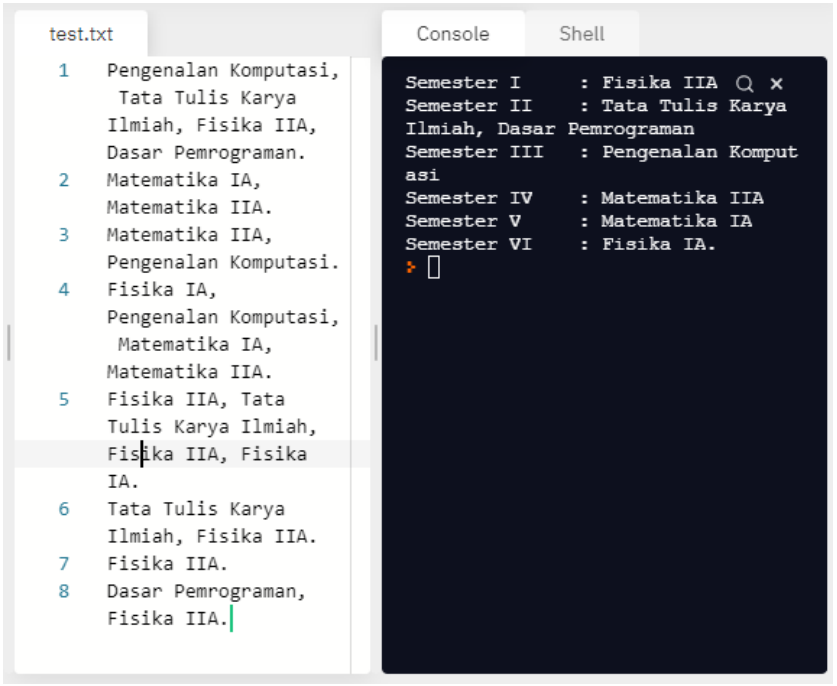


Gambar 3. Input dan output test case 3

- Contoh keempat

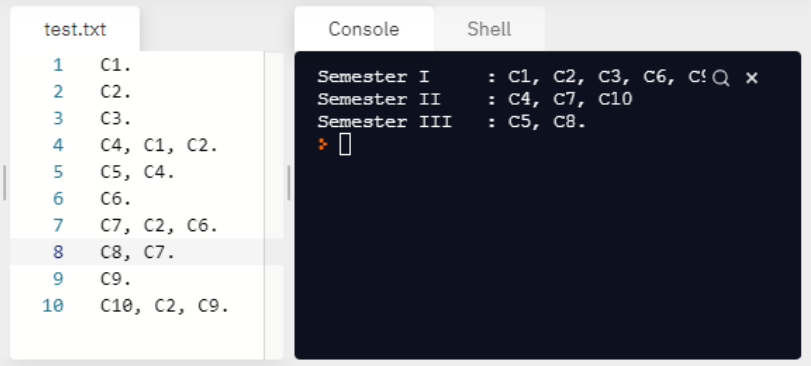
Gambar 4. Input dan output test case 4

- Contoh kelima (kasus nama mata kuliah dengan spasi)



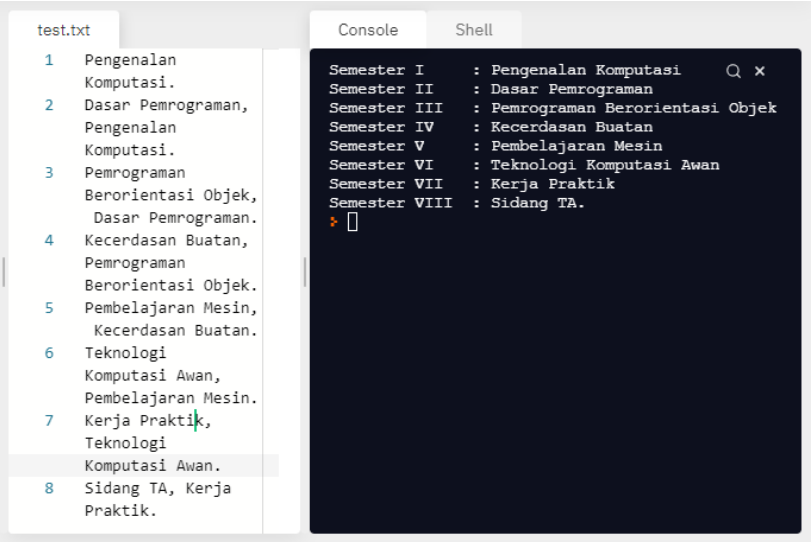
Gambar 5. Input dan output test case 5

- Contoh keenam



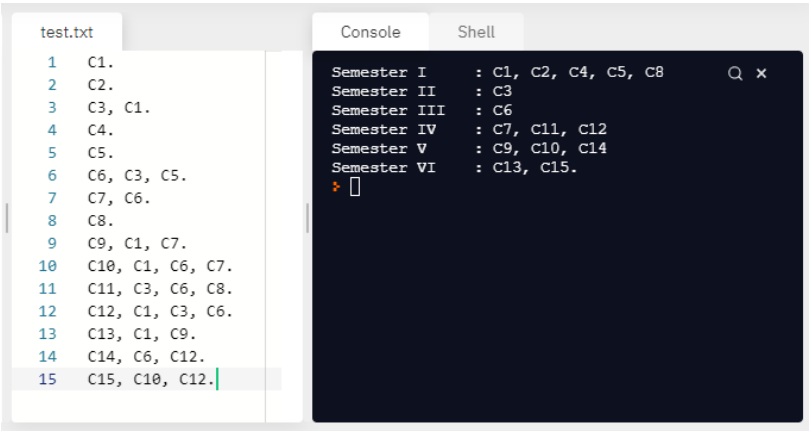
Gambar 6. Input dan output test case 6

- Contoh ketujuh (kasus mata kuliah beruntut)



Gambar 7. Input dan output test case 7

- Contoh kedelapan



Gambar 8. Input dan output test case 8

4. ALAMAT DRIVE YANG BERISI KODE PROGRAM

Program beserta dokumentasi berada pada tautan berikut.
Kelas K2 : https://drive.google.com/drive/folders/1aNQeD2kumHw5ztnWNS-rANsnMb-ef6Ej
Github : https://github.com/jamesteguh/Topological-Sort

5. CEK LIST 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.	✓	