**LAPORAN TUGAS KECIL II**

**PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT***

**(PENERAPAN *DECREASE AND CONQUER*)**

**IF2211 Strategi Algoritma**



Oleh:

**Alif Bhadrika Parikesit 13519186**

Kelas:

**K-04**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2021**

**BAB I**

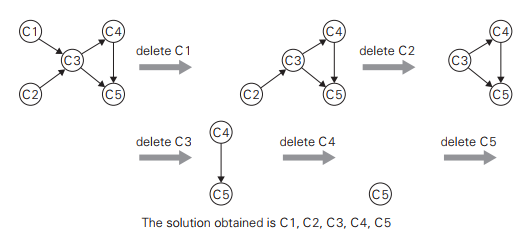
**ALGORITMA *DECREASE AND CONQUER***

Algoritma *decrease and conquer* merupakan salah satu metode penyelesaian permasalahan secara komputasional. Algoritma ini bekerja dengan cara mereduksi persoalan menjadi *sub-*persoalan yang lebih kecil, kemudian memproses satu sub-persoalan itu saja. Berbeda dengan *divide and conquer* yang memproses semua *sub-*persoalandan menggabungkan semua solusi dari setiap sub-persoalan menjadi solusi persoalan secara keseluruhan.

Tahapan pada algoritma *decrease and conquer* seperti namanya, yaitu *decrease* kemudian *conquer. Decrease* bekerja dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil kemudian *conquer,* menaklukan, yaitu memproses satu sub-persoalan secara rekursif.

Pada tugas kali ini, penulis akan menerapkan algoritma *decrease and conquer* pada permasalahan penyusunan rencana kuliah yang memiliki mata kuliah prasayarat dengan *topological sort*. Metode *topological sort* dapat diimplementasikan secara *decrease and conquer.* Adapun yang harus diperhatikan adalah apabila menggunakan pendekatan *topological sort*, maka harus dipastikan bahwa graf berarah tersebut harus tidak membentuk siklus *Directed Acyclic Graph* (DAG), karena apabila membentuk siklus maka tidak dapat menggunakan pendekatan *topological sort* ini. Kemudian, ada batasan yaitu mata kuliah tidak boleh diambil bersamaan dengan prasyaratnya.

Berikut ini adalah ilustrasi dari penerapa *topological sorting* pada sebuah graf.



**Gambar 1**. Ilustrasi *Topological Sorting*   
Sumber: Levitin, Anany. *Introduction to the design & analysis of algorithms*

Adapun implementasi *topological sort* dengan algoritma *decrease and conquer* yaitu sebagai berikut:

1. Representasikan daftar mata kuliah sebagai simpul pada graf berarah, dengan arah keluar menyatakan mata kuliah tersebut memiliki prasyarat yaitu simpul mata kuliah tujuan sisi berarah tersebut.
2. Identifikasi simpul pada graf yang memiliki derajat masuk = 0, dalam kasus mata kuliah, artinya mata kuliah tersebut tidak memiliki prasyarat.
3. Kemudian, hapus simpul tersebut pada graf dan sisi yang keluar dari simpul tersebut. Dalam kasus ini, misalkan simpul mata kuliah A berderajat masuk 0, maka simpul A akan dihapus dari graf dan seluruh mata kuliah yang tadinya memiliki prasyarat A sekarang tidak memiliki prasyarat A. Dengan demikian ukuran graph berkurang satu.
4. Masukkan simpul yang terhapus pada container yang berisi daftar semester beserta mata kuliah tiap semester. Jika sudah ada simpul yang mengisi pada semester terkait, maka mata kuliah dapat ditempatkan pada satu semester yang sama.
5. Ulangi langkah 2-4 untuk menemukan urutan mata kuliah per semesternya hingga graf kosong.

Alasan solusi ini merupakan pendekatan algoritma *decrease and conquer* adalah karena metode *topological sort* mengikuti kaidah varian *decrease and conquer* yang pertama yaitu *decrease by constant,* artinya ukuran dari *instance* berkurang secara konstan setiap pengulangannya. Ditunjukkan pada langkah ke-3, setiap simpul berderajat masuk = 0, maka simpul tersebut akan dihapus dari graf dan dimasukkan ke dalam solusi.

**BAB II**

***SOURCE CODE* PROGRAM *COURSE PREREQ SCHEDULING***

Source code program selengkapnya dapat diakses melalui tautan berikut:

<https://github.com/alifbhadrika/course-prereq-scheduling>

'''

February 27 2021, Alif Bhadrika Parikesit

Revisi 1

'''

def createGraphfromFile(inputFile):

'''

createGraphfromfile adalahh fungsi parse file kemudian

merepresentasikan data mata kuliah dari file sebagai

graf dengan berarah dengan struktur adjacency list

'''

with open('../test/'+inputFile,'r') as file:

graph = {}

for line in file:

line = (line.replace(',','').rstrip('.\n')).split()

graph[line[0]] = (line[1:])

return graph

def inDeg(graph):

vertex = list(graph.keys())

inDegree = dict.fromkeys(vertex,0)

for v in vertex:

inDegree[v] = len(graph[v])

return inDegree

def getStudyPlan(graph, sorted\_course, course\_per\_semester):

'''

getStudyPlan melakukan topsort pada graf mata kuliah dan menyeleksi

apakah mata kuliah dapat diletakkan pada satu semester yang sama

atau tidak

'''

smt = 1

same\_semester = [] # list course pada satu semester

while len(graph) != 0:

vertex = list(graph.keys()) # list simpul graf

inDegree = inDeg(graph) # dict derajat masuk tiap simpul graf

same\_semester.clear()

for v in vertex:

if inDegree[v] == 0:

sorted\_course.append(v)

# seleksi apakah boleh matkul v pada semester smt

# jika sudah ada 1 atau lebih matkul pada semester smt

# maka v ditempatkan pada semester yang sama

if len(same\_semester) != 0:

course\_per\_semester[smt].append(v)

else:

course\_per\_semester[smt] = [v]

#penghapusan sisi keluar dari simpul yang berderajat 0

for course,neighbors in graph.items():

for neighbor in neighbors:

if v == neighbor:

neighbors.remove(neighbor)

#penghapusan simpul berderajat masuk 0

graph.pop(v, None)

#penambahan v pada semester smt

same\_semester.append(v)

smt += 1

return

def print\_solution(course\_per\_semester):

'''

print\_solution mencetak solusi ke layar sesuai dengan

format SEMESTER : <nama\_course> tiap semesternya

'''

for semester,courses in course\_per\_semester.items():

print("SEMESTER {}: ".format(semester),end="")

for course in courses:

print(course," ",end="")

print()

if \_\_name\_\_ == '\_\_main\_\_':

inputFile = input("ENTER inputFilename.txt: ")

print()

sorted\_course = []

course\_per\_semester = {}

course\_graph = createGraphfromFile(inputFile)

getStudyPlan(course\_graph,sorted\_course,course\_per\_semester)

print("==== Sorted Courses List =====")

print(sorted\_course)

print()

print("======= Your Study Plan ======")

print\_solution(course\_per\_semester)

**BAB III**

**PENGUJIAN PROGRAM**

|  |  |  |
| --- | --- | --- |
| **NAMA FILE** | **INPUT** | **OUTPUT** |
| 1.txt |  |  |
| 2.txt |  |  |
| 3.txt |  |  |
| 4.txt |  |  |
| 5.txt |  |  |
| 6.txt |  |  |
| 7.txt |  |  |
| 8.txt |  |  |

**TABEL PENILAIAN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Poin** | **Kriteria** | **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** |  |

**DAFTAR PUSTAKA**

Levitin, Anany. 2003. *Introduction to the design & analysis of algorithms -3rd ed.* USA: Adison-Wesley.

Munir, Rinaldi. Slide kuliah “Algoritma *Decrease & Conquer*”. *http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf*. Diakses *online* pada 27 Februari 2021.

Munir, Rinaldi. Slide kuliah “Tugas Kecil 1 IF2211”. *https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf.* Diakses *online* pada 27 Februari 2021.