

Penyusunan Rencana Kuliah dengan *Topological Sort* (Penerapan *Decrease and Conquer*)

LAPORAN TUGAS KECIL 2

Diajukan Untuk Memenuhi Tugas Kecil IF 2211 Strategi Algoritma
Semester II 2020/2021



Disusun oleh

Reihan Andhika Putra

(13519043)

**TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020**

BAB I

LANDASAN TEORI

1.1. *Decrease and Conquer*

Strategi algoritma ini memiliki cara dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil. Perbedaannya dengan *divide and conquer* adalah metode ini tidak memproses semua sub-persoalan dan menggabungkan semua solusi setiap sub-persoalan. *Decrease and conquer* terdiri dari dua tahapan:

1. *Decrease*: mereduksi persoalan menjadi beberapa persoalan yang lebih kecil (biasanya dua sub-persoalan).
2. *Conquer*: memproses satu sub-persoalan secara rekursif.

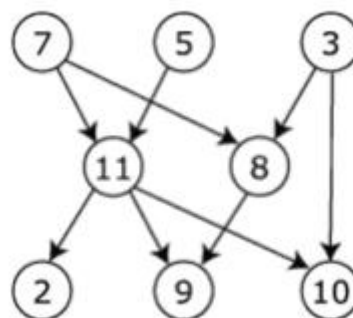
Perlu digarisbawahi bahwa tidak ada tahap *combine* dalam *decrease and conquer*. Ada tiga varian *decrease and conquer*:

1. ***Decrease by a constant***: Ukuran instans dari persoalan direduksi sebesar konstanta yang sama setiap iterasi algoritma. Biasanya konstanta. Ada beberapa permasalahan yang dapat diselesaikan dengan tipe ini, antara lain:
 - a) *Insertion sort*: Untuk melakukan pengurutan pada larik $A[0..n-1]$, lakukan pengurutan pada $A[0..n-2]$ secara rekursif lalu masukkan $A[n-1]$ pada tempat yang benar pada larik $A[0..n-2]$ yang sudah terurut. Biasanya diimplementasikan dengan skema non-rekursif (bottom up).
 - b) *Selection sort*
 - c) Algoritma Graf Traversal (DFS dan BFS)
 - d) *Topological sorting*: Mengurutkan simpul simpul yang ada di sebuah rantai, dengan menggunakan metode
 - *DFS-based Algorithm*, atau
 - *Source Removal Algorithm*
2. ***Decrease by a constant factor***: Ukuran instans dari sebuah persoalan direduksi sebesar faktor konstanta yang sama setiap iterasi. Biasanya, faktor konstanta bernilai 2. Gambar 2-3 Flowchart decrease by a constant factor. Ada beberapa permasalahan yang dapat diselesaikan dengan tipe ini, antara lain:
 - a) *Binary search* dan Metode *bisection*
 - b) *Multiplication à la russe*: Melakukan komputasi pada produk dari 2 integer positif
 - c) *Interpolation Search*

- d) Mencari koin palsu: Diberikan n buah koin yang identik, tetapi salah satu diantaranya merupakan koin palsu
- 3. ***Decrease by a variable size***: Ukuran instans persoalan direduksi bervariasi pada setiap iterasi algoritma.
 - a) Algoritma Euclid untuk *Greatest Common Divisor* (GCD)
 - b) Algoritma *Partition-based* untuk masalah seleksi
 - c) Beberapa algoritma pada *Binary Search Tree* (BST).
 - d) Menghitung median dan *Selection Problem*: Mencari median dari *unsorted array* namun tidak perlu mengurutkannya terlebih dahulu.

1.2. Topological Sort

Topological sorting atau *topological ordering* adalah algoritma untuk melakukan pengurutan secara linier terhadap semua simpul dari sebuah graf berarah, di mana setiap sisi uv , simpul u berada sebelum v pada hasil pengurutan. Setiap simpul dari graf tersebut dapat saja dimisalkan sebagai suatu pekerjaan yang harus dilakukan. Sisi yang ada menandakan urutan pengerjaan, bahwa suatu pekerjaan harus dilakukan terlebih dahulu sebelum pekerjaan yang lain. Dalam hal ini, *topological sorting* atau *topological ordering* adalah keterurutan yang sah dari penjadwalan pekerjaan ini. *Topological sorting* dapat dilakukan jika dan hanya jika graf yang bersangkutan tidak mempunyai siklus berarah, dengan kata lain graf itu adalah *Directed Acyclic Graph* (DAG) atau graf berarah yang tidak mempunyai siklus. Setiap DAG pasti mempunyai minimal satu keterurutan secara topologi dan algoritma *topological sorting* digunakan untuk menemukan keterurutan topologi dari DAG tersebut secara linier.



Gambar 1 Contoh Directed Acyclic Graph (DAG)

Salah satu keterurutan topologi dari DAG di atas adalah: 5, 7, 3, 8, 11, 10, 9, 2. Keterurutan ini dilihat dari sedikitnya sisi yang masuk ke suatu simpul. Berikut adalah algoritma dari topological sorting secara umum:

```

L ← list kosong penampung simpul terurut
S ← himpunan simpul yang tidak mempunyai sisi yang masuk
while S not-empty do
    ambil simpul n dari S
    masukkan n ke L
    for each simpul m dengan sisi e dari n ke m do
        hilangkan sisi e dari graf
        if m tidak punya sisi masuk lagi then
            masukkan m ke dalam S
if graf memiliki sisi then
    return error (graf memiliki setidaknya satu siklus)
else
    return L (keterurutan secara topologi diperoleh)

```

1.3. Hubungan *Topological Sort* dengan *Decrease and Conquer*

Topological Sort adalah salah satu penerapan *decrease and conquer* dengan variasi *decrease by a constant*. *Constant* yang dipakai adalah satu. Tahap *decrease* dari *topological sort* adalah saat mencari simpul yang tidak mempunyai sisi masuk. Simpul yang tidak mempunyai sisi masuk akan langsung dimasukkan kedalam urutan *topological sort* dan tidak perlu diproses lagi. Simpul yang tidak mempunyai sisi masuk dipilih satu-persatu selama proses berlangsung sehingga jumlah simpul yang harus diurutkan akan berkurang satu-persatu juga (*decrease by a constant 1*). Tahap *conquer*-nya adalah setelah memilih simpul maka semua sisi keluar dari simpul yang terpilih dihilangkan dan proses *topological sorting* diulangi secara rekursif untuk simpul yang belum diurutkan.

BAB II

SOURCE CODE PROGRAM

```
#-----***** Fungsi dan Prosedur *****-----#
##### Kelompok Kerja Manajemen Node dan Edge #####
def unvisited(node):
    # I.S node merupakan sebuah course_id yang akan dicek apakah sudah pernah dikunjungi a
    tau belum
    # F.S Mengirimkan true jika node belum pernah dikunjungi, false jika sudah
    for course in visitedNodes:
        if (node == course):
            return False
    return True

def addEdge(edge,u,v):
    # I.S edge adalah dictionary , u adalah key dan v adalah valuenya
    # F.S value dari dictionary "edge" elemen "u" akan ditambah dengan v
    # Salah satu kegunaan fungsi ini adalah merepresentasikan node(course) "u" terhubung de
    ngan node "v" dengan "v" adalah prerequisite dari "u"
    edge[u].append(v)

def findZeroPrereqCourse():
    # Merupakan fungsi untuk mencari course yang prereqnya sudah 0
    # Course yang dipilih haruslah course yang belum pernah dikunjungi
    for i in range(len(prereqCount)):
        if(prereqCount[i] == 0 and unvisited(uniqueCourseId[i])):
            return uniqueCourseId[i]
    # Tidak ada course yang prereqnya 0, dipastikan graf bukan DAG
    return None

#####
#-----*****-----*****-----#

#-----***** Main Program *****----- #
# Import library
from collections import defaultdict
from time import sleep
import os
import roman # library yang harus di download -> untuk romawi di semester

# Deklarasi Variabel-Variabel Global
# dictionary yang merepresentasikan edge antara course dan prereqnya (original)
originaledge = defaultdict(list)
# mirip seperti originaledge namun data edge akan dinamis sesuai algoritma TopoSort
edge = defaultdict(list)
listOfMatkul = defaultdict(list) # Dictionary antara course_id dan course_fullname
uniqueCourseId = [] # Kumpulan course_id unik dari input file
prereqCount = [] # Jumlah prereq tiap modul
```

```

semValue = [] # Semester paling rendah tiap course_id
visitedNodes = [] # Node yang sudah dikunjungi
coursePrereqs = []
# Membaca input dari file txt
filename = input("Masukkan nama file (tanpa ekstensi): ")
inputss = open('test/'+ filename + '.txt','r').read().split('\n')
listMatkul = open('test/'+ "List Matkul" + '.txt','r').read().split('\n')

# Parsing input (Membagi mana yang course_id dan mana yang preq_id dan lain-lain)
for inputs in inputss:
    inputs = inputs.replace(' ', '')
    inputs = inputs.replace('.', '')
    courseSpec = inputs.split(',')
    courseId = courseSpec[0]
    uniqueCourseId.append(courseId)
    prereqCount.append(len(courseSpec)-1)
    semValue.append(1)
    coursePrereq = courseSpec[1:]
    coursePrereqs.append(coursePrereq)
    for course in coursePrereq:
        addEdge(edge, courseId, course)
        addEdge(originalEdge, courseId, course)

# Parsing input (Membentuk dictionary antara course_id dan course_fullname)
for matkul in listMatkul:
    matkul = matkul.replace('\t', '')
    infoMatkul = matkul.split("-")
    addEdge(listOfMatkul, infoMatkul[0], infoMatkul[1])

# Algoritma Topological Sort dengan Pendekatan Decrease and Conquer
# Decrease and Conquer yang digunakan adalah decrease by constant (n=1)
def decreaseAndConquer(visitedNodes):
    # Base case : Semua node sudah dikunjungi
    if(len(visitedNodes)==len(uniqueCourseId)):
        print("Decrease and Conquer ~ selesai")
    else:
        print("Melakukan Decrease and Conquer ~ " + str(len(uniqueCourseId)-
len(visitedNodes))+ " node yang harus dikunjungi")
        sleep(0.125)
        # Decrease : Ambil satu node(matkul) yang semua prequisisite nya sudah terpenuhi
        # Masukkan ke daftar node yang sudah dikunjungi dan jangan proses node itu lagi
        zeroPrereqCourse = findZeroPrereqCourse()
        if zeroPrereqCourse is None:
            print("Siklus ditemukan!!!")
            print("Anda tidak memasukkan DAG")
            sleep(3)
            quit()

```

```

visitedNodes.append(zeroPrereqCourse)
# Proses node(matkul) lain yang berhubungan dengan node yang baru saja dikunjungi
for course in edge:
    for prereq in edge[course]:
        if (prereq == zeroPrereqCourse):
            # Kurangi jumlah prequisisite tersisa dari matkul yang prequisisitenya adalah
            node(matkul) yang baru saja dikunjungi
            prereqCount[uniqueCourseId.index(course)]-=1
            # Pastikan bahwa semua semester pelaksanaan suatu matkul haruslah setelah matkul
            prequisisitenya dilaksanakan
            if (semValue[uniqueCourseId.index(course)] <= semValue[uniqueCourseId.index(zeroPrereqCourse)]):
                semValue[uniqueCourseId.index(course)] = semValue[uniqueCourseId.index(zeroPrereqCourse)] + 1
            # Hilangkan edge(hubungan prequisisite) dari node(matkul) yang terhubung dengan node yang baru saja dikunjungi
            edge[course].pop((edge[course].index(prereq)))
# Conquer : Secara rekursif, selesaikan hingga semua node dikunjungi
decreaseAndConquer(visitedNodes)

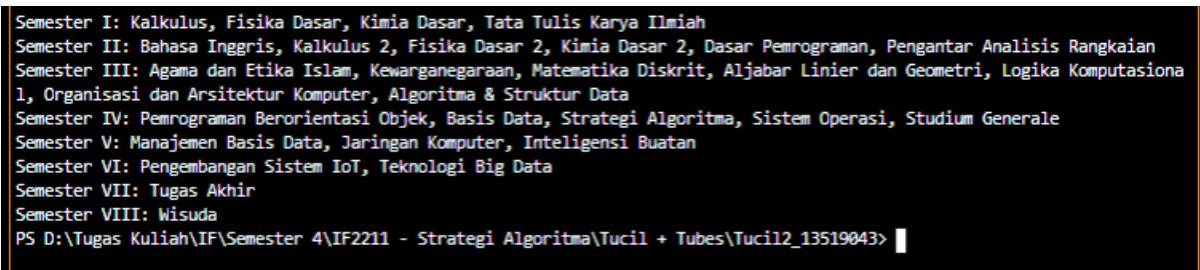
# Inisialisasi Decrease and Conquer
decreaseAndConquer(visitedNodes)

# Menulis hasil Decrease and Conquer sesuai aturan
# Tulis peringatan apabila dibutuhkan lebih dari 8 semester untuk semua matkul
if(max(semValue)>8):
    print("Warning!! Untuk menyelesaikan semua matkul dibutuhkan lebih dari 8 Semester")
for i in range (max(semValue)):
    first = True
    print("Semester " + roman.toRoman(i+1)+": ", end="")
    for course in uniqueCourseId:
        if(semValue[uniqueCourseId.index(course)]==i+1):
            if (first):
                print(*listOfMatkul[course], end="")
                first = False
            else :
                print(", ", end="")
                print(*listOfMatkul[course], end="")
    print("")

```

BAB III

SKRINSHUT DAN PENJELASAN

Soal		Penjelasan
Kalkulus. Fisdas. Kimdas. TTKI. Bing, TTKI. AEI, Bing, TTKI. KWN, Bing, TTKI. Kalkulus2, Kalkulus. Fisdas2, Fisdas. Kimdas2, Kimdas. Daspro, Kalkulus, Fisdas, Kimdas. PAR, Kalkulus, Fisdas, Kimdas. Matdis, Kalkulus2. Algeo, Kalkulus2.	Logkom, Kalkulus2. Orkom, Daspro. Alstrukdat, Daspro. OOP, Alstrukdat. Basdat, Logkom, Algeo, Matdis. Stima, Algeo, Matdis. MBD, Basdat. OS, Orkom. Jarkom, OS. AI, Logkom, Stima, OOP. IOT, AI, Jarkom. Bigdata, MBD, Basdat. TA, IOT, Bigdata. SG, KWN. Wisuda, TA.	<p>Testcase-1</p> <p><i>Testcase</i> ini dibuat dengan jumlah mata kuliah dan prequisite lebih banyak dari <i>testcase</i> yang diberika asisten. Urutan penulisan di file .txt juga sudah urut sehingga program akan otomatis membaca mata kuliah dasar terlebih dahulu.</p> <p>Jumlah semester yang dibutuhkan juga tepat delapan.</p> <p>Kesimpulan Program dapat menyelesaikan persoalan dengan matkul yang lebih banyak dari <i>testcase</i> dari asisten. Program bisa menyelesaikan persoalan dengan total semester yang dibutuhkan tepat delapan.</p>
<p style="text-align: center;">Skrinshut-01</p> 		
AI, Logkom, Stima, OOP. Bigdata, MBD, Basdat. TA, IOT, Bigdata. TTKI. Bing, TTKI. AEI, Bing, TTKI. Kalkulus2, Kalkulus. OS, Orkom. IOT, AI, Jarkom. Fisdas2, Fisdas.	Alstrukdat, Daspro. OOP, Alstrukdat. Basdat, Logkom, Algeo, Matdis. Stima, Algeo, Matdis. MBD, Basdat. Jarkom, OS. Matdis, Kalkulus2. Daspro, Kalkulus, Fisdas, Kimdas.	<p>Testcase-2</p> <p><i>Testcase</i> ini sama dengan <i>testcase-1</i> namun urutan penulisan dalam file .txt diacak.</p> <p>Kesimpulan: Program dapat menyelesaikan persoalan tanpa mempedulikan urutan penulisan dalam file .txt</p>

PAR, Kalkulus, Fisdas, Kimdas. Kimdas. Algeo, Kalkulus2. Fisdas. Orkom, Daspro.	SG, KWN. Kimdas2, Kimdas. Wisuda, TA. KWN, Bing, TTKI. Logkom, Kalkulus2. Kalkulus.	
--	--	--

Skrinshut-02

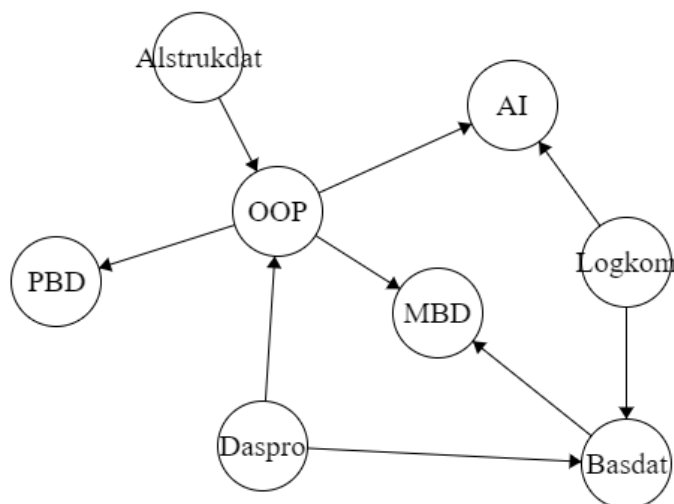
Semester I: Tata Tulis Karya Ilmiah, Kimia Dasar, Fisika Dasar, Kalkulus
Semester II: Bahasa Inggris, Kalkulus 2, Fisika Dasar 2, Pengantar Analisis Rangkaian, Dasar Pemrograman, Kimia Dasar 2
Semester III: Agama dan Etika Islam, Aljabar Linier dan Geometri, Organisasi dan Arsitektur Komputer, Algoritma & Struktur Data, Matematika Diskrit, Kewarganegaraan, Logika Komputasional
Semester IV: Sistem Operasi, Pemrograman Berorientasi Objek, Basis Data, Strategi Algoritma, Studium Generale
Semester V: Inteligensi Buatan, Manajemen Basis Data, Jaringan Komputer
Semester VI: Teknologi Big Data, Pengembangan Sistem IoT
Semester VII: Tugas Akhir
Semester VIII: Wisuda

MBD, OOP, Basdat. Basdat, Daspro, Logkom. Daspro. OOP, Alstrukdat, Daspro.	AI, OOP, Logkom. PBD, OOP. Logkom. Alstrukdat.	Testcase-3 <i>Testcase</i> ini dibuat sedemikian hingga semester yang dibutuhkan kurang dari delapan . Kesimpulan Program bisa menyelesaikan persoalan dengan total semester yang dibutuhkan kurang dari delapan .
---	---	---

Skrinshut-03

Hooray!! Kamu bisa lulus tanpa harus sampai 8 Semester :)
Semester I: Dasar Pemrograman, Logika Komputasional, Algoritma & Struktur Data
Semester II: Basis Data, Pemrograman Berorientasi Objek
Semester III: Manajemen Basis Data, Inteligensi Buatan, Pengembangan Aplikasi pada Platform Khusus

Soal Dalam Bentuk Graf



Sister, OS. OS, OOP, Stima, Orkom. PBD, OS, Orkom. Grafkom, PBD, Orkom. Orkom. Stima. OOP, Alstrukdat, Logkom. Logkom.	TA, AI. AI, Orkom, Socif. Socif, Grafkom. Wisuda, PAR, Logkom, TA. PAR. Daspro. Alstrukdat, Daspro, PAR.	Testcase-4 Testcase ini dibuat sedemikian hingga semester yang dibutuhkan lebih dari delapan. Kesimpulan Program bisa menyelesaikan persoalan dengan total semester yang dibutuhkan lebih dari delapan. Akan ditambahkan peringatan apabila jumlah semesternya lebih dari delapan.
---	---	---

Skrinshut-04

Warning!! Untuk dapat menyelesaikan semua matkul dibutuhkan lebih dari 8 Semester

Semester I: Organisasi dan Arsitektur Komputer, Strategi Algoritma, Logika Komputasional, Pengantar Analisis Rangkaian, Dasar Pemrograman

Semester II: Algoritma & Struktur Data

Semester III: Pemrograman Berorientasi Objek

Semester IV: Sistem Operasi

Semester V: Sistem Paralel dan Terdistribusi, Pengembangan Aplikasi pada Platform Khusus

Semester VI: Grafika Komputer

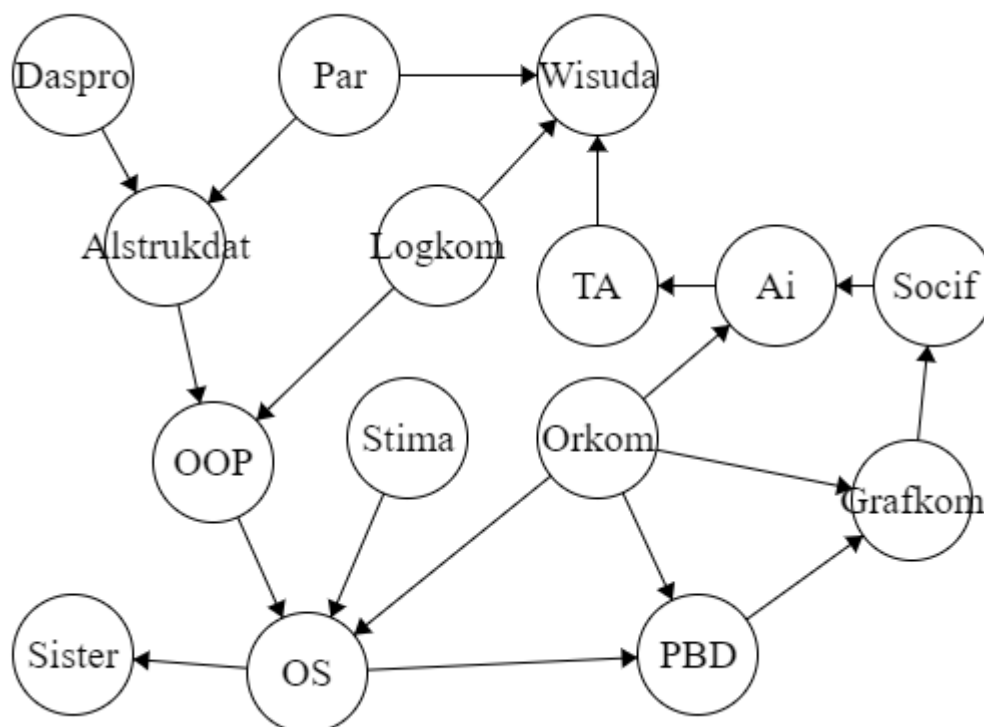
Semester VII: Socio Informatika dan Profesionalisme

Semester VIII: Inteligensi Buatan

Semester IX: Tugas Akhir

Semester X: Wisuda

Soal Dalam Bentuk Graf

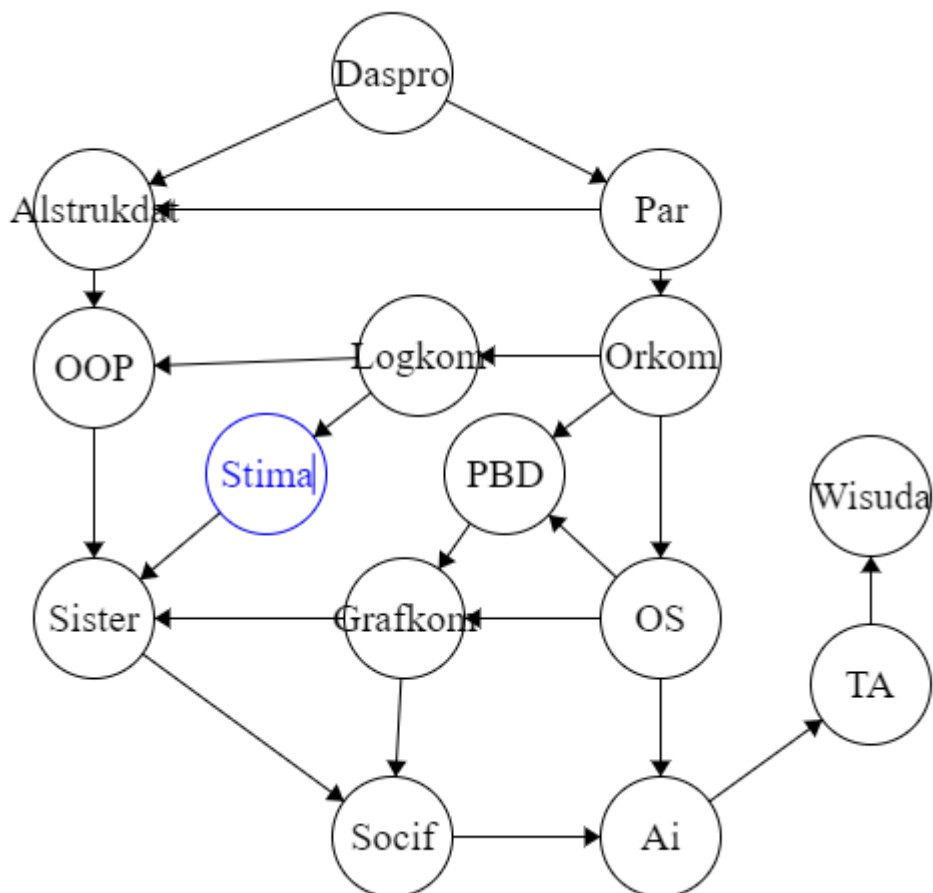


Daspro. Alstrukdat, PAR, Daspro. PAR, Daspro. OOP, Alstrukdat, Stima. Logkom, Orkom. Orkom, PAR. Stima, Logkom. PBD, Orkom, OS.	Sister, OOP, Stima, Grafkomp. Grafkomp, PBD, OS. OS, Orkom. Socif, Grafkomp, Sister. AI, OS, Socif. TA, AI. Wisuda, TA.	Testcase-5 <i>Testcase</i> ini dibuat sedemikian hingga lebih rumit bentuk graf nya daripada 4 <i>testcase</i> diatas. Kesimpulan Tidak ada kesimpulan spesifik, hanya untuk mengetes ke <i>valid</i> -an program saja.
--	--	--

Skrinshut-05

```
Warning!! Untuk dapat menyelesaikan semua matkul dibutuhkan lebih dari 8 Semester
Semester I: Dasar Pemrograman
Semester II: Pengantar Analisis Rangkaian
Semester III: Algoritma & Struktur Data, Organisasi dan Arsitektur Komputer
Semester IV: Logika Komputasional, Sistem Operasi
Semester V: Strategi Algoritma, Pengembangan Aplikasi pada Platform Khusus
Semester VI: Pemrograman Berorientasi Objek, Grafika Komputer
Semester VII: Sistem Paralel dan Terdistribusi
Semester VIII: Socio Informatika dan Profesionalisme
Semester IX: Inteligensi Buatan
Semester X: Tugas Akhir
Semester XI: Wisuda
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Algoritma\Tucil + Tubes\Tucil2_13519043>
```

Soal Dalam Bentuk Graf



AEI. Alstrukdat. Daspro. Kalkulus2. Fisdas2. TTKI. OOP, Alstrukdat. Orkom, AEI. Logkom, OOP, Orkom. Bigdata, OOP, Stima, ML, Daspro, Matdis.	Matdis, Kalkulus2. Algeo, Matdis, PAR. PAR, Fisdas2. NLP, Orkom, IOT, AI, PAR, TTKI. Stima, OOP, Logkom. IOT, Orkom, Logkom. ML, Matdis, Algeo. AI, Algeo, PAR. Wisuda, Stima, IOT, Bigdata, ML, AI, NLP.	Testcase-6 Testcase ini dibuat sedemikian hingga lebih rumit bentuk graf nya daripada 5 testcase diatas. Kesimpulan Tidak ada kesimpulan spesifik, hanya untuk mengetes ke <i>valid</i> -an program saja.
--	---	--

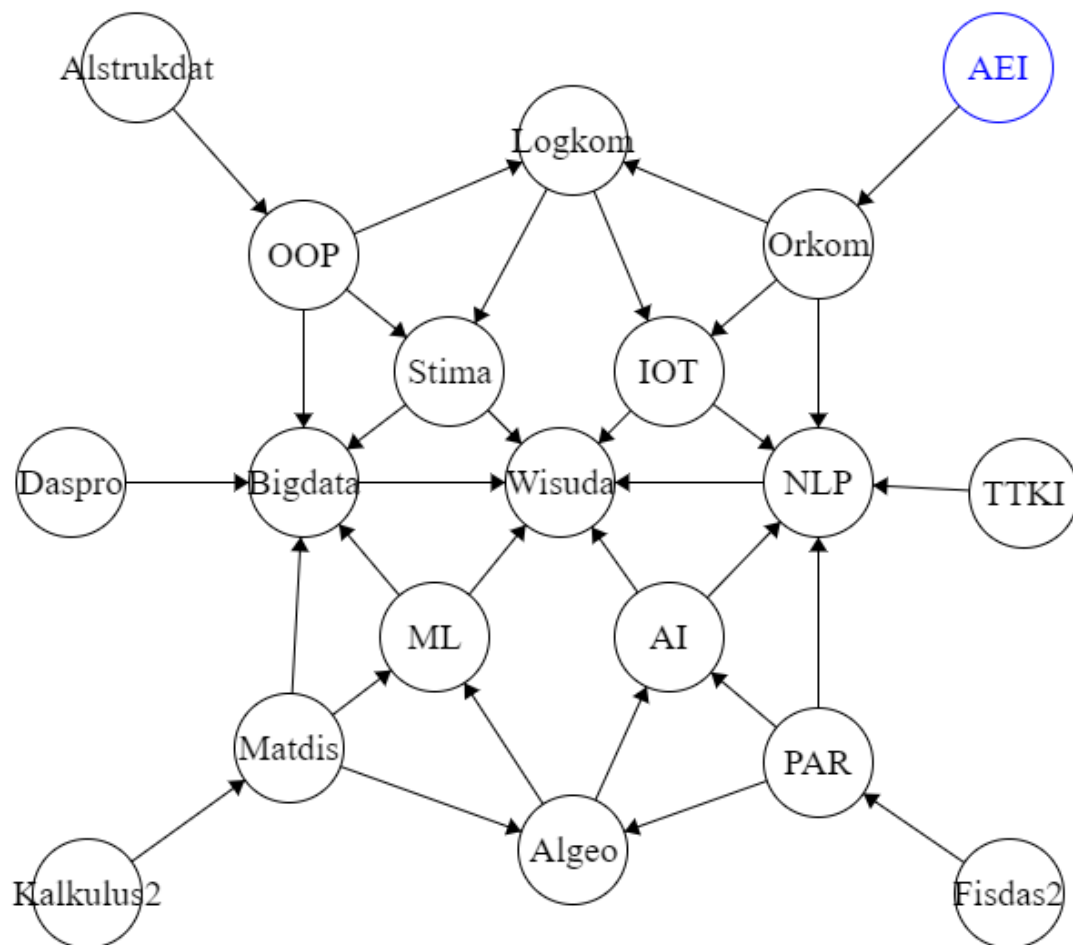
Skrinshut-06

```

Hooray!! Kamu bisa lulus tanpa harus sampai 8 Semester :)
Semester I: Agama dan Etika Islam, Algoritma & Struktur Data, Dasar Pemrograman, Kalkulus 2, Fisika D
asar 2, Tata Tulis Karya Ilmiah
Semester II: Pemrograman Berorientasi Objek, Organisasi dan Arsitektur Komputer, Matematika Diskrit,
Pengantar Analisis Rangkaian
Semester III: Logika Komputasional, Aljabar Linier dan Geometri
Semester IV: Strategi Algoritma, Pengembangan Sistem IoT, Pembelajaran Mesin, Inteligensi Buatan
Semester V: Teknologi Big Data, Pemrosesan Bahasa Alami
Semester VI: Wisuda
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Algoritma\Tucil + Tubes\Tucil2_13519043>

```

Soal Dalam Bentuk Graf

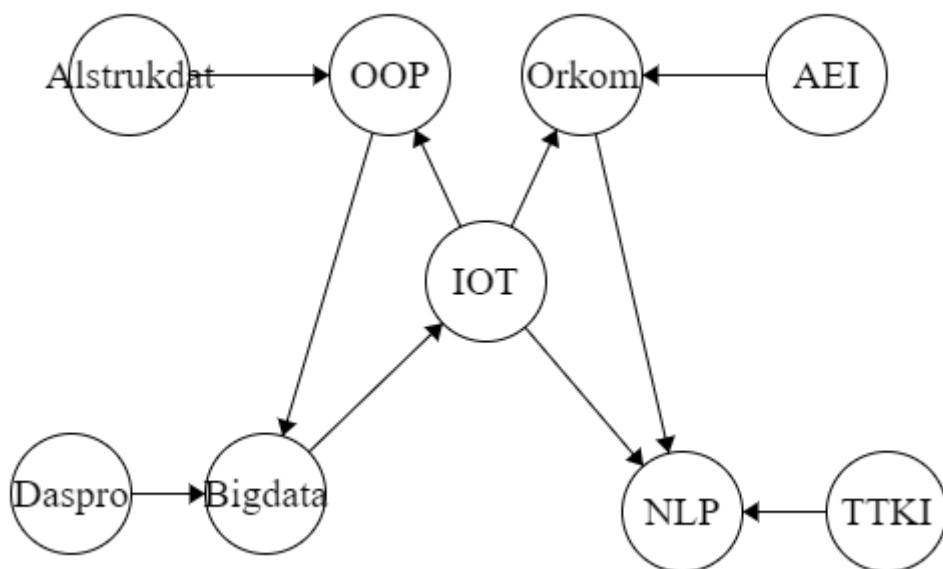


Alstrukdat. AEI. Daspro. TTKI. OOP, Alstrukdat, IOT.	Orkom, IOT, AEI. Bigdata, Daspro, OOP. NLP, TTKI Orkom, IOT. IOT, Bigdata.	Testcase-7 <i>Testcase</i> ini dibuat sedemikian hingga terdapat siklus sehingga graf bukannya graf DAG. Kesimpulan Program akan menolak input dari user yang bukan merupakan DAG .
--	---	--

Skrinshut-07

```
Melakukan Decrease and Conquer ~ 9 node yang harus dikunjungi
Melakukan Decrease and Conquer ~ 8 node yang harus dikunjungi
Melakukan Decrease and Conquer ~ 7 node yang harus dikunjungi
Melakukan Decrease and Conquer ~ 6 node yang harus dikunjungi
Melakukan Decrease and Conquer ~ 5 node yang harus dikunjungi
Siklus ditemukan!!!
Anda tidak memasukkan DAG
PS D:\Tugas Kuliah\IF\Semester 4\IF2211 - Strategi Algoritma\Tucil + Tubes\Tucil2_13519043>
```

Soal Dalam Bentuk Graf



AEI. Alstrukdat. Daspro. Kalkulus2. Fisdas2. TTKI. OOP, Alstrukdat. Orkom, AEI. Logkom, OOP, Orkom.	PAR, Fisdas2. NLP, Orkom, IOT, AI, PAR, TTKI. Stima, OOP, Logkom, IOT. IOT, Orkom, Logkom, Wisuda. ML, Matdis, Algeo, Wisuda.	Testcase-8 <i>Testcase</i> ini dibuat sedemikian hingga terdapat siklus sehingga graf bukannya graf DAG. Kesimpulan Program akan menolak input dari user yang bukan merupakan DAG .
---	---	--

Bigdata, OOP, Stima, ML,
Daspro, Matdis.
Matdis, Kalkulus2.
Algeo, Matdis, PAR.

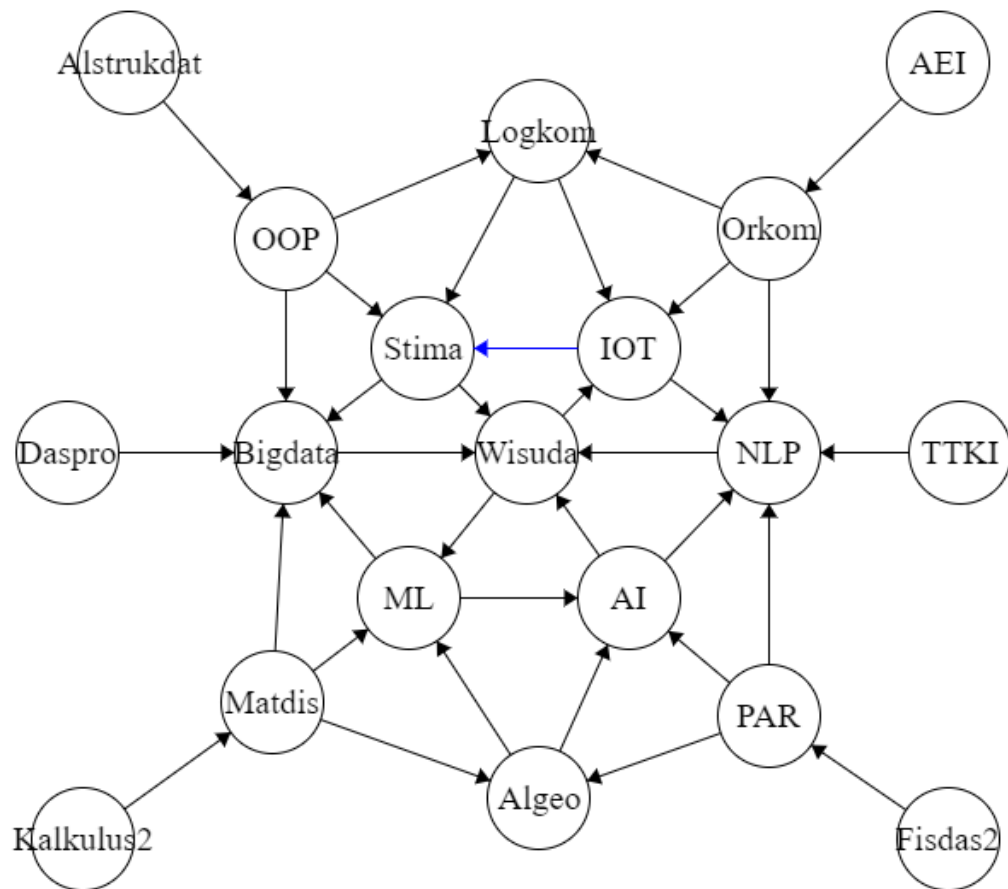
AI, Algeo, PAR, ML.
Wisuda, Stima, Bigdata,
AI, NLP.

Skrinshut-08

Melakukan Decrease and Conquer ~ 8 node yang harus dikunjungi
Melakukan Decrease and Conquer ~ 7 node yang harus dikunjungi
Siklus ditemukan!!!
Anda tidak memasukkan DAG

Ss hasil dipotong karena terlalu panjang

Soal Dalam Bentuk Graf



BAB IV

LAMPIRAN

Poin	Ya	Tidak
1. Program berhasil dikompilasi	<input checked="" type="radio"/>	<input type="radio"/>
2. Program berhasil <i>running</i>	<input checked="" type="radio"/>	<input type="radio"/>
3. Program dapat menerima berkas input dan menuliskan output.	<input checked="" type="radio"/>	<input type="radio"/>
4. Luaran sudah benar untuk semua kasus input.	<input checked="" type="radio"/>	<input type="radio"/>

Link Github

<https://drive.google.com/drive/folders/1dn29tPtcQHrIrWUzwtST9DKmIBUx8bwk?usp=sharing>

Referensi

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2012-2013/Makalah2012/Makalah-IF3051-2012-038.pdf>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2012-2013/Makalah2012/Makalah-IF3051-2012-060.pdf>