LAPORAN HASIL TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2020/2021
PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT*
(PENERAPAN *DECREASE AND CONQUER*)

Disusun oleh :

Billy Julius 13519094

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

DAFTAR ISI

BAB I
ALGORITMA *TOPOLOGICAL SORT* DAN KAITANNYA DENGAN PENDEKATAN *DECREASE AND CONQUER*

1.1 Algoritma *Decrease and Conquer*

Algoritma *decrease and conquer* merupakan salah satu algoritma yang sering digunakan dalam berbagai permasalahan pemrograman. Ciri khas dari algoritma ini adalah alurnya yang membagi persoalan menjadi dua bagian, dengan satu bagian akan "dibuang" dan permasalahan akan disederhanakan dengan hanya melibatkan satu bagian lainnya.

Berdasarkan jumlah bagian yang "dibuang", algoritma *decrease and conquer* dapat dibedakan menjadi tiga jenis, yaitu :

1. *Decrease by a constant*

   Algoritma ini memanfaatkan *decrease and conquer* dengan mengurangi ukuran instan persoalan sebesar suatu konstanta. Pada kebanyakan kasus, konstanta yang dipakai adalah 1 (satu).

2. *Decrease by a constant factor*

   Algoritma ini memanfaatkan *decrease and conquer* dengan mengurangi ukuran instan persoalan sebesar suatu faktor konstanta yang merepresentasikan proporsi instan yang "dibuang". Pada kebanyakan kasus, faktor konstanta yang dipakai adalah ½ (setengah).

3. *Decrease by a variable size*

   Algoritma ini memanfaatkan *decrease and conquer* dengan mengurangi ukuran instan persoalan sebesar suatu jumlah yang bersifat sembarang.

1.2 *Topological Sort* dan Kaitannya dengan *Decrease and Conquer*

Topological Sort adalah salah satu persoalan pemrograman yang melibatkan struktur data graf berarah. Secara sederhana, dari suatu graf berarah, akan diurutkan elemen-elemen yang dapat "dilepas" dari graf, yaitu yang memiliki *in*

degree (jumlah elemen yang sedang menunjuk ke dirinya) bernilai 0 (nol). Ketika suatu elemen “dilepas”, *in degree* dari elemen-elemen yang ditunjuknya akan diperbarui dengan menguranginya sebanyak 1. Kemudian, dicari kembali elemen yang memiliki *in degree* bernilai 0, elemen tersebut “dilepas”. Hal ini dilakukan berulang sampai semua elemen “terlepas” dari graf. *Topological Sort* akan diselesaikan dengan menghasilkan urutan “pelepasan” elemen yang mana dari awal sampai akhir.

Dalam kaitannya dengan *decrease and conquer*, *Topological Sort* dapat digolongkan ke dalam *decrease by constant*. Hal ini dikarenakan *Topological Sort* memenuhi ciri *decrease and conquer* dengen menyederhanakan masalah dengan mengurangi instansi persoalan. Adapun pengurangan yang dilakukan bersifat konstan, yaitu per 1 elemen. Persoalan akan menjadi sederhana dan semakin sederhana untuk tiap tahap *decrease* yang dilakukan.

1.3 Masalah *Topological Sort* pada Tugas Kecil 2 IF2211 Strategi Algoritma

Pada Tugas Kecil 2 IF2211 Strategi Algoritma, masalah yang diberikan memanfaatkan *Topological Sort* untuk membaca suatu file masukan yang berupa daftar nama mata kuliah dan daftar mata kuliah *prerequisite*-nya. Program yang akan dibuat dituntut untuk dapat melakukan *Topological Sort* yang akan menghasilkan daftar nama mata kuliah yang dapat diambil pada tiap semester.

Adapun beberapa asumsi dan batasan yang diberikan adalah :

- Data input dipastikan mendefinisikan DAG (*Directed Acylic Graph*), dengan arti graf berarah dan tidak ada *loop* di dalam graf.
- Tidak ada batasan jumlah mata kuliah yang dapat diambil pada tiap semester
- Jumlah semester yang ada dibatasi sampai semester kedelapan.

BAB II

KODE PROGRAM DALAM BAHASA *PYTHON*

2.1 *Class graph_elmt*

```python
# deklarasi kelas elemen graf
# graf akan didefinisikan sebagai array of elemen graf
class graph_elmt :
    def __init__(self,name,indeg,next_courses) :
        self.name = name
        self.indeg = indeg
        self.next_courses = next_courses
```

2.2 Prosedur *readFile*

```python
# prosedur membaca input file dan mengubahnya menjadi array of pasangan course dan prerequisites nya
def readFile(fileName) :
    f = open(fileName,"r")
    read_out = f.readlines()

    course_prereq = []
    for line in read_out :
        if (line != '\n') :
            course_prereq.append(line.replace('\n','').replace('.','').split(", "))

    return course_prereq
```

2.3 Prosedur *makeGraph*

```
# prosedur untuk membuat graf dari array of pasangan course dan prerequisitenya
def makeGraph(course_prereq) :
    # bagi menjadi array of course dan array of prerequisite
    names = []
    prereqs = []
    for course in course_prereq :
        names.append(course[0])
        curr_prereqs = []
        for i in range(len(course)-1) :
            curr_prereqs.append(course[i+1])
        prereqs.append(curr_prereqs)

    # secara sederhana, akan dilakukan "inverse", sehingga tiap elemen dari graf
    # akan mencatat course lanjutannya, bukan prerequisite nya
    graphList = []
    for name in names :
        curr_elmt = graph_elmt(name,0,[])
        graphList.append(curr_elmt)

    courseNum = 0
    for prereq in prereqs :
        for name in prereq :
            # tambahkan mata kuliah yang merupakan lanjutan
            graphIdx = searchName(graphList,name)
            graphList[graphIdx].next_courses.append(graphList[courseNum].name)

            # tambahkan in degree dari mata kuliah lanjutan
            graphList[courseNum].indeg += 1
        courseNum += 1

    return graphList
```

2.4 Fungsi *searchName*

```
# fungsi mencari indeks diberikan nama mata kuliah pada suatu graphList
# name dipastikan ada
def searchName(graphList,name) :
    i = 0
    while(graphList[i].name != name) :
        i += 1
    return i
```

2.5 Fungsi *allPrinted*

```
# fungsi mengecek apakah in degree dari tiap graph sudah 0
# true jika semua in degree 0
# false jika ada mata kuliah dengan in degree > 0
def allPrinted(graphList):
    for elmt in graphList :
        if elmt.indeg >= 0 :
            return False
    return True
```

2.6 Fungsi *zeroIndeg*

```
# fungsi mengembalikan list nama mata kuliah yang memiliki in degree 0
def zeroIndeg(graphList) :
    result = []
    for elmt in graphList :
        if elmt.indeg == 0 :
            result.append(elmt.name)
    return result
```

2.7 Program Utama

```
# main program
def main() :
    # baca input dan ubah jadi graf
    graphList = makeGraph(readFile("../test/" + sys.argv[1]))
    romanNum = ['I','II','III','IV','V','VI','VII','VIII']

    # cek memastikan ada data valid yang masuk
    if (graphList != []) :
        # cek memastikan tidak ada cycle dalam graf
        if (zeroIndeg(graphList) != []) :
            semesterCounter = 0
            # dilakukan print sampai semua diprint habis atau sudah mencapai semester 8
            # atau sudah tidak dapat mengambil mata kuliah lagi
            while (not allPrinted(graphList) and (semesterCounter <= 7) and zeroIndeg(graphList) != []) :
                print("Semester " + romanNum[semesterCounter] + " : ", end="")
                printables = zeroIndeg(graphList)
                # print mata kuliah dengan in degree 0
                for to_print in printables :
                    print(to_print, end="")
                    if (to_print != printables[len(printables)-1]) :
                        print(", ", end="")
                    else :
                        print()

                    # di next courses nya, in degree dikurangi 1
                    graphIdx = searchName(graphList,to_print)
                    curr_next_courses = graphList[graphIdx].next_courses
                    for name in curr_next_courses :
                        next_graphIdx = searchName(graphList,name)
                        graphList[next_graphIdx].indeg -= 1

                    # agar tidak di print lagi, yg sudah 0 dibuat jadi -1
                    graphList[graphIdx].indeg = -1

                semesterCounter += 1
        else :
            print("Tidak ada mata kuliah yang dapat diambil.")
    else :
        print("Tidak ada data yang valid!")
```

2.8 Cara *run*

Run dapat dilakukan dengan mengetikkan perintah pada direktori *src* dari *project* ini:

python3 13519094.py <testcase_name>.txt

dengan *testcase_name* diisi dengan nama *file* yang menjadi *data input*.

Adapun perintah dijalankan pada *Ubuntu 20.04.1*, untuk lingkungan sistem lainnya, mungkin perintah dapat berbeda.

BAB III
HASIL TANGKAPAN LAYAR *INPUT/OUTPUT*

Pada hasil tangkapan layar, penulisan perintah masih saat belum dilakukan perubahan nama pada file *main.py* menjadi *13519094.py*.

3.1 *Testcase* 1

```
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.
MA1201, MA1101.
FI1201, FI1101.
IF1210, KU1102.
KU1202, KU1102.
KI1002, KU1011.
EL1200, FI1101.

IF2121, IF1210, MA1101, MA1201.
IF2110, KU1102, IF1210.
IF2120, MA1201, MA1101.
IF2124, EL1200.
IF2123, MA1201.
IF2130, KU1202.

IF2210, IF2110.
IF2211, IF2110.
IF2220, MA1101, MA1201, IF2120.
IF2230, IF2130.
IF2240, IF2121, IF2120.
IF2250, KU1202, IF2110.

IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140, IF2240.
IF3151, IF2250.

IF3210, IF2110, IF2130, IF3110.
IF3270, IF2210, IF3170.
IF3230, IF3130.
IF3250, IF2250, IF3150.
IF3260, IF2123, IF2110, IF2130, IF3151.
IF3280, IF3151, IF3150.

IF4090, IF3280.
IF4091, IF3280.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 1.txt
Semester I : MA1101, FI1101, KU1001, KU1102, KU1011, KU1024
Semester II : MA1201, FI1201, IF1210, KU1202, KI1002, EL1200
Semester III : IF2121, IF2110, IF2120, IF2124, IF2123, IF2130
Semester IV : IF2210, IF2211, IF2220, IF2230, IF2240, IF2250
Semester V : IF3170, IF3110, IF3130, IF3141, IF3150, IF3140, IF3151
Semester VI : IF3210, IF3270, IF3230, IF3250, IF3260, IF3280
Semester VII : IF4090, IF4091
Semester VIII : IF4092
```

3.2 *Testcase 2*

```
MK1.
MK2, MK1, MK3.
MK3, MK4.
MK4.
MK5, MK6.
MK6, MK3.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 2.txt
Semester I : MK1, MK4
Semester II : MK3
Semester III : MK2, MK6
Semester IV : MK5
```

3.3 *Testcase 3*

```
C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.
C6.
C7.
C8, C3.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 3.txt
Semester I : C3, C6, C7
Semester II : C1, C8
Semester III : C4
Semester IV : C2
Semester V : C5
```

3.4 *Testcase 4*

```
DD1, DD2, DD3, DD4.
DD2, DD3.
DD3, DD6.
DD4, DD5.
DD5.
DD6.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 4.txt
Semester I : DD5, DD6
Semester II : DD3, DD4
Semester III : DD2
Semester IV : DD1
```

3.5 *Testcase* 5

```
EE1, EE6, EE7, EE8
EE2, EE3.
EE3, EE1.
EE4, EE1, EE2, EE3.
EE5, EE6, EE7, EE4.
EE6, EE7.
EE7.
EE8, EE7.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 5.txt
Semester I : EE7
Semester II : EE6, EE8
Semester III : EE1
Semester IV : EE3
Semester V : EE2
Semester VI : EE4
Semester VII : EE5
```

3.6 *Testcase* 6

```
FF1, FF2.
FF2, FF5.
FF3, FF1, FF2, FF5.
FF4, FF3, FF5.
FF5, FF7.
FF6.
FF7, FF6.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 6.txt
Semester I : FF6
Semester II : FF7
Semester III : FF5
Semester IV : FF2
Semester V : FF1
Semester VI : FF3
Semester VII : FF4
```

3.7 *Testcase* 7

```
GG1, GG2, GG3, GG4.
GG2, GG3, GG4.
GG3, GG5.
GG4, GG3.
GG5, GG6.
GG6.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 7.txt
Semester I : GG6
Semester II : GG5
Semester III : GG3
Semester IV : GG4
Semester V : GG2
Semester VI : GG1
```

3.8 *Testcase* 8

```
H1, H3, H5.
H2, H3, H4.
H3, H5.
H4, H3.
H5, H6.
H6.
H7, H5.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 8.txt
Semester I : H6
Semester II : H5
Semester III : H3, H7
Semester IV : H1, H4
Semester V : H2
```

3.9 *Testcase* 9

```

```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 9.txt
Tidak ada data yang valid!
```

3.10 *Testcase* 10

```
XX1, XX2.
XX2, XX1.
```

```
qymaque@DESKTOP-MDBJNH8:/mnt/c/users/hp/desktop/stima/src$ python3 main.py 10.txt
Tidak ada mata kuliah yang dapat diambil.
```

BAB IV
ALAMAT KODE SUMBER PROGRAM

Alamat kode sumber program dapat diakses melalui :

https://github.com/billyjuliux/Tucil2Stima.git

BAB V
TABEL PENILAIAN

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil *running*	√	
3. Program dapat menerima berkas i*nput* dan menuliskan *output*	√	
4. Luaran sudah benar untuk semua kasus *input*	√	