

**PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL SORT*
(PENERAPAN *DECREASE AND CONQUER*)**

Laporan Tugas Kecil 2 IF 2211 Strategi Algoritma
Semester II Tahun 2020/2021



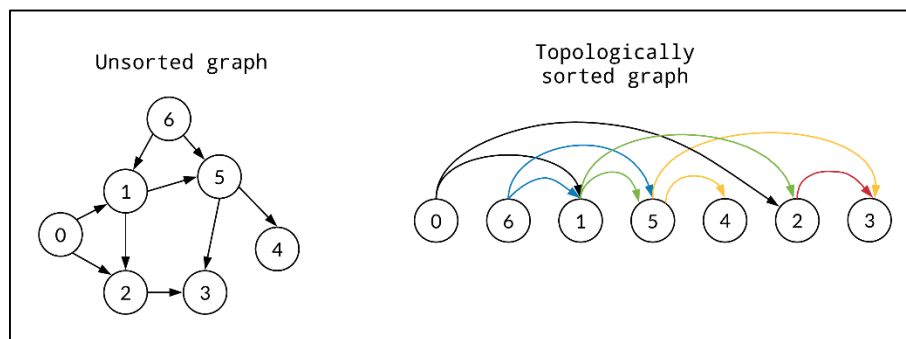
Oleh:

Dionisius Darryl Hermansyah
13519058 / Kelas 02

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

I. ALGORITMA *TOPOLOGICAL SORT* DAN *DECREASE AND CONQUER*

Topological sort merupakan sebuah algoritma yang umumnya digunakan untuk melakukan *topological ordering*. *Topological ordering* adalah kegiatan mengurutkan objek-objek (biasanya direpresentasikan dalam bentuk graf berarah), dimana, untuk setiap sisi (*edge*) dari simpul (*vertex*) A ke simpul B, simpul A harus berada pada urutan di depan simpul B. Contoh dari permasalahan *topological ordering* yang dapat diselesaikan menggunakan *topological sort* adalah pengambilan mata kuliah yang memilih mata kuliah prasyarat. Pada algoritma *topological sort*, ada sebuah simpul yang harus menjadi *source vertex*, dimana, simpul tersebut memiliki derajat masuk 0. Hal ini juga dijelaskan secara visual dengan adanya *directed acyclic graphs* (DAG) yang menyatakan bahwa *topological sort* hanya dapat diimplementasikan pada graf yang valid yaitu graf yang tidak memiliki sirkuit.



Gambar 1. Contoh penerapan algoritma *topological sort* pada sebuah *directed acyclic graph* (DAG)

Algoritma *topological sort* merupakan salah satu penerapan dari algoritma *decrease and conquer*. Algoritma *decrease and conquer* memanfaatkan metode perancangan algoritma dengan umumnya mereduksi persoalan menjadi dua upa-persoalan yang lebih kecil, tetapi selanjutnya melakukan seleksi dan hanya memproses satu sub-persoalan saja. Berikut merupakan penjelasan lebih lanjut mengenai langkah-langkah algoritma *topological sort* secara umum, serta kaitannya dengan algoritma *decrease and conquer*:

1. Asumsikan graf telah dibaca dari file .txt dan graf merupakan DAG. Pada kasus ini diambil contoh graf pada gambar 1.

2. Hitung seluruh derajat masuk (*in-degree*) dari masing-masing simpul (*vertex*) yang ada pada graf. Derajat masuk merupakan jumlah busur (*edge*) yang masuk menuju ke simpul terkait. Pada gambar 1, derajat masing-masing simpul adalah:

Simpul 0: 0

Simpul 4: 1

Simpul 1: 2

Simpul 5: 2

Simpul 2: 1

Simpul 6: 0

Simpul 3: 2

3. Pilih simpul dengan derajat masuk paling rendah (Umumnya pada iterasi pertama adalah 0). Dari contoh, didapatkan simpul 0 dan 6 (*Divide*).
4. Hapus simpul beserta semua busur yang keluar dari simpul tersebut, kemudian kurangi derajat simpul yang berhubungan dengan simpul terkait (*Conquer*). Setelah itu, graf akan menjadi:

Simpul 1: 0

Simpul 4: 1

Simpul 2: 1

Simpul 5: 1

Simpul 3: 2

5. Ulangi langkah 3 dan 4 hingga semua simpul pada graf telah dipilih dan terhapus (kondisi graf kosong, didefinisikan sebagai graf yang tidak memiliki simpul dan busur).
6. Keluarkan hasil akhir pemilihan mata kuliah berdasarkan *topological sort*, dalam hal ini:

Semester I : 0, 6

Semester II : 1

Semester III : 2, 5

Semester IV : 3, 4

Algoritma *topological sort* termasuk ke dalam algoritma *divide and conquer*. Hal ini karena algoritma ini memanfaatkan strategi yang serupa dengan algoritma *divide and conquer*, yaitu membagi persoalan ke upa-persoalan yang lebih kecil, menyeleksi upa-persoalan dan memprosesnya, serta meninggalkan upa-persoalan lainnya sehingga tidak diproses. Pada contoh pemilihan simpul graf di atas, mula-mulanya algoritma *topological sort* akan membagi (*divide*) himpunan simpul menjadi 2 upa-himpunan yaitu simpul dengan derajat masuk terkecil (misalnya A) dan simpul lainnya (misalnya B). Upa-himpunan simpul dengan derajat masuk terkecil akan dihilangkan dari graf dan diabaikan (*conquer*) dengan menghapus setiap simpul yang ada serta busur yang terkait dengan simpul tersebut. Setelah itu, hanya upa-himpunan B yang akan diproses

dengan strategi yang sama dengan pemrosesan sebelumnya. *Topological sort* akan berakhir saat graf sudah kosong, yaitu setelah graf tidak memiliki simpul dan bidang apapun lagi. Jadi, berdasarkan penjelasan tersebut, maka *topological sort* merupakan sebuah aplikasi nyata dari algoritma *divide and conquer* dalam kehidupan sehari-hari, contohnya dalam penyusunan jadwal perkuliahan.

II. SOURCE CODE PROGRAM

Dalam pembuatan program, diambil beberapa asumsi sebagai berikut:

- Input data uji yang dimasukkan sudah pasti merupakan DAG jika dikonversi ke dalam graf.
- Input data mata kuliah sudah pasti dapat diselesaikan dalam 8 semester atau kurang.
- Masa kuliah (semester) mahasiswa hanya ada 8 semester.
- Dalam satu semester, mahasiswa dapat mengambil jumlah mata kuliah tanpa batas.

Program ini dibuat menggunakan bahasa Python secara modular. Ada 2 modul utama yaitu 13519058-main.py dan 13519058-graph.py. Program utama diletakkan pada 13519058-main.py, sedangkan 13519058-graph.py berisi definisi dan deskripsi kelas graph beserta segala fungsi yang terkait dengannya. *Source code* yang dilampirkan telah disesuaikan kode, komentar, dan indentasinya agar lebih rapi dan singkat. Berikut merupakan *source code* dari program:

[illegible]

```

show_steps_input = str(input("Apakah anda ingin menampilkan langkah-langkah penyelesaian secara detail? (Y/N): "))

if (show_steps_input == "Y" or show_steps_input == "y"):
    show_steps = True

while (not exit):
    result = []      # Hasil pengambilan mata kuliah
    semester = 1    # Semester tracker

    # Meminta input nama file
    # Input dalam bentuk nama file saja, tanpa path dan ekstensi
    file_name = str(input("\nMasukkan nama file (tanpa path / ekstensi): "))

    # Membuat graph dari file teks terkait
    g2 = graphm.makeGraphFromTxt(file_name)

    print("\nGraf yang anda masukkan: \n")
    g2.printGraph()

    # Lakukan iterasi pencarian mata kuliah selama graph belum kosong
    while (not g2.isGraphEmpty()):
        lowest_degree_v = graphm.getLowestDegree(g2) # Ambil vertex lowest in-degree

        if (show_steps):
            print(f"\nSemester {semester_romawi[semester]} mengambil: {'',
'.join(lowest_degree_v)}") # Output matakuliah yang dapat diambil

            # Manipulasi graph dengan mendelete vertex terkait
            curr = []
            for v in lowest_degree_v:
                curr.append(v)
                graphm.removeAllEdgeFrom(g2, v)
            result.append(curr)

            if (show_steps):
                # Mencetak graph
                print("\nGraph:")
                g2.printGraph()

                # Mencetak sisa jumlah mata kuliah
                print(f"Jumlah mata kuliah tersisa: {g2.V}")

            semester += 1

    # Output hasil akhir
    print("\nSelesai merencanakan pengambilan mata kuliah.\n")

    for i in range(len(result)):
        print(f"Semester {semester_romawi[i+1]}: ", end="")
        print(result[i][0], end="")
        for j in range(1, len(result[i])):
            print(f", {result[i][j]}", end="")
        print()

    # Pilihan exit
    exit_choice = str(input("\nApakah anda ingin memproses file lain? (Y/N): "))
    if (exit_choice == "N" or exit_choice == "n"):
        exit = True

print("\nTerima kasih telah menggunakan program ini!")

```

13519058-graph.py

```
"""
Definisi kelas Graph dan method yang berhubungan dengan pemrosesan graph
"""

class Graph:
    def __init__(self, vertices):
        self.E = dict()      # Edges / Kumpulan edge
        self.V = vertices    # Vertices / Kumpulan vertex

    # Menambahkan sebuah edge ke Graph
    def addEdge(self, e, v):
        if v == None: # Vertex tanpa derajat masuk
            self.E[e] = []
        else:
            if e not in self.E.keys():
                self.E[e] = [v]
            else:
                self.E[e].append(v)

    # Mengoutput struktur graph dengan format contoh:
    # A -> B -> C
    # D
    # E -> F
    def printGraph(self):
        for k, v in self.E.items():
            print(str(k) + " ", end="")
            for v_item in v:
                print("->", end="")
                print(f" {v_item} ", end="")
            print()

    # Cek apakah sebuah graph kosong
    # Graph kosong jika vertices dan edge kosong (0)
    def isGraphEmpty(self):
        return self.V == 0

# Membaca data graph dari file .txt dan mereturn sebuah Graph
# Path default : "./test"
# file_name tanpa dituliskan ekstensi (.txt)
def makeGraphFromTxt(file_name):
    # Variabel
    res = []
    res_cleaned = []
    all_vertex = list()

    # Open dan read file
    f = open(f"./test/{file_name}.txt", "r")

    # Append setiap line ke array result
    for line in f:
        res.append(line.split(","))

    # Cleaning tanda ' ', '.' dan '\n'
    for el in res:
        res_cleaned.append([])
        for i in range(len(el)):
            el_cleaned = el[i].replace(" ", "").replace(".", "").replace("\n", "")
            res_cleaned[len(res_cleaned)-1].append(el_cleaned)

    # Menyimpan semua vertex yang ada
    for el in res_cleaned:
```

```

        for v in el:
            all_vertex.append(v)

    all_vertex = set(all_vertex)

    # Membuat sebuah graph dengan vertex sebanyak panjang himpunan all_vertex
    graph = Graph(len(all_vertex))

    # Adding setiap edge yang ada
    for el in res_cleaned:
        if (len(el) == 1): # Vertex tanpa derajat masuk
            graph.addEdge(el[0], None)
        else:
            for i in range(1, len(el)):
                graph.addEdge(el[0], el[i])

    return graph

# Mereturn vertex dengan derajat masuk terendah dari sebuah graph
def getLowestDegree(graph):
    d = {}

    # Mengambil value (derajat masuk) setiap vertex
    for k, v in graph.E.items():
        d[k] = len(v)

    # Mencari value (derajat masuk) terendah
    min_v = min(d.values())
    res = [k for k, v in d.items() if v==min_v]

    return res

# Menghapus semua hubungan edge pada graph p dari semua vertex ke vertex x
def removeAllEdgeFrom(graph, x):
    # Jika vertex k terhubung dengan x, hapus edge antara k dan x
    for k, v in graph.E.items():
        if x in v:
            v.remove(x)

    # Jika vertex tidak punya derajat masuk, maka hapus vertex
    if len(graph.E[x]) == 0:
        graph.E.pop(x)
        graph.V -= 1

```

III. INPUT DAN OUTPUT

Program tugas kecil 2 ini diberi nama *Course Organizer* atau CourseOrg. Pada menu awal program, pengguna dapat memilih apakah ingin menampilkan langkah-langkah penyelesaian menggunakan algoritma *topological sort* secara detail atau hanya ingin menampilkan hasil akhirnya. Berikut ini merupakan contoh tampilan awal program:

Tabel 1. menunjukkan hasil *test case* yang telah dipastikan merupakan DAG.

Tabel 1. Input dan output program

No	Input	Output
1	C1, C3. C2, C1, C4. C3. C4, C1, C3. C5, C2, C4.	<pre> Graf yang anda masukkan: C1 -> C3 C2 -> C1 -> C4 C3 C4 -> C1 -> C3 C5 -> C2 -> C4 Selesai merencanakan pengambilan mata kuliah Semester I: C3 Semester II: C1 Semester III: C4 Semester IV: C2 Semester V: C5 </pre>
2	C1, C2. C2. C3. C4, C1, C3. C5, C3. C6.	<pre> Graf yang anda masukkan: C1 -> C2 C2 C3 C4 -> C1 -> C3 C5 -> C3 C6 Selesai merencanakan pengambilan mata kuliah. Semester I: C2, C3, C6 Semester II: C1, C5 Semester III: C4 </pre>
3	A, B, C. B. C, D, E, F. D, E.	

	<p>E, F.</p> <p>F, B.</p> <p>G, H, I.</p> <p>H, A.</p> <p>I, B.</p> <p>J, K.</p> <p>K, F, D.</p>	<p>Graf yang anda masukkan:</p> <pre> A -> B -> C B C -> D -> E -> F D -> E E -> F F -> B G -> H -> I H -> A I -> B J -> K K -> F -> D </pre> <p>Selesai merencanakan pengambilan mata kuliah.</p> <p>Semester I: B Semester II: F, I Semester III: E Semester IV: D Semester V: C, K Semester VI: A, J Semester VII: H Semester VIII: G</p>
4	<p>Matematika_Lanjutan, Matematika_Dasar.</p> <p>Fisika_Lanjutan, Fisika_Dasar.</p> <p>Matematika_Dasar.</p> <p>Fisika_Dasar.</p> <p>Kalkulus, Matematika_Lanjutan, Matematika_Dasar.</p> <p>Kalkulus_Lanjutan, Kalkulus, Fisika_Lanjutan,</p> <p>Matematika_Lanjutan, Matematika_Dasar.</p> <p>Biologi_Dasar.</p> <p>Kimia_Dasar.</p> <p>Biokimia, Biologi_Dasar, Kimia_Dasar.</p> <p>Kerja_Praktek, Biokimia, Kalkulus_Lanjutan.</p>	<p>Graf yang anda masukkan:</p> <pre> Matematika_Lanjutan -> Matematika_Dasar Fisika_Lanjutan -> Fisika_Dasar Matematika_Dasar Fisika_Dasar Kalkulus -> Matematika_Lanjutan -> Matematika_Dasar Kalkulus_Lanjutan -> Kalkulus -> Fisika_Lanjutan -> Matematika_Lanjutan -> Matematika_Dasar Biologi_Dasar Kimia_Dasar Biokimia -> Biologi_Dasar -> Kimia_Dasar Kerja_Praktek -> Biokimia -> Kalkulus_Lanjutan </pre> <p>Selesai merencanakan pengambilan mata kuliah.</p> <p>Semester I: Matematika_Dasar, Fisika_Dasar, Biologi_Dasar, Kimia_Dasar Semester II: Matematika_Lanjutan, Fisika_Lanjutan, Biokimia Semester III: Kalkulus Semester IV: Kalkulus_Lanjutan Semester V: Kerja_Praktek</p>
5	<p>A.</p> <p>B.</p> <p>C.</p> <p>D.</p> <p>E.</p> <p>F.</p> <p>G.</p> <p>H.</p>	<p>Graf yang anda masukkan:</p> <pre> A B C D E F G H </pre> <p>Selesai merencanakan pengambilan mata kuliah.</p> <p>Semester I: A, B, C, D, E, F, G, H</p>

6	<p>Matematika.</p> <p>Matdis, Matematika.</p> <p>Stima, Matdis.</p> <p>Algeo.</p> <p>Logkom.</p> <p>Pengkom.</p> <p>Alstrukdat, Pengkom.</p> <p>OOP, Alstrukdat.</p> <p>Orkom, Pengkom.</p> <p>OS, Orkom.</p> <p>Basdat, Alstrukdat.</p> <p>MBD, Basdat, Alstrukdat.</p> <p>AI, ML, Matdis.</p> <p>ML, Stima, Matdis.</p> <p>RPL.</p> <p>KP, AI, RPL, MBD, OS.</p> <p>KP2, KP.</p> <p>TA, KP2, KP.</p>	<pre> Graf yang anda masukkan: Matematika Matdis -> Matematika Stima -> Matdis Algeo Logkom Pengkom Alstrukdat -> Pengkom OOP -> Alstrukdat Orkom -> Pengkom OS -> Orkom Basdat -> Alstrukdat MBD -> Basdat -> Alstrukdat AI -> ML -> Matdis ML -> Stima -> Matdis RPL KP -> AI -> RPL -> MBD -> OS KP2 -> KP TA -> KP2 -> KP Selesai merencanakan pengambilan mata kuliah. Semester I: Matematika, Algeo, Logkom, Pengkom, RPL Semester II: Matdis, Alstrukdat, Orkom Semester III: Stima, OOP, OS, Basdat Semester IV: MBD, ML Semester V: AI Semester VI: KP Semester VII: KP2 Semester VIII: TA </pre>
7	<p>1, 2.</p> <p>3, 4.</p> <p>2, 3.</p> <p>4.</p> <p>5, 1.</p> <p>6, 3, 2.</p> <p>7.</p> <p>8, 7, 9.</p> <p>9, 2.</p>	<pre> Graf yang anda masukkan: 1 -> 2 3 -> 4 2 -> 3 4 5 -> 1 6 -> 3 -> 2 7 8 -> 7 -> 9 9 -> 2 Selesai merencanakan pengambilan mata kuliah. Semester I: 4, 7 Semester II: 3 Semester III: 2 Semester IV: 1, 6, 9 Semester V: 5, 8 </pre>

8	A0. A1, A0, A6. A2, A0, A1. A3, A2, A5. A4, A5. A5, A6, A1. A6.	<pre> Graf yang anda masukkan: A0 A1 -> A0 -> A6 A2 -> A0 -> A1 A3 -> A2 -> A5 A4 -> A5 A5 -> A6 -> A1 A6 Selesai merencanakan pengambilan mata kuliah. Semester I: A0, A6 Semester II: A1 Semester III: A2, A5 Semester IV: A3, A4 </pre>
---	---	--

Evaluasi program secara umum ditunjukkan oleh tabel 2, dimana, program telah dapat memenuhi seluruh poin persyaratan yang ada.

Tabel 2. Evaluasi program

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

IV. ALAMAT GITHUB

Berikut merupakan alamat *repository* GitHub dari *source code* yang digunakan, laporan, program dalam bentuk file Python (.py), beserta file test. Untuk menguji program, harus dipastikan bahwa file test berada dalam direktori ./test/.

<https://github.com/dionisiusdh/courses-organizer>

V. DAFTAR PUSTAKA

- [1] Levitin, A. 2012. *Introduction to the Design & Analysis of Algorithms, 3rd Edition*. London: Pearson.
- [2] Munir, R. 2021. *Algoritma Decrease and Conquer*. Bandung: Institut Teknologi Bandung.
- [3] Svirin, A. 2021. *Topological Sorting*. Dilansir dari www.math24.net pada 24 Februari 2021.