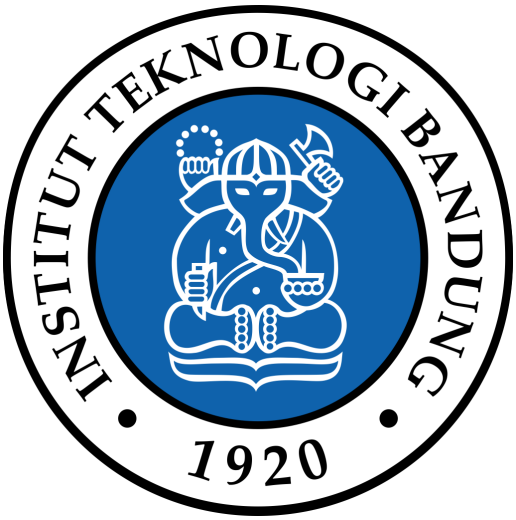
LAPORAN TUGAS KECIL 2 STRATEGI ALGORITMA

Penyusunan Rencana Kuliah dengan Topological Sort (Decrease and Conquer)



Oleh :

Made Kharisma Jagaddhita (13519176)

Institut Teknologi Bandung
2021

A. Topological Sort

Topological sort adalah pengurutan simpul-simpul dari graf berarah sedemikian rupa sehingga untuk setiap garis penghubung u-v dari simpul u ke simpul v, simpul u berada sebelum simpul v dalam pengurutannya. *Topological sort* hanya dapat dilakukan jika graf yang ingin diurutkan merupakan *Directed Acyclic Graph* (DAG), yaitu graf yang tidak memiliki siklus. Hasil pengurutan dari *topological sort* untuk satu DAG dapat berbeda (tidak unik).

Berikut merupakan cara kerja algoritma dari *topological sort*:

1. Hitung semua derajat-masuk (in-degree) setiap simpul, yaitu banyaknya busur yang masuk pada simpul tersebut.
2. Pilih sembarang simpul yang memiliki derajat-masuk 0.
3. Ambil simpul tersebut, dan hilangkan simpul tersebut beserta semua busur yang keluar dari simpul tersebut pada graf, dan kurangi derajat simpul yang berhubungan dengan simpul tersebut dengan 1.
4. Ulangi langkah (2) dan (3) hingga semua simpul pada DAG terpilih.

Topological sorting merupakan implementasi dari algoritma *Decrease and Conquer. Decrease and Conquer* adalah perancangan algoritma dengan mereduksi persoalan menjadi dua upa-persoalan (sub-problem) yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja. Dalam *topological sorting* persoalan direduksi menjadi dua upa-persoalan. Satu upa-persoalan merupakan simpul yang memiliki derajat-masuk 0 yang merupakan solusi, sedangkan upa-persoalan lainnya merupakan sebuah graf yang simpul berderajat-masuk 0-nya “direduksi”. Upa-persoalan yang berbentuk graf akan direduksi terus-menerus hingga tidak lagi terdapat simpul (*Decrease*). Kemudian solusi-solusi dari setiap upa-persoalan akan digabungkan sehingga membentuk *topological ordering* (*Conquer*).

B. *Source code*

Program memakai bahasa python dan diberi nama matkul.py. *Source code* dapat diakses di link ini: https://github.com/kharismajd/Tucil2_13519176

```python
### Fungsi-fungsi tambahan ###
def add_vertice(graph, vertice_name):
    # Menambahkan simpul baru pada graph
    # Jika nama simpul sudah ada, akan mengeluarkan pesan
    if (vertice_name not in graph):
        graph[vertice_name] = []
    else:
        print("Sudah ada simpul " + vertice_name + " pada graph")

```

```python
def add_edge(graph, from_vertice, to_vertice):
    # Menambahkan edge baru pada graph
    # Jika tidak simpul bernama from_vertice atau to_vertice pada graph,
    # akan mengeluarkan pesan
    if ((from_vertice in graph) and (to_vertice in graph)):
        graph[from_vertice].append(to_vertice)
    else:
        print("Tidak ada simpul " + from_vertice + " atau " + to_vertice + " pada graph")

```

```python
def delete_vertice(graph, vertices):
    # Menghapus simpul beserta busur yang keluar dari simpul tersebut dari graph.
    # Vertices berupa array berisi nama simpul
    # Jika simpul yang ingin di hapus tidak ada pada graph,
    # akan mengeluarkan pesan.
    for vertice in vertices:
        if (vertice in graph):
            graph.pop(vertice)
        else:
            print("Tidak ada simpul " + vertice + " pada graph")

```

```python
def get_graph(fileName) :
    # Mengembalikan graph yang terbentuk dari file fileName.
    # Graph yang dibentuk merupakan representasi adjacency list.
    # Dalam python memanfaatkan dictionary.
    # Contoh: isi file.txt:
    # C1, C3.
    # C2, C1, C4.
    # C3.
    # C4, C1, C3.
    # C5, C2, C4.
    # Maka akan mengembalikan:
    # {
    # 'C1': ['C2', 'C4'],
    # 'C2': ['C5'],
    # 'C3': ['C1', 'C4'],
    # 'C4': ['C2', 'C5'],
    # 'C5': []
    # }
```

```
    f = open("../test/" + fileName, "r")
    problem = []
    for line in f :
        line = line.replace(".", "")
        line = line.replace(" ", "")
        line = line.replace("\n", "")
        line = line.split(",")
        problem.append(line)
    f.close()

    graph = {}
    for vertices in problem:
        add_vertice(graph, vertices[0])

    for vertices in problem:
        for i in range(1, len(vertices)):
            add_edge(graph, vertices[i], vertices[0])

    return graph

def get_zero_in_degree_vertices(graph):
    # # Mengembalikan array simpul-simpul yang
    # memiliki in-degree 0
    # Contoh: graph =
    # {
    # 'C2': ['C4'],
    # 'C3': ['C4', 'C5'],
    # 'C4': ['C5'],
    # 'C5': []
    # }
    # Maka akan mengembalikan:
    # ["C2", "C3"]
    vertices_in_degree = {}
    for vertice in graph:
        vertices_in_degree[vertice] = 0

    for vertice in graph:
        for in_vertice in graph[vertice]:
            vertices_in_degree[in_vertice] += 1

    zero_in_degree_vertices = []
    for vertice in vertices_in_degree:
        if (vertices_in_degree[vertice] == 0):
            zero_in_degree_vertices.append(vertice)

    return zero_in_degree_vertices
```

```python
def topo_sort(graph):
    # Mengembalikan simpul hasil sorting graph
    # Jika graph bukan dag, maka akan mengembalikan 0
    # Contoh:
    # graph =
    # {
    # 'C1': ['C2', 'C3', 'C4', 'C5'],
    # 'C2': ['C4'],
    # 'C3': ['C4', 'C5'],
    # 'C4': ['C5'],
    # 'C5': []
    # }
    # Maka akan mengembalikan
    # [["C1"], ["C2", "C3"], ["C4"], ["C5"]]
    sorted = []
    graph_copy = dict(graph)    # Copy agar graph pada parameter tidak terhapus
    while (len(graph_copy) != 0):
        zero_in_degree_vertices = get_zero_in_degree_vertices(graph_copy)
        if (len(zero_in_degree_vertices) > 0):
            sorted.append(zero_in_degree_vertices)
            delete_vertice(graph_copy, zero_in_degree_vertices)
        else:
            return 0
    return sorted

```

```python
### Main program ###
graph = get_graph(input("Masukkan nama file: "))
solution = topo_sort(graph)
if (not(solution)) :
    print("Graph bukan sebuah DAG")
else :
    for i in range(len(solution)) :
        print("Semester " + str(i + 1) + "  : ", end="")
        for j in range(len(solution[i])) :
            if (j == 0) :
                print(solution[i][j], end="")
            else :
                print(", " + solution[i][j], end="")
        print()
```

C. Pengujian Program

Checklist pengujian:

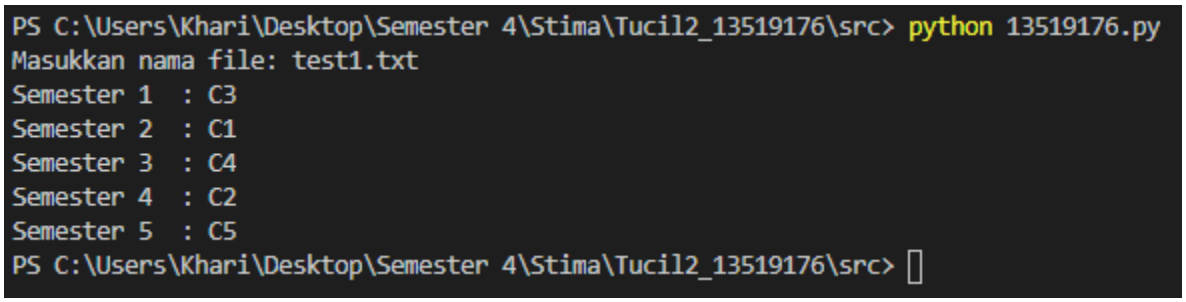
Poin	Ya	Tidak
1. Program berhasil dikompilasi.	✓	
2. Program berhasil running.	✓	
3. Program dapat menerima berkas input dan menuliskan output.	✓	
4. Luaran sudah benar untuk semua kasus input.	✓	

Screenshot pengujian:

1. test1.txt :

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

Hasil kompilasi:



2. test2.txt:

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

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test2.txt
Semester 1  : C1
Semester 2  : C2, C3
Semester 3  : C4
Semester 4  : C5
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

3. test3.txt:

```
C1.
C2, C1.
C3, C1, C2.
C4, C3.
C5, C3.
C6, C5.
C7, C6, C10.
C8, C4, C7, C12.
C9, C5.
C10, C6, C9.
C11, C3.
C12, C11.
C13, C8.
C14, C8, C13, C15.
C15, C13.
```

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test3.txt
Semester 1  : C1
Semester 2  : C2
Semester 3  : C3
Semester 4  : C4, C5, C11
Semester 5  : C6, C9, C12
Semester 6  : C10
Semester 7  : C7
Semester 8  : C8
Semester 9  : C13
Semester 10  : C15
Semester 11  : C14
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

4. test4.txt:

```
C0, C3, C7.
C1, C5, C7.
```

```
C2, C1.
C3, C7.
C4, C1, C3.
C5, C7.
C6, C0, C1.
C7.
```

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test4.txt
Semester 1  : C7
Semester 2  : C3, C5
Semester 3  : C0, C1
Semester 4  : C2, C4, C6
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

5. test5.txt:

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

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test5.txt
Semester 1  : C1
Semester 2  : C2
Semester 3  : C3
Semester 4  : C4
Semester 5  : C5
Semester 6  : C6
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

6. test6.txt:

```
C2, C11.
C3.
C5.
C7.
C8, C7, C3.
C9, C8, C11.
C10, C3, C11.
C11, C5, C7.
```

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test6.txt
Semester 1  : C3, C5, C7
Semester 2  : C8, C11
Semester 3  : C2, C9, C10
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

7. test7.txt:

```
C1.
C2, C1.
C3, C2.
C4, C2.
C5, C3, C4.
C6, C5.
C7, C5.
C8, C3, C4, C6, C7.
C9, C8, C1.
```

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test7.txt
Semester 1  : C1
Semester 2  : C2
Semester 3  : C3, C4
Semester 4  : C5
Semester 5  : C6, C7
Semester 6  : C8
Semester 7  : C9
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```

8. test8.txt:

```
C1, C2, C6, C8.
C2.
C3, C7.
C4, C1, C3, C7.
C5, C1, C6, C10.
C6, C11.
C7, C1, C11.
C8.
C9, C3.
C10.
C11.
```

Hasil kompilasi:

```
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src> python 13519176.py
Masukkan nama file: test8.txt
Semester 1  : C2, C8, C10, C11
Semester 2  : C6
Semester 3  : C1
Semester 4  : C5, C7
Semester 5  : C3
Semester 6  : C4, C9
PS C:\Users\Khari\Desktop\Semester 4\Stima\Tucil2_13519176\src>
```