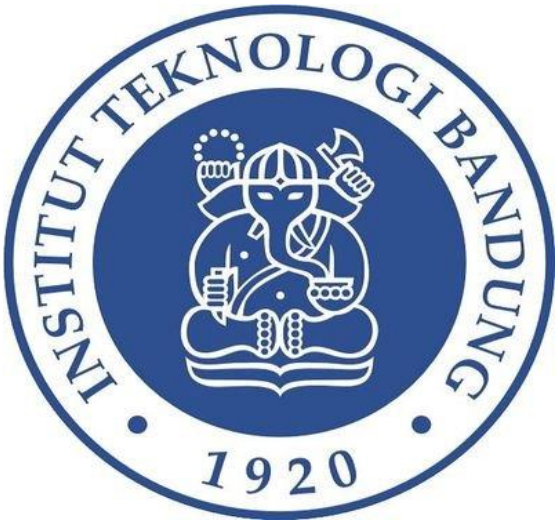
Tugas Kecil 2 IF2211 Strategi Algoritma
Penyusunan Rencana Kuliah dengan *Topological Sort*
(Penerapan *Decrease and Conquer*)



Disusun Oleh:

Richard Rivaldo
13519185
Kelas 04

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika

INSTITUT TEKNOLOGI BANDUNG
TAHUN AJARAN 2020/2021

1. Domain Permasalahan

Semua perguruan tinggi di dunia ini memberlakukan sistem mata kuliah yang memiliki *pre-requisites* dalam pengambilannya. Mata kuliah *pre-requisites* adalah mata kuliah yang harus diambil dan diselesaikan terlebih dahulu oleh mahasiswa ketika ingin mengambil suatu mata kuliah lain. Biasanya, perguruan-perguruan tinggi melarang pengambilan suatu mata kuliah dengan mata kuliah lain yang menjadi *pre-requisites* nya dalam semester yang sama, sehingga kedua mata kuliah ini harus diambil secara bertahap. Permasalahannya adalah persoalan mengurutkan tahapan-tahapan pengambilan mata kuliah yang banyak ini cukup sulit untuk dilakukan secara manual.

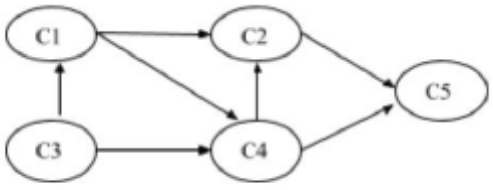
Oleh karena itu, diberikan suatu dokumen yang berisi daftar mata kuliah lengkap dengan *pre-requisites* nya, akan disusun rencana kuliah dengan menggunakan algoritma *Decrease and Conquer* melalui pendekatan *Topological Sorting*. Adapun sebuah mata kuliah bisa saja tidak memiliki prasyarat apapun dan diasumsikan bisa diambil di sembarang semester. Selain itu, juga diasumsikan jumlah maksimal dari semester yang boleh diambil, yaitu delapan semester. Berikut merupakan format dan contoh file yang menjadi input dalam program. Jika direpresentasikan ke dalam struktur graf, maka graf yang terbentuk haruslah berupa DAG atau *Directed Acyclic Graph*.

<kode_kuliah_1>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah prasyarat - 3>.
<kode_kuliah_2>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>.
<kode_kuliah_3>,<kode kuliah prasyarat - 1>, <kode kuliah prasyarat - 2>, <kode kuliah prasyarat - 3>, <kode kuliah prasyarat - 4>.
<kode_kuliah_4>.
.
.
.

Gambar 1. Format File Teks untuk Masukan Daftar Kuliah

C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.

Gambar 2. Contoh sebuah berkas masukan Daftar Kuliah



Gambar 3. DAG dari daftar kuliah pada Gambar 2

2. *Decrease and Conquer* dan *Topological Sort*

2.1 Algoritma *Decrease and Conquer*

Algoritma *Decrease and Conquer* mungkin kurang terkenal dibandingkan algoritma *Divine and Conquer* karena kedua istilah sering digabungkan menjadi *Divide and Conquer* saja. Padahal, *Decrease and Conquer* merupakan metode perancangan algoritma yang dilakukan dengan mereduksi persoalan menjadi upa-persoalan yang lebih kecil dan hanya memroses salah satunya saja. Hal ini berbeda dengan *Divide and Conquer* yang memroses dan menggabungkan semua upa-persoalan yang ada.

Pada prinsipnya, algoritma ini memiliki dua tahapan umum yang digunakan dalam pemecahan masalahnya. Pertama, algoritma akan melakukan tahap *Decrease*, yaitu melakukan reduksi atau pengurangan persoalan menjadi upa-persoalan yang lebih kecil (biasanya dua buah upa-persoalan). Tahapan selanjutnya adalah *Conquer*, yaitu memroses salah satu upa-persoalan dengan menggunakan prinsip rekursivitas. Dapat diperhatikan bahwa tidak ada tahap *Combine* seperti pada *Divide and Conquer* karena algoritma ini hanya memroses satu upa-persoalan saja.

Berdasarkan jumlah upa-persoalan yang berkurang untuk setiap iterasinya, algoritma ini bisa diklasifikasikan menjadi tiga bagian berikut, yaitu:

a. *Decrease By A Constant*
   Pada varian ini, ukuran instans persoalan direduksi sebesar konstanta yang sama setiap iterasinya. Biasanya, konstanta yang digunakan dalam kebanyakan kasus adalah konstanta satu. Contoh implementasi algoritma jenis ini adalah pada *topological sort*, *insertion sort*, dan algoritma pencarian traversal pada graf seperti DFS dan BFS.
b. *Decrease By A Constant Factor*
   Pada jenis kedua, ukuran instans persoalan direduksi sebesar faktor konstanta yang sama setiap iterasinya. Biasanya, faktor konstanta yang digunakan dalam kebanyakan kasus adalah faktor konstanta dua. Contoh implementasi algoritma jenis ini adalah pada *binary search* atau persoalan mencari koin palsu.
c. *Decrease By A Variable Size*
   Pada varian terakhir, ukuran instans persoalan direduksi bervariasi atau berubah-ubah pada setiap iterasi algoritmanya sehingga disebut dengan *variable*. Contoh dari persoalan ini adalah algoritma Euclidean atau *interpolation search*.

2.2 *Topological Sort*

Directed Acyclic Graph adalah sebutan untuk sebuah graf berarah yang untuk setiap kemungkinan sisi dan simpulnya tidak membentuk siklus sama sekali. Misalkan sebuah graf yang memiliki simpul A dan B serta bersisi E yang menjadi penghubung keduanya. Sisi yang bersifat *forward edges* ini menggambarkan sebuah relasi *dependency* bagi kedua simpul tersebut. Jika A memiliki *forward edges* ke B, artinya A menjadi *dependency*, atau dalam kasus persoalan kita, menjadi *pre-requisites* untuk B.

Salah satu algoritma pengurutan yang memerlukan sebuah graf untuk memenuhi syarat sebagai DAG adalah *Topological Sort*. Algoritma *topological sort* akan mencari sebuah simpul yang tidak memiliki derajat masuk sama sekali pada setiap iterasi pengurutannya. Siklus pada graf akan menyebabkan terjadinya *deadlock* pada

forward edges yang ada, sehingga menyebabkan tidak ada lagi simpul yang bisa diambil pada langkah tertentu oleh algoritma ini.

Lebih lanjut, *topological sort* bekerja secara iteratif dengan mengambil dan mengeluarkan simpul beserta sisi yang *dependent* dengan simpul tersebut jika derajat masuknya tidak ada. Iterasi ini akna dilakukan secara berulang-ulang sampai dengan dihasilkan graf kosong tanpa simpul. Dalam hal ini, graf harus memiliki sebuah simpul yang bisa digunakan sebagai titik awal iterasi, supaya iterasi pencarian yang dilakukan oleh graf tidak menjadi *infinite*.

Dengan cara kerja yang demikian, *topological sorting* menjadi salah satu contoh implementasi algoritma *Decrease and Conquer*. Hal ini dikarenakan pada setiap iterasi yang dilakukan oleh *topological sorting*, akan dilakukan pengurangan instans persoalan, yakni simpul yang diurutkan. Dengan berkurangnya instans persoalan, maka proses pemecahan persoalan pada tahap berikutnya menjadi lebih mudah dan cepat.

Secara umum, algoritma *topological sort* dikelompokkan ke dalam algoritma *Decrease and Conquer* berjenis *Decrease By A Constant*. Dalam hal ini, konstanta yang digunakan dalam algoritma tersebut adalah satu, yang menyatakan jumlah simpul yang dihilangkan dari graf untuk setiap iterasinya.

Namun, dalam permasalahan *prerequisites* dan penyusunan rencana kuliah ini, algoritma *topological sort* dapat dimasukkan ke dalam kategori *Decrease By A Variable Size*. Pada setiap semester kita tidak bisa memprediksi jumlah mata kuliah yang bisa diambil pada satu semester tanpa melanggar *constraint* prasyarat yang ada. Kasus yang mungkin terjadi adalah pada suatu semester hanya bisa diambil satu mata kuliah saja, ataupun bisa diambil beberapa mata kuliah sekaligus yang tidak saling berhubungan dan tidak memiliki prasyarat apapun.

3. *Source Code* Program

Program dibuat secara modular dalam satu file dengan menggunakan bahasa pemrograman Python. Aplikasi penyusunan rencana perkuliahan dengan *topological sort* ini diberi nama *Abyss*. Dalam proses pengerjaannya, terdapat dua *approach* yang dilakukan. Pendekatan pertama menggunakan pendekatan yang dilakukan secara iterattif terhadap struktur data yang digunakan. Pendekatan kedua dilakukan secara rekursif untuk struktur data pada *state* tertentu ketika program dieksekusi. Program pertama berhasil memberikan solusi yang benar tetapi kurang optimal dalam pengambilan mata kuliah yang seharusnya bisa diambil dalam satu semester yang sama. Oleh karena itu, pada akhirnya digunakan pendekatan kedua yang memberikan hasil yang optimal.
Berikut merupakan *screenshot* dari *source code* program *Abyss_13519185.py* yang mengandung kedua *approach*.

```python
# Richard Rivaldo
# 13519185 - K04

# Read and process the subjects in each line of the file
def readIntoList(filename):
    # Open and Read File
    file = open(filename, 'r')

    # Initialize an empty list
    fullList = []

    # Preprocess the string
    for line in file:
        # Ignore empty line that contains newline
        if(line != '\n'):
            # Clean the text
            line = line.replace(".", "")
            line = line.replace('\n', "")
            line = line.replace(" ", "")

            # Split and get each subjects
            lineList = (line.split(","))

            # Append to the end list
            fullList.append(lineList)

    # Close the file
    file.close()

    return fullList
```

Gambar 4. Fungsi readIntoList untuk Membaca Masukan File

```python
# Topological Sort with list comprehension and recursive approach
# This list comprehension acts analogously with the concept of graph as
# the data structure in most topological sort programs
def topologicalSort(subject):
    # Check if the input file contains any subject
    # If not, terminate all process
    if(len(subject) == 0):
        print("\nZERO ZERO ZERO")
        print("Is it just me, or I cannot see a thing in the file you just give me?")
        print("PROCESS TERMINATED: Code 0, invalid file!")
        return

    # Initialize boolean to track the sorting process
    # If the graph does not contain any vertex with 0 In-Degree
    foundOne = False

    # Initialize an empty list of list that contains yet another list
    # of valid subject to take each semester
    semSet = []

    # Init a list to contain a list of every taken subject each semester
    listSem = []

    # Find a subject without any prerequisites (In-Degree = 0)
    for prereq in subject:
        if(len(prereq) == 1):
            # Append the subject
            listSem.append(prereq)

            # Change the value of the boolean to indicate that at least
            # one vertex has no input edges
            foundOne = True

    # Check if the subject can be allocated with Topological Sort schema
    # If there is no subject without input edges, terminate the process
    if(not foundOne):
        print("\nONE ONE ONE")
        print("BEEPBEEP. Endure it anymore, and we will break!")
        print("This is as far as we go, mate. Goodbye..")
        print("PROCESS TERMINATED: Code 1: No more valid subject to take!")
        return
```

Gambar 5. Fungsi topologicalSort untuk Mengurutkan Rencana Perkuliahan (1)

```
    # Decrease and Conquer Approach
    # Remove the subject from the all subject-prereq list
    subject = [prereq for prereq in subject if prereq not in listSem]

    # Append the list of the semester to the result list
    semSet.append(listSem)

    # Init an empty list to contain an updated version of the subject-prereq list
    updatedSubject = []

    # Iterate over the remaining original subject-prereq list
    for remaining in subject:
        # Iterate over the result we got on each semester
        for prereq in listSem:
            # Remove every dependencies that the choosen subject of each semester
            # with remaining subjects from the list
            if(prereq[0] in remaining):
                remaining.remove(prereq[0])
        # Append the subject to the list of updated subjects
        updatedSubject.append(remaining)

    # Join every choosen subjects of each semester into a string
    string = ', '.join([prereq[0] for prereq in semSet[0]])

    # Make the counter for every semester global to keep track of the count
    global semCounter

    # Initialize an array to contain roman numbers
    roman = ["I", "II", "III", "IV", "V", "VI", "VII", "VIII",]

    # Format the string to print the result of each semester with roman number
    print(f'Semester {roman[semCounter]}:'.format(roman[semCounter]), string)

    # Increment of the semester counter
    semCounter += 1

    # Main basis of the recursion
    # No more subjects left in the list
    # (No more nodes left in the graph)
    if(len(updatedSubject) == 0):
        return

    # Handle possibility of having more than 8 Semester
    # to take all subjects
    if(semCounter == 8 and len(updatedSubject) != 0):
        return

    # Recursively call the function until all subjects are allocated to every semester
    # i.e. no more subjects left to choose from
    return topologicalSort(updatedSubject)
```

Gambar 6. Fungsi topologicalSort untuk Mengurutkan Rencana Perkuliahan (2)

```python
# Main Program
def abyss():
    # Abyss is Chaos in Greek :)
    print("HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO")
    print("Don't know what subject to take? We are here!\n")
    print("Abyss: Your chaos solver is ready to help!\n")

    # Receive a .txt file name as an input
    filename = input("Give us a food! FYI, we eat .txt files~~\n")

    print("\n")

    # Init semCounter as a global variable
    # Track the counter of Roman Number in the list
    global semCounter
    semCounter = 0

    # Try to read the file into list of subjects-prerequisites
    try:
        # Read the file into list and save it to subject
        subject = readIntoList(filename)
    except:
        # Error message if file invalid or not found
        print("Invalid: Wrong file name or file not found!")
        print("We, Eat, .txt, Files!")
        return

    # Do topological sorting and gives output
    topologicalSort(subject)

    # ~~~
    print("\nMay chaos not be with you anymore.. and me..")
    print("But Abyss is still here. Call me anytime!")
    print("A       B          B           Y        S")

    return

# Call from terminal
if __name__ == "__main__":
    abyss()
```

Gambar 7. Program Utama *Abyss*

```python
# def schedule(sortedResult):
#     # Init counter
#     counter = 1
#
#     # Iterate over result
#     for semester in sortedResult:
#         # Can take only one subject each semester
#         if(len(semester) == 1):
#             print("Semester", counter, ":", semester[0])
#         else:
#         # Can take than one subjects each semester
#             print("Semester", counter, ":", end = " ")
#             for idx in range(len(semester)):
#                 if(idx == len(semester) - 1):
#                     print(semester[idx])
```

Gambar 8. Fungsi schedule dengan Pendekatan Iteratif

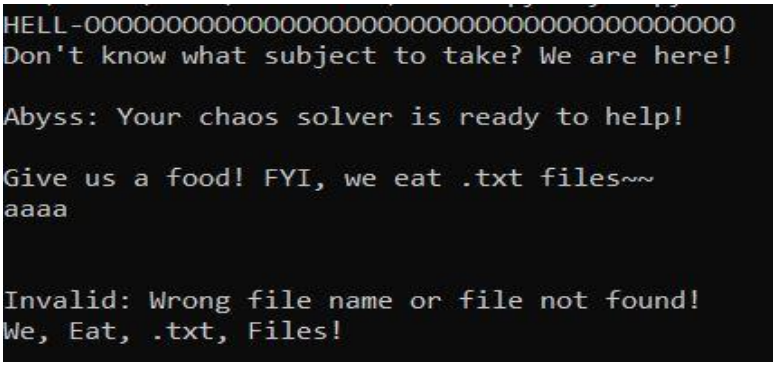
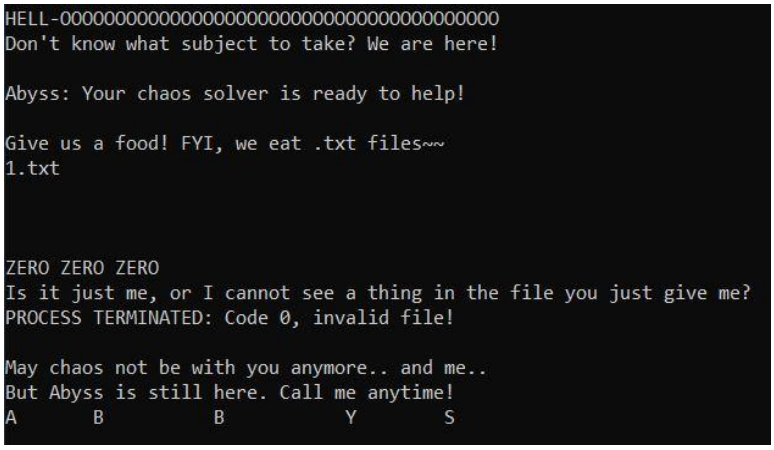
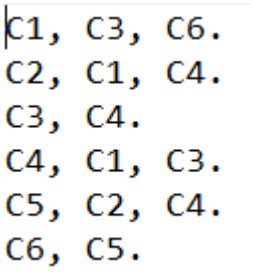
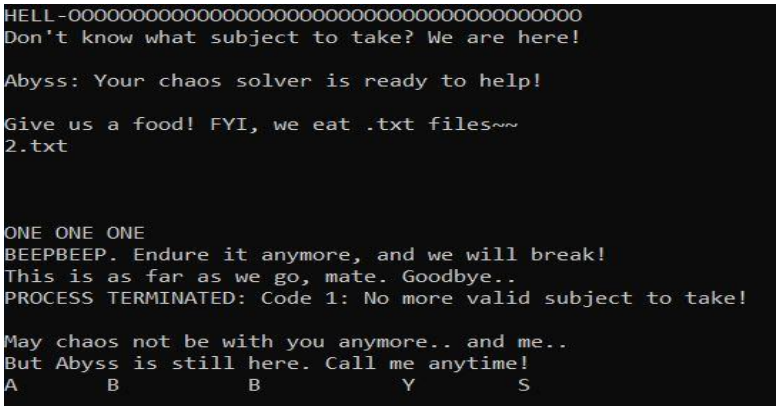
```
# Old approach without recursive
# Works, but does not give optimal output based on the concept of Topological Sorting
# ================================================================================
# # Check if a subject is a prerequisite of other subject
# def checkPrereq(subject, new, target):
#     # Find Target Subject
#     prereqList = []
#     foundNew = False
#     foundTarget = False
#
#     for prereq in subject:
#         if(prereq[0] == target):
#             prereqList = prereq
#             if(new in prereqList):
#                 foundNew = True
#         if(prereq[0] == new):
#             prereqList = prereq
#             if(target in prereqList):
#                 foundTarget = True
#
#     # Check if new is in the prerequisite list of target
#     return foundNew and foundTarget
#
# def topologicalSort(subject):
#
#     # Init a copy of Subject List and an empty list to contain results
#     copySubject = subject
#     result = []
#     matkul = ""
#
#     # Iterate until the subject list is empty -> no more nodes in graph
#     while(len(subject) != 0):
#         # Find subject without any input edges
#         for subjects in subject:
#
#             # Init empty list to contain subjects in each semester
#             semesterList = []
#
#             # Get the subjects if the length of the list is 1 -> In-Degree = 0
#             if(len(subjects) == 1):
#                 matkul = subjects[0]
#                 semesterList.append(matkul)
#
#                 # Remove the subject from current list
#                 subject.remove(subjects)
#
#                 # Append the subject to corresponding semester
#                 for rest in subject:
#                     if(len(rest) == 1 and not checkPrereq(copySubject, rest[0], matkul)):
#                         matkul = rest[0]
#                         semesterList.append(matkul)
#
#                         # Remove the subject from current list
#                         subject.remove(rest)
#
#                 # Add each semester to the end result
#                 result.append(semesterList)
#
#             # Remove the subject from remaining list (graph nodes connected with the subject)
#             for choosen in semesterList:
#                 for remaining in subject:
#                     if(choosen in remaining):
#                         remaining.remove(choosen)
#
#     return result
```

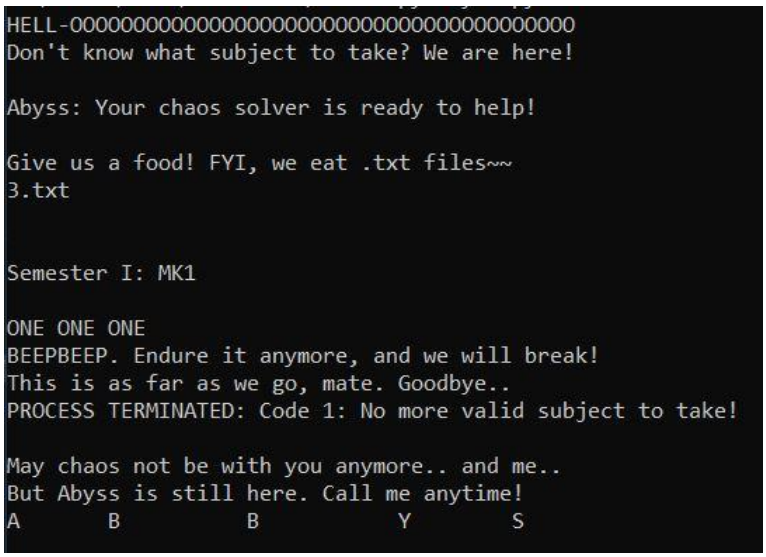
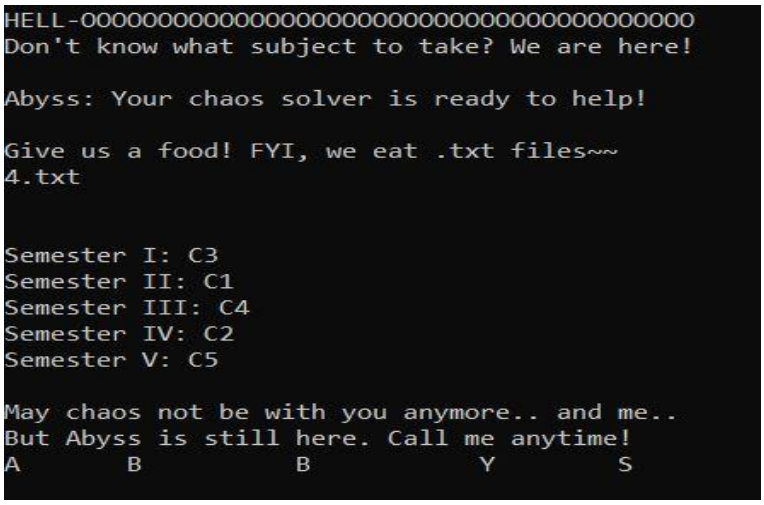
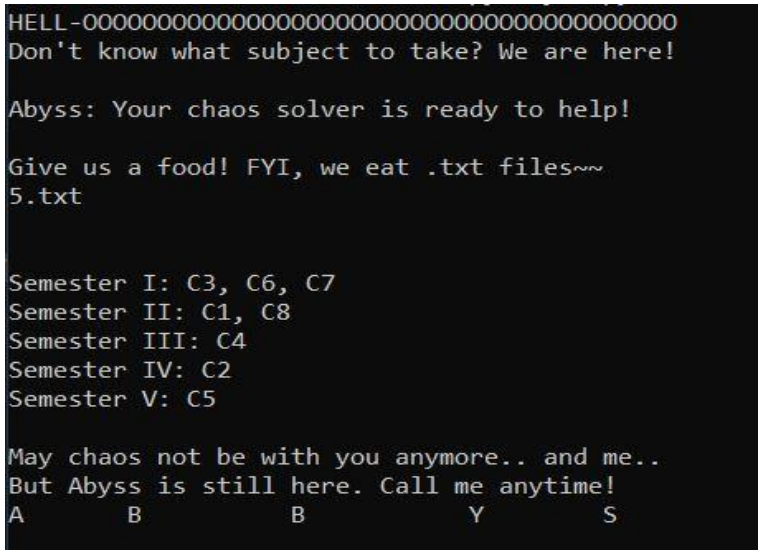
Gambar 9. Fungsi checkPrereq dan topologicalSort dengan Pendekatan Iteratif

5. Hasil Eksekusi *Testcases*

Eksekusi *testcases* berikut dilakukan di dengan pemanggilan program *Abyss.py* melalui Terminal atau Command Prompt dengan *command `python Abyss_13519185.py`*. Harus dipastikan terlebih dahulu bahwa file yang digunakan sebagai *testcase* sudah terdapat pada direktori yang sama dengan *Abyss_13519185.py* dan lokasi terminal adalah lokasi dari *Abyss_13519185.py*.

Tabel 1 Hasil Eksekusi *Testcases*

Input File	Output *Abyss*
Invalid File	
File Kosong	
 Bukan DAG	

MK1. MK2, MK1, MK3. MK3, MK4. MK4, MK5, MK6. MK5, MK6. MK6, MK3. Tidak ada lagi simpul yang tidak memiliki *forward edges*	
C1, C3. C2, C1, C4. C3. C4, C1, C3. C5, C2, C4.	
C1, C3. C2, C1, C4. C3. C4, C1, C3. C5, C2, C4. C6. C7. C8, C3.	

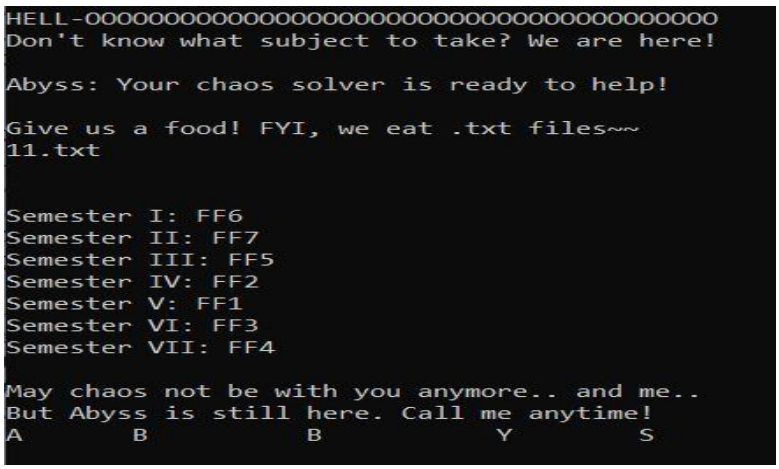
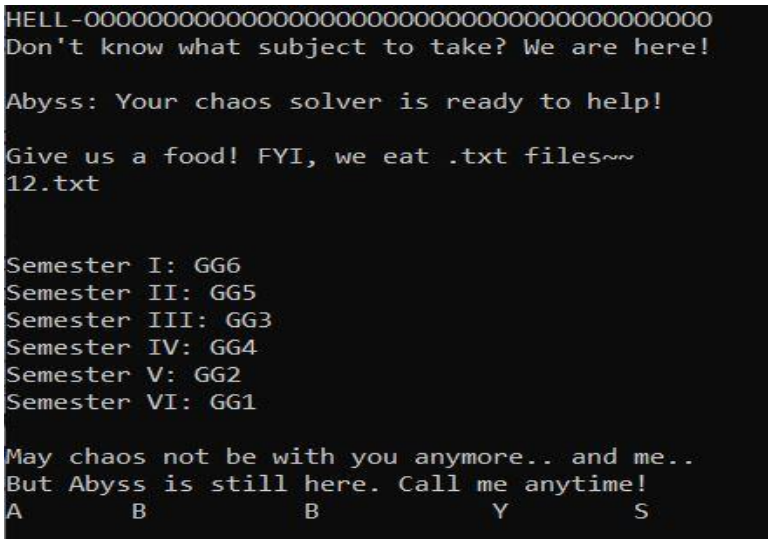
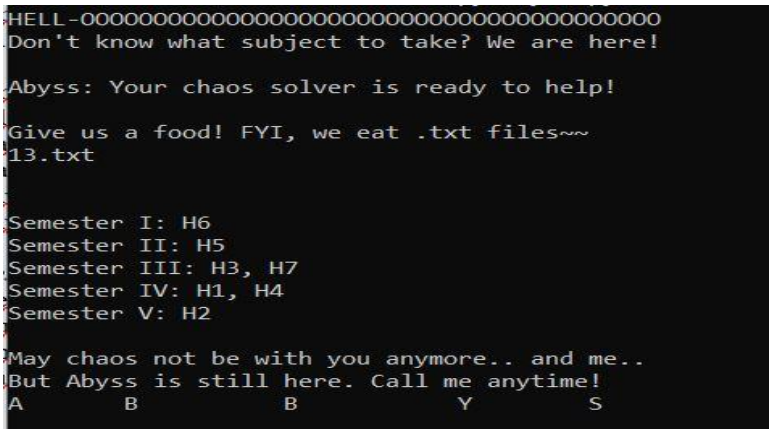
```
STI0000.
STI0058, STI0000, STI4269, STI2807, STI1345, STI5614, STI9883, STI8421, STI2690, STI6959, STI12
STI0116, STI0000, STI4269, STI2807, STI7076, STI5614, STI9883, STI4152, STI8421, STI5497, STI97
STI0175, STI0000, STI4269, STI8538, STI2807, STI7076, STI5614, STI4152, STI8421, STI2690, STI97
STI0233, STI0000, STI4269, STI8538, STI1345, STI9883, STI8421, STI2690, STI8304, STI2573, STI53
STI0292, STI0000, STI8538, STI7076, STI8421, STI6959, STI5497, STI9766, STI8304, STI2573, STI68
STI0350, STI0000, STI5614, STI4152, STI8421, STI6959, STI5497, STI9766, STI4035, STI2573, STI68
STI0409, STI4269, STI7076, STI5614, STI9883, STI8421, STI2690, STI6959, STI5497, STI1111, STI96
STI0467, STI0000, STI2807, STI1345, STI5614, STI9883, STI4152, STI6959, STI1228, STI5497, STI83
STI0526, STI0000, STI4269, STI2807, STI5614, STI9883, STI8421, STI6959, STI1228, STI9766, STI83
STI0584, STI4269, STI8538, STI7076, STI9883, STI4152, STI1228, STI5497, STI9766, STI6842, STI96
STI0643, STI8538, STI2807, STI7076, STI5614, STI4152, STI2690, STI1228, STI6842, STI1111, STI53
STI0701, STI0000, STI4269, STI8538, STI7076, STI1345, STI5614, STI8421, STI1228, STI5497, STI97
STI0760, STI0000, STI8538, STI4152, STI8421, STI2690, STI1228, STI5497, STI9766, STI4035, STI68
STI0818, STI8538, STI2807, STI1345, STI5614, STI9883, STI8421, STI2690, STI5497, STI2573, STI68
STI0877, STI0000, STI8538, STI2807, STI7076, STI5614, STI4152.
STI0935, STI4269, STI1345, STI4152, STI8421, STI2690, STI6959, STI1228, STI9766, STI4035, STI11
STI0994, STI4269, STI7076, STI1345, STI5614, STI8421, STI2690.
STI1052, STI7076, STI1345, STI6959, STI1228, STI4035, STI8304, STI2573, STI6842, STI8187, STI24
STI1110, STI4269, STI8538, STI2807, STI1345, STI9883, STI4152, STI8421, STI9766, STI8304, STI25
STI1111, STI2807, STI7076, STI8421.
STI1169, STI0000, STI4269, STI2807, STI1345, STI5614, STI4152, STI8421, STI2690, STI1228, STI97
STI1227, STI0000, STI4269, STI8538, STI2807, STI7076, STI1345, STI5614, STI9883, STI4152, STI84
STI1228, STI8538, STI1345, STI5614, STI8421.
STI1286, STI8538, STI2807, STI8421, STI2690, STI9766, STI2573, STI6842, STI9649, STI8187, STI09
STI1344, STI0000, STI4269, STI8538, STI2807, STI1345, STI5614, STI4152, STI8421, STI2690, STI12
STI1345, STI0000.
STI1403, STI4269, STI2807, STI5614, STI8421, STI2690, STI6959, STI1228, STI8304, STI2573, STI68
```

```
HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
Don't know what subject to take? We are here!

Abyss: Your chaos solver is ready to help!

Give us a food! FYI, we eat .txt files~~
6.txt


Semester I: STI0000, STI2690, STI2807, STI4152, STI4269, STI5614, STI8421, STI8538, STI9883
Semester II: STI1345, STI2339, STI2456, STI4035, STI5146, STI5497, STI6725, STI7076
Semester III: STI0877, STI0994, STI1111, STI1228, STI2222, STI2573, STI3684, STI3881, STI3918, STI5263, STI5380, STI6491, STI6608, STI6842, STI6959, STI7953, STI8070, STI8187, STI8304, STI9415, STI9532, STI9649,
 STI9766
Semester IV: STI0409, STI0526, STI0643, STI0760, STI1871, STI1988, STI2105, STI3333, STI3450, STI3567, STI4678, STI4795, STI4912, STI5029, STI6140, STI6257, STI6374, STI7602, STI7719, STI7836, STI8947, STI9064,
STI9181, STI9298
Semester V: STI3216, STI7485
Semester VI: STI1754, STI3099, STI4561
Semester VII: STI0292, STI2982, STI6023, STI8830
Semester VIII: STI1637, STI5906, STI7251, STI7368

May chaos not be with you anymore.. and me..
But Abyss is still here. Call me anytime!
A     B       B       Y     S
```

```
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.

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

```
HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
Don't know what subject to take? We are here!

Abyss: Your chaos solver is ready to help!

Give us a food! FYI, we eat .txt files~~
7.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

May chaos not be with you anymore.. and me..
But Abyss is still here. Call me anytime!
A       B          B          Y        S
```

Lahir. Sekolah, Lahir. Kuliah, Sekolah. Kerja, Kuliah. Stres, Lahir. Bermain, Lahir. Bunuh Diri, Stres. Mati, Bunuh Diri, Tua. Nikah, Kerja. Beranak, Nikah. Gajian, Kerja. Anak Besar, Beranak. Tua, Lahir. Bercucu, Anak Besar. Sakit, Tua.	HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO Don't know what subject to take? We are here! Abyss: Your chaos solver is ready to help! Give us a food! FYI, we eat .txt files~~ 8.txt Semester I: Lahir Semester II: Sekolah, Stres, Bermain, Tua Semester III: Kuliah, BunuhDiri, Sakit Semester IV: Kerja, Mati Semester V: Nikah, Gajian Semester VI: Beranak Semester VII: AnakBesar Semester VIII: Bercucu May chaos not be with you anymore.. and me.. But Abyss is still here. Call me anytime! A B B Y S
DD1, DD2, DD3, DD4. DD2, DD3. DD3, DD6. DD4, DD5. DD5. DD6.	HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO Don't know what subject to take? We are here! Abyss: Your chaos solver is ready to help! Give us a food! FYI, we eat .txt files~~ 9.txt Semester I: DD5, DD6 Semester II: DD3, DD4 Semester III: DD2 Semester IV: DD1 May chaos not be with you anymore.. and me.. But Abyss is still here. Call me anytime! A B B Y S
EE1, EE6, EE7, EE8 EE2, EE3. EE3, EE1. EE4, EE1, EE2, EE3. EE5, EE6, EE7, EE4. EE6, EE7. EE7. EE8, EE7.	HELL-OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO Don't know what subject to take? We are here! Abyss: Your chaos solver is ready to help! Give us a food! FYI, we eat .txt files~~ 10.txt Semester I: EE7 Semester II: EE6, EE8 Semester III: EE1 Semester IV: EE3 Semester V: EE2 Semester VI: EE4 Semester VII: EE5 May chaos not be with you anymore.. and me.. But Abyss is still here. Call me anytime! A B B Y S

FF1, FF2. FF2, FF5. FF3, FF1, FF2, FF5. FF4, FF3, FF5. FF5, FF7. FF6. FF7, FF6.	
GG1, GG2, GG3, GG4. GG2, GG3, GG4. GG3, GG5. GG4, GG3. GG5, GG6. GG6.	
H1, H3, H5. H2, H3, H4. H3, H5. H4, H3. H5, H6. H6. H7, H5.	

Testcases yang tidak diberi keterangan adalah *testcases* yang berhasil dan bukan *testcase* khusus. Selain itu, beberapa file masukan yang isinya terlalu panjang hanya dimasukan sebagian gambarnya saja, seperti pada *testcase* 6 dan *testcase* 7.

6. Kode Program

Berikut merupakan alamat yang dapat diakses untuk mengunduh kode program *Abyss.py* beserta file *testcases* yang digunakan dalam laporan.

Google Drive: *https://drive.google.com/drive/folders/1ICq49x2qsYYM8-CNkx2veOzMxTXO9uSM?usp=sharing*.

Github: *https://github.com/RichardRivaldo/Algorithm-Strategies*.

7. Simpulan dan Refleksi

Algoritma *Topological Sorting* yang diimplementasikan dalam penyusunan rencana perkuliahan pada aplikasi *Abyss* menggunakan analogi struktur data *list* menjadi graf. Dalam hal ini, kompleksitas waktu yang diperlukan dalam eksekusi program tidak terlalu tinggi dan dapat mengeluarkan *output* dengan cepat. Namun, *readability* dari kode program dan algoritma yang digunakan mungkin cukup sulit untuk dipahami. Oleh karena itu, *topological sorting* ini memang lebih cocok menggunakan struktur data graf sebagai representasi data yang menjadi masukannya.

8. Sumber

MTU. *Decrease and Conquer Sorts and Graph Searches.* Diakses pada 24 Februari 2021 pukul 18.21 WIB melalui *http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%2010%20-%20Decrease%20and%20Conquer%20Sorts%20and%20Graph%20Searches.htm*.

Stack Overflow. Diakses pada 26 Januari 2021 pukul 19.42 WIB melalui *https://stackoverflow.com/*.

Tim Strategi Algoritma Informatika 2021. *Algoritma Divide and Conquer*. Diakses pada 25 Februari 2021 pukul 21.42 WIB melalui *https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm*.

Wangyy. *Course Schedule and Topological Sorting.* Diakses pada 25 Februari 2021 pukul 20.21 WIB melalui *https://wangyy395.medium.com/course-schedule-and-topological-sorting-7deac2802053*.

9. Lampiran

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.	✓	