Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

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

Oleh:

Nama : Nabilah Erfariani

NIM : 13519181

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

BAB 1

DESKRIPSI MASALAH

Penyusunan Rencana Kuliah dengan Topological Sort

Pada tugas kali ini, mahasiswa diminta membuat aplikasi sederhana yang dapat menyusun rencana pengambilan kuliah, dengan memanfaatkan algoritma Decrease and Conquer. Penyusunan Rencana Kuliah diimplementasikan dengan menggunakan pendekatan Topological Sorting. Berikut akan dijelaskan tugas yang dikerjakan secara detail.

1. Aplikasi akan menerima daftar mata kuliah beserta prasyarat yang harus diambil seorang mahasiswa sebelum mengambil mata kuliah tersebut. Daftar mata kuliah tersebut dituliskan dalam suatu file teks dengan format:

<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

Contoh input:

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

Gambar 2. Contoh sebuah berkas masukan Daftar Kuliah

Maka output dari input diatas adalah:

Semester I : C3

Semester II : C1

Semester III : C4

Semester IV : C2

SemesterV : C5.

BAB 2

SOLUSI

Algoritma *Topological Sort*

Topological sorting merupakan suatu urutan linear setiap simpul dimana setiap $\boldsymbol{uv}$ yang saling terhubung secara langsung (direct), simpul $\boldsymbol{u}$ muncul sebelum simpul $\boldsymbol{v}$ dalam suatu urutan. Topological sorting menerapkan algoritma traversal yang diterapkan pada metode searching DFS (depth first search) dan disajikan dalam bentuk digraph/direct graph. Untuk merepresantiskan digraph, maka digunakan *adjacency matrix* dan *adjacency list.*

Algoritma topological sort :

1. Temukan simpul dengan jumlah derajat masuk = 0
2. 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.
3. Letakkan simpul yang dikeluarkan tersebut pada awal list.
4. Ulangi langkah-langkah diatas hingga semua simpul pada DAG terpilih.

Decrease and conquer adalah sebuah teknik yang berdasar pada relasi antara solusi untuk suatu kasus dan solusi untuk bagian yang lebih kecil dari kasus tersebut. Pendekatan yang digunakan pada teknik decrease and conquer yaitu top-down-approach atau bottom-upapproach (incremental approach). Terdapat 3 bentuk variasi dari teknik *decrease-and-conquer*:

1. Decrease by constant
2. Decrease by constant factor
3. Variable size decrease

Topological sort sendiri merupakan bentuk variasi variable size decrease dari teknik decrease and conquer, yaitu ukuran kasus berkurang (size-reduction) dari operasi algoritma sebelumnya. Hal ini dapat terlihat pada langkah pengurangan simpul dari graf jika simpul memiliki derajat 0.

BAB III

IMPLEMENTASI PROGRAM DALAM C++

```
/*  Nama    : Nabilah Erfariani
    NIM     : 13519181
    Kelas   : K-04
    Tugas Kecil 2 Strategi Algoritma
    Penerapan Decrease and Conquer
*/

#include <fstream>
#include <iostream>
#include <vector>
#include <cassert>

namespace std { };
using namespace std;

void topologicalsort(vector<vector<size_t> >& dag, vector<bool>& sudahexplored, si
ze_t i, vector<size_t>& hasilsorting, size_t& jumlahderajat)
// melakukan proses topological sort
{
    sudahexplored[i] = true;

    size_t j = 0;
    while(j < dag[i].size()){
        //mencari vertex yang belum di explore
        if(sudahexplored[dag[i][j]] == false){
            topologicalsort(dag, sudahexplored, dag[i][j], hasilsorting, jumlahder
ajat);
        }
        j++;
    }

    //mengurangi jumlah derajat
    --jumlahderajat;

    //meletakkan derajat tersebut pada list
    hasilsorting[jumlahderajat] =i;

    return;
}

void exploredvertex (vector<vector<size_t> >& dag, vector<size_t>& hasilsorting)
//fungsi untuk mencari vertex yang tidak memiliki preferences
{

    vector<bool> sudahexplored(dag.size(), false);
```

```
    size_t jumlahderajat = dag.size();
    size_t i = 0;

    while(i < dag.size()){
        //mencari vertex yang belum di explore
        if(sudahexplored[i]==false){
            //melakukan topological sort
            topologicalsort(dag, sudahexplored, i, hasilsorting, jumlahderajat);
        }
        ++i;
    }

    //set jumlah derajat menjadi 0
    assert(jumlahderajat == 0);

    return;
}

istream& operator>> (istream& operatorassign, vector<vector<size_t> > & dag)
//mendefinisikan method operator
{
    // KAMUS
    size_t i = 0;
    size_t j = 0;

    //ALGORITMA
    operatorassign >> i; //membaca operator untuk i

    //melakukan resize untuk DAG
    dag.resize(i);

    //melakukan loop hingga panjang DAG
    while(operatorassign >> i >> j){

        --i;//decrement i
        --j;//decrement j

        //jika i dan j tidak sama
        if(i != j){
            dag[i].push_back(j);
        }
    }

    return operatorassign;
}

int main(int argc, char* argv[]){
    //Definisikan vektor
    vector<vector<size_t> > dag;
```

```
    //set nama file pada command line
    if(argc>1){
        ifstream myfile;

        //membuka file txt
        myfile.open(argv[1]);
        myfile >> dag;
        myfile.close();

    }

    //set jumlah simpul pada DAG / jumlah variabel pada DAG
    assert(dag.size() == 5);

    //definisikan vector
    vector<size_t> hasilsorting(dag.size(), 0);
    exploredvertex(dag, hasilsorting);

    cout << "Hasil Topological Sort: " << endl;

    //melakukan loop sebanyak panjang DAG
    for (size_t i =0; i<hasilsorting.size(); ++i){
        //menampilkan hasil topological sort
        cout << "Semester " << i+1 << " : " << (hasilsorting[i]+1) << "\n";
    }
    cout << endl;
    return 0;
}
```

BAB IV

TESTING PROGRAM

Input	Output
5 1 3 2 1 4 3 4 1 3 5 2 4	C:\Users\ASUS\TUCILSTIMA>topologicalsort input1.txt Hasil Topological Sort: Semester 1 : 2 Semester 2 : 4 Semester 3 : 1 Semester 4 : 3 Semester 5 : 5
5 1 2 5 3 4 5 4 5 5 2	C:\Users\ASUS\TUCILSTIMA>topologicalsort input2.txt Hasil Topological Sort: Semester 1 : 4 Semester 2 : 5 Semester 3 : 3 Semester 4 : 1 Semester 5 : 2
6 1 2 3 4 2 4 3 5 6 4 6 5 6 2 6	C:\Users\ASUS\TUCILSTIMA>topologicalsort input3.txt Hasil Topological Sort: Semester 1 : 6 Semester 2 : 3 Semester 3 : 5 Semester 4 : 1 Semester 5 : 2 Semester 6 : 4
4 1 2 2 3 3 4 4 1	C:\Users\ASUS\TUCILSTIMA>topologicalsort input4.txt Hasil Topological Sort: Semester 1 : 1 Semester 2 : 2 Semester 3 : 3 Semester 4 : 4
3 1 3 2 1 2 3	C:\Users\ASUS\TUCILSTIMA>topologicalsort input5.txt Hasil Topological Sort: Semester 1 : 2 Semester 2 : 1 Semester 3 : 3

9 8 9 1 4 1 1 5 2 2 4 3 3 5 4 3 8 5 4 6 6 4 7 7 4 8 8 5 7 9	C:\Users\ASUS\TUCILSTIMA>topologicalsort input6.txt Hasil Topological Sort: Semester 1 : 6 Semester 2 : 5 Semester 3 : 3 Semester 4 : 8 Semester 5 : 9 Semester 6 : 2 Semester 7 : 1 Semester 8 : 4 Semester 9 : 7
6 1 2 2 3 2 4 4 5 5 6	C:\Users\ASUS\TUCILSTIMA>topologicalsort input7.txt Hasil Topological Sort: Semester 1 : 1 Semester 2 : 2 Semester 3 : 4 Semester 4 : 5 Semester 5 : 6 Semester 6 : 3

```
50
1 2
1 4
2 3
2 5
3 6
4 7
5 8
6 8
7 8
8 9
9 10
10 11
11 12
11 14
12 13
12 15
13 16
14 17
15 18
16 18
17 18
18 19
19 20
20 21
21 22
21 24
22 23
22 25
23 26
24 27
25 28
26 28
27 28
28 29
29 30
30 31
31 32
31 34
32 33
32 35
33 36
34 37
35 38
36 38
37 38
38 39
39 40
40 41
```

```
C:\Users\ASUS\TUCILSTIMA>topologicalsort input8.txt
Hasil Topological Sort:
Semester 1 : 1
Semester 2 : 4
Semester 3 : 7
Semester 4 : 2
Semester 5 : 5
Semester 6 : 3
Semester 7 : 6
Semester 8 : 8
Semester 9 : 9
Semester 10 : 10
Semester 11 : 11
Semester 12 : 14
Semester 13 : 17
Semester 14 : 12
Semester 15 : 15
Semester 16 : 13
Semester 17 : 16
Semester 18 : 18
Semester 19 : 19
Semester 20 : 20
Semester 21 : 21
Semester 22 : 24
Semester 23 : 27
Semester 24 : 22
Semester 25 : 25
Semester 26 : 23
Semester 27 : 26
Semester 28 : 28
Semester 29 : 29
Semester 30 : 30
Semester 31 : 31
Semester 32 : 34
Semester 33 : 37
Semester 34 : 32
Semester 35 : 35
Semester 36 : 33
Semester 37 : 36
Semester 38 : 38
Semester 39 : 39
Semester 40 : 40
Semester 41 : 41
```

```
Semester 41 : 41
Semester 42 : 44
Semester 43 : 47
Semester 44 : 42
Semester 45 : 45
Semester 46 : 43
Semester 47 : 46
Semester 48 : 48
Semester 49 : 49
Semester 50 : 50
```

41 42 41 44 42 43 42 45 43 46 44 47 45 48 46 48 47 48 48 49 49 50	

Link source code:

https://github.com/nabilaherfar/Tucil-2-Stima

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.	✓	