

TUGAS KECIL 2 IF 2211 STRATEGI ALGORITMA
PENYUSUNAN RENCANA KULIAH DENGAN *TOPOLOGICAL*
***SORT* (PENERAPAN *DECREASE AND CONQUER*)**

Disusun untuk memenuhi tugas Strategi Algoritma

DISUSUN OLEH:

NAUFAL ALEXANDER SURYASUMIRAT 13519135



INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

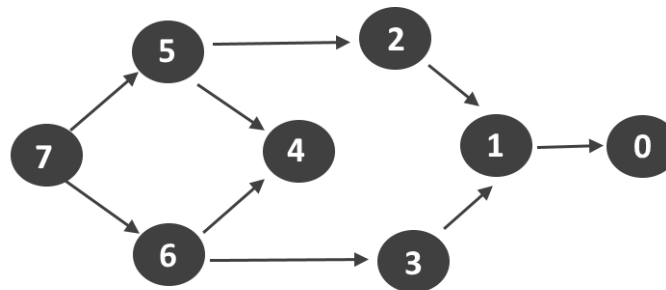
CEK LIST

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

Algoritma *Topological Sort* dan Kaitannya dengan Pendekatan *Decrease and Conquer*

I. Topological Sort

Topological Sort merupakan proses penyortiran atau pengurutan simpul-simpul (*vertices*) pada graf berarah yang tidak memiliki siklus pada sub-graf atau graf tersebut sendiri atau juga disebut sebagai *acyclic graph*, dengan kata lain merupakan *Directed Acyclic Graph* (DAG). Secara umum, *topological sort* mengurutkan pengambilan simpul pada DAG sehingga *pre-requisite* atau simpul-simpul/ketergantungannya yang didahuluinya telah diambil/terurut terlebih dahulu sebelum berlanjut ke simpul-simpul lainnya. Jika terdapat sebuah siklus dalam sebuah graf maka persoalan tersebut tidak dapat diselesaikan dengan *Topological Sort*. Contoh dari penggunaan algoritma tersebut sangat banyak, diantaranya adalah penjadwalan urutan pekerjaan berdasarkan ketergantungannya sampai dengan penjadwalan instruksi pada ilmu komputer (*Computer Science*).



Topological Sort : 7 6 5 4 3 2 1 0

Gambar 1.1 Topological Sort pada DAG

Terlihat pada Gambar 1.1 adalah contoh persoalan keterurutan Graf yang dapat diselesaikan menggunakan algoritma *Topological Sort* karena dalam graf tersebut tidak terdapat sebuah siklus. Yang pertama dilakukan pada *Topological Sort* adalah untuk mengambil sebuah simpul yang tidak memiliki ketergantungan dan menghilangkan ketergantungan simpul tersebut yang dimiliki oleh simpul-simpul lainnya. Pada contoh Gambar 1.1, setelah simpul dengan angka 7 diambil, simpul dengan angka 5 dan 6 tidak lagi memiliki ketergantungan dan dapat diambil pada iterasi selanjutnya yang menghasilkan salah satu solusi yang dapat dilihat pada gambar tersebut.

II. *Decrease and Conquer*

Algoritma *Decrease and Conquer* memiliki keterkaitan atau kesamaan dengan algoritma *Divide and Conquer*, namun jika pada *Divide and Conquer* permasalahan dibagi menjadi bagian-bagian yang lebih kecil untuk tiap bagian tersebut diselesaikan dan digabungkan (*Divide-Conquer-Combine*), pada algoritma *Decrease and Conquer*, suatu permasalahan dibagi menjadi beberapa bagian permasalahan yang lebih kecil dan dipilih salah satu bagian permasalahan tersebut dan diselesaikan, kemudian hasil tersebut disambungkan dengan solusi dari instansi permasalahan kecil lainnya sehingga didapatkan solusi untuk permasalahan tersebut.

Terdapat tiga varian pada algoritma yang menerapkan pendekatan *decrease and conquer*, yaitu *decrease by a constant*, *decrease by a constant factor*, dan *decrease by a variable size*. Untuk yang pertama, ukuran instansi permasalahan akan direduksi sebesar konstanta yang sama pada tiap iterasi algoritma, kemudian untuk yang kedua, ukuran instansi permasalahan akan direduksi sebesar faktor konstanta yang besarnya sama pada tiap iterasi algoritma, dan untuk yang terakhir, ukuran instansi permasalahan direduksi dengan ukuran yang bervariasi pada tiap iterasi algoritma.

Algoritma yang merupakan implementasi atau menggunakan pendekatan *Decrease and Conquer* beberapa diantaranya adalah *Depth-First-Search*, *Breadth-First-Search*, *Insertion Sort*, *Binary search*, dan algoritma yang diimplementasikan pada tugas kecil ini, yaitu *Topological Sort*.

III. Kaitan *Topological Sort* dengan *Decrease and Conquer*

Seperti yang telah dijelaskan pada sebelumnya dan telah disinggung juga bahwa algoritma *Topological Sort* merupakan algoritma yang menggunakan pendekatan *Decrease and Conquer* untuk menyelesaikan permasalahannya. Lebih rincinya lagi, *Topological Sort* menggunakan pendekatan *decrease and conquer* varian *decrease by a constant*. Untuk diperjelas, pada tiap iterasi pengambilan simpul pada graf permasalahan, diambil simpul pada graf yang tidak memiliki *in-degree* (tidak terdapat penghubung/*edge* yang mengarah kepada simpul tersebut) dan akibatnya adalah permasalahan menjadi upa-permasalahan yang lebih kecil yang dikurangi/direduksi sebesar jumlah simpul pada graf yang memiliki *in-degree* berjumlah 0. Kemudian tiap simpul yang diambil pada tiap iterasi dijadikan sebuah solusi yang merupakan urutan dari pengambilan simpul pada graf berarah tanpa siklus tersebut.

Source Code Program dalam Bahasa C++

Topological Sort yang kali ini diimplementasikan untuk tugas kecil mata kuliah Strategi Algoritma bertujuan untuk mengurutkan urutan pengambilan mata kuliah dan dalam test case kali ini menggunakan mata kuliah pada berbagai jurusan pada ITB, beserta juga dengan satu contoh jika program menerima sebuah *cyclic graph* dalam persoalan. *Pre-requisite* dalam *file* persoalan kali ini tidak 100% mencerminkan kenyataannya untuk mengambil mata kuliah di ITB, tetapi secara garis besar dibuat agar memiliki kesamaan dengan pengambilan mata kuliah pada kurikulum yang disediakan pada website SIX ITB.

Secara umum, *Topological Sort* yang diimplementasikan pada tugas kecil ini menggunakan kelas *Vertex*, *Graph*, dan *fileHandler* yang masing-masing memiliki tugas/rincian sebagai berikut:

1. Vertex

Kelas *Vertex* merupakan kelas simpul yang memiliki atribut sebagai berikut:

- *name* (string yang menyimpan nama/kode yang membedakan tiap simpul yang ada),
- *inDegree* (integer yang menyimpan banyaknya *edge* yang mengarah ke simpul tersebut),
- *outVectors* (vector yang berisi pointer ke simpul-simpul yang menyimpan simpul tujuan dari simpul tersebut / *edge* yang mengarah keluar dari simpul tersebut).
- Terdapat method-method yang tersedia dalam kelas *Vertex* (konstruktor dan lainnya) untuk melengkapi atribut yang telah disebutkan ketika membuat suatu instansi *Vertex*.

2. Graph

Kelas *Graph* merupakan kelas graf berarah yang menyimpan instansi *vertex-vertex* yang telah dibuat dan juga memiliki *method topoSort* untuk mengurutkan *vertex-vertex* tersebut dengan algoritma *Topological Sort*. Atribut yang dimilikinya adalah antara lain:

- *verticesCount* (integer yang menyimpan jumlah simpul yang dimiliki graf tersebut),
- *VList* (vector yang menyimpan pointer ke *vertex* yang terdapat dalam graf tersebut),
- *sortedVertex* (vector dari vector yang menyimpan pointer ke *vertex* hasil pengurutan dengan algoritma *Topological Sort*).
- Terdapat juga method-method lainnya untuk melengkapi atribut yang telah disebutkan.

3. fileHandler

Kelas yang bertujuan untuk membaca *file input* berbentuk txt dan menyimpan hasil pembacaan tersebut sebagai *vertex-vertex* yang nantinya akan diurutkan dengan algoritma *Topological Sort*.

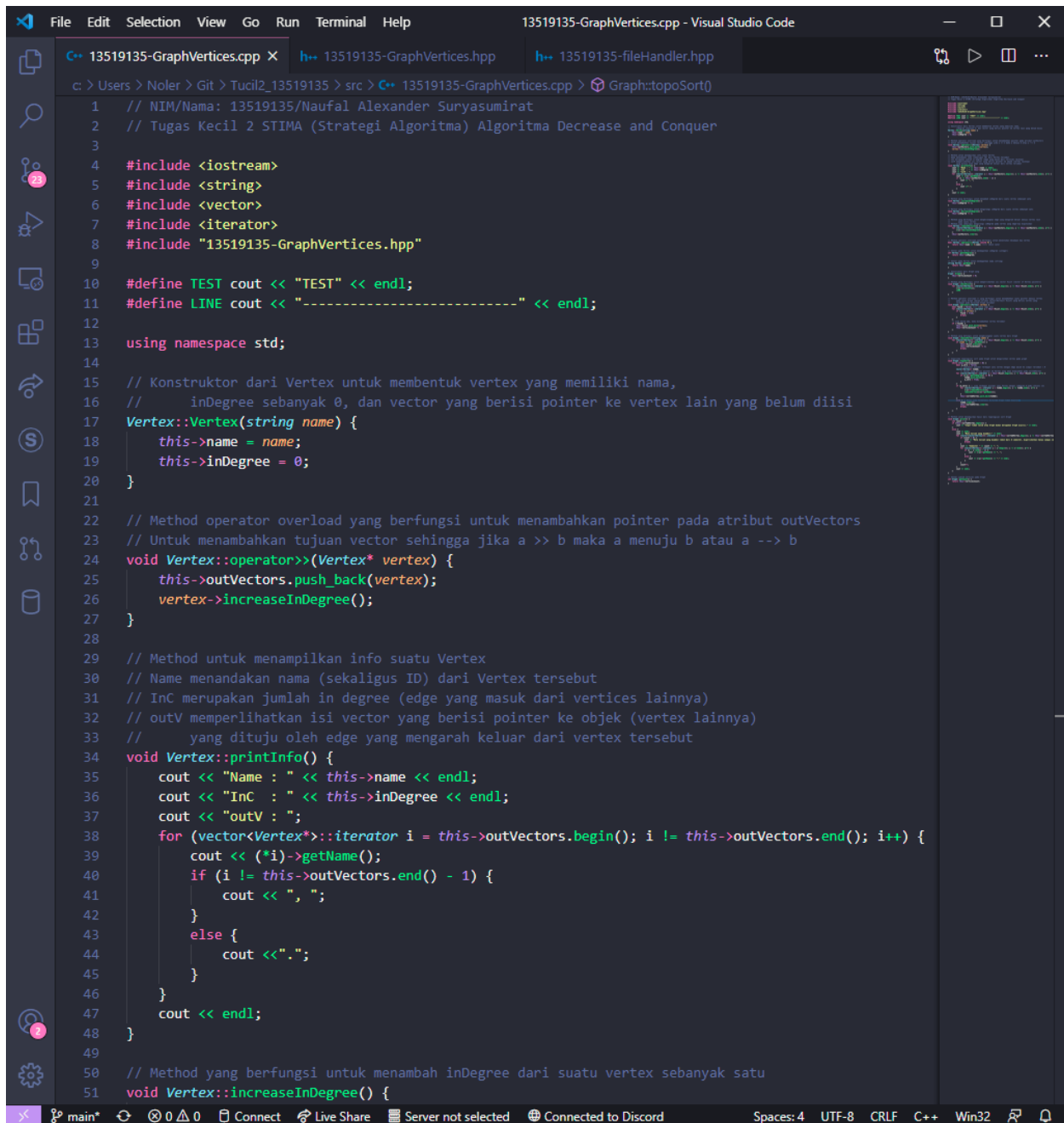
Algoritma *Topological Sort* pada tugas kecil ini diimplementasikan dalam kelas *Graph* dan dinamakan sebagai method *topoSort*. Berikut adalah screenshot hanya algoritma *Topological Sort* yang diimplementasikan.

```
// Algoritma topological sort pada Graph untuk mengurutkan vertex pada graph
void Graph::topoSort() {
    while (this->verticesCount > 0) {
        bool a_zero = false;
        // indikator minimal terdapat satu vertex dengan edge masuk ke simpul tersebut = 0
        vector<Vertex*> toAdd;
        // vector yang berisi pointer ke Vertex yang akan dimasukkan pada sortedVertex
        for (vector<Vertex*>::iterator i = this->VList.begin(); i != this->VList.end(); i++) {
            if ((*i)->getInDegree() == 0) {
                toAdd.push_back(*i);
                a_zero = true;
            }
        }
        if (a_zero) { // Jika terdapat minimal satu Vertex dengan inDegree 0 pada iterasi ini
            for (vector<Vertex*>::iterator i = toAdd.begin(); i != toAdd.end(); i++) {
                (*i)->removeVertices();
                removeVertex((*i)->getName());
            }
            this->sortedVertex.push_back(toAdd);
        }
        else { // Kasus jika terdeteksi Directed Graph tidak Acyclic
            toAdd.clear();
            this->sortedVertex.clear();
            break;
        }
    }
}
```

Gambar 2.1 Algoritma Implementasi Topological Sort

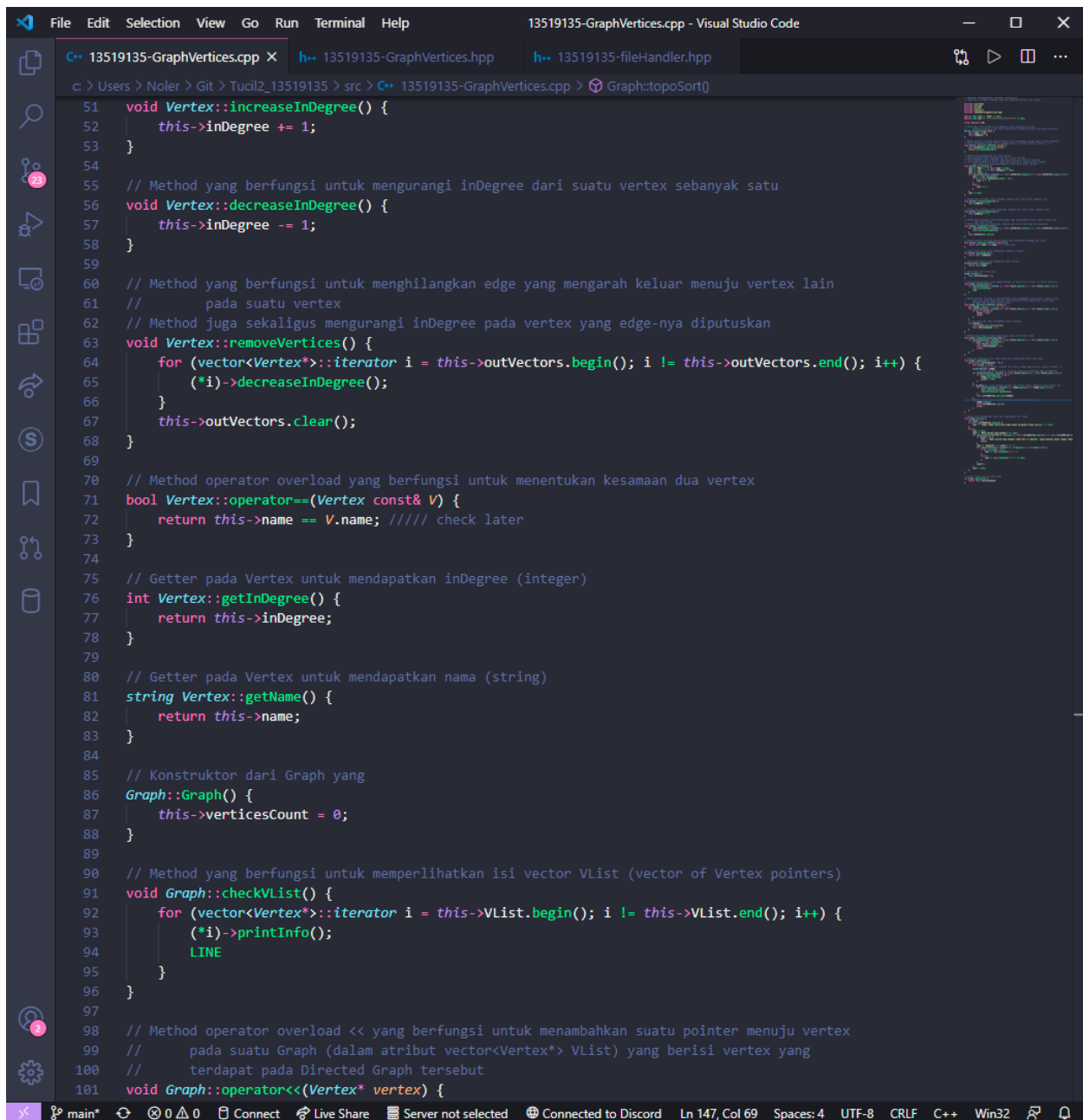
Seperti yang dapat dilihat pada Gambar 2.1 adalah algoritma *topological sort* yang diimplementasikan pada program sederhana ini. Secara umum, setelah dibuat instansi-instansi dari kelas *Vertex* beserta dengan tiap *edge* yang dimiliki tiap vertexnya, simpul-simpul tersebut kemudian dimasukkan ke dalam suatu instansi dari kelas *Graph*. Kemudian digunakan metode *topoSort* yang bekerja dengan mencari simpul yang memiliki *in-degree* berjumlah 0, kemudian menambahkan simpul tersebut (dimasukkan ke dalam vector terlebih dahulu) ke dalam atribut *sortedVertex* pada kelas *Graph*. Setelah ditambahkan, tiap *edge* yang mengarah keluar dari simpul tersebut dihilangkan (metode *removeVertices* dari kelas *Vertex*), akibatnya *in-degree* dari simpul-simpul yang terhubung berkurang dan simpul yang ditambahkan ke *sortedVertex* tersebut pun dihilangkan (metode *removeVertex* dari kelas *Graph*). setelah iterasi tersebut dari permasalahan dan didapat permasalahan dalam bentuk yang lebih kecil untuk diselesaikan. Proses tersebut pun dijalankan sampai ditemukan sebuah siklus (jika ditemukan maka tidak ada solusi dari permasalahan tersebut) atau hingga seluruh simpul telah ditambahkan sebagai simpul yang telah diurutkan. Terakhir, solusi tersebut diperlihatkan dengan menggunakan metode *print* yang dimiliki oleh kelas *Graph*.

Screenshot Source Code (GraphVertices.cpp dan main.cpp)



```
1 // NIM>Nama: 13519135/Naufal Alexander Suryasumirat
2 // Tugas Kecil 2 STIMA (Strategi Algoritma) Algoritma Decrease and Conquer
3
4 #include <iostream>
5 #include <string>
6 #include <vector>
7 #include <iterator>
8 #include "13519135-GraphVertices.hpp"
9
10 #define TEST cout << "TEST" << endl;
11 #define LINE cout << "-----" << endl;
12
13 using namespace std;
14
15 // Konstruktor dari Vertex untuk membentuk vertex yang memiliki nama,
16 // inDegree sebanyak 0, dan vector yang berisi pointer ke vertex lain yang belum diisi
17 Vertex::Vertex(string name) {
18     this->name = name;
19     this->inDegree = 0;
20 }
21
22 // Method operator overload yang berfungsi untuk menambahkan pointer pada atribut outVectors
23 // Untuk menambahkan tujuan vector sehingga jika a >> b maka a menuju b atau a --> b
24 void Vertex::operator>>(Vertex* vertex) {
25     this->outVectors.push_back(vertex);
26     vertex->increaseInDegree();
27 }
28
29 // Method untuk menampilkan info suatu Vertex
30 // Name menandakan nama (sekali ID) dari Vertex tersebut
31 // InC merupakan jumlah in degree (edge yang masuk dari vertices lainnya)
32 // outV memperlihatkan isi vector yang berisi pointer ke objek (vertex lainnya)
33 // yang dituju oleh edge yang mengarah keluar dari vertex tersebut
34 void Vertex::printInfo() {
35     cout << "Name : " << this->name << endl;
36     cout << "InC : " << this->inDegree << endl;
37     cout << "outV : ";
38     for (vector<Vertex*>::iterator i = this->outVectors.begin(); i != this->outVectors.end(); i++) {
39         cout << (*i)->getName();
40         if (i != this->outVectors.end() - 1) {
41             cout << ", ";
42         }
43         else {
44             cout << ".";
45         }
46     }
47     cout << endl;
48 }
49
50 // Method yang berfungsi untuk menambah inDegree dari suatu vertex sebanyak satu
51 void Vertex::increaseInDegree() {
```

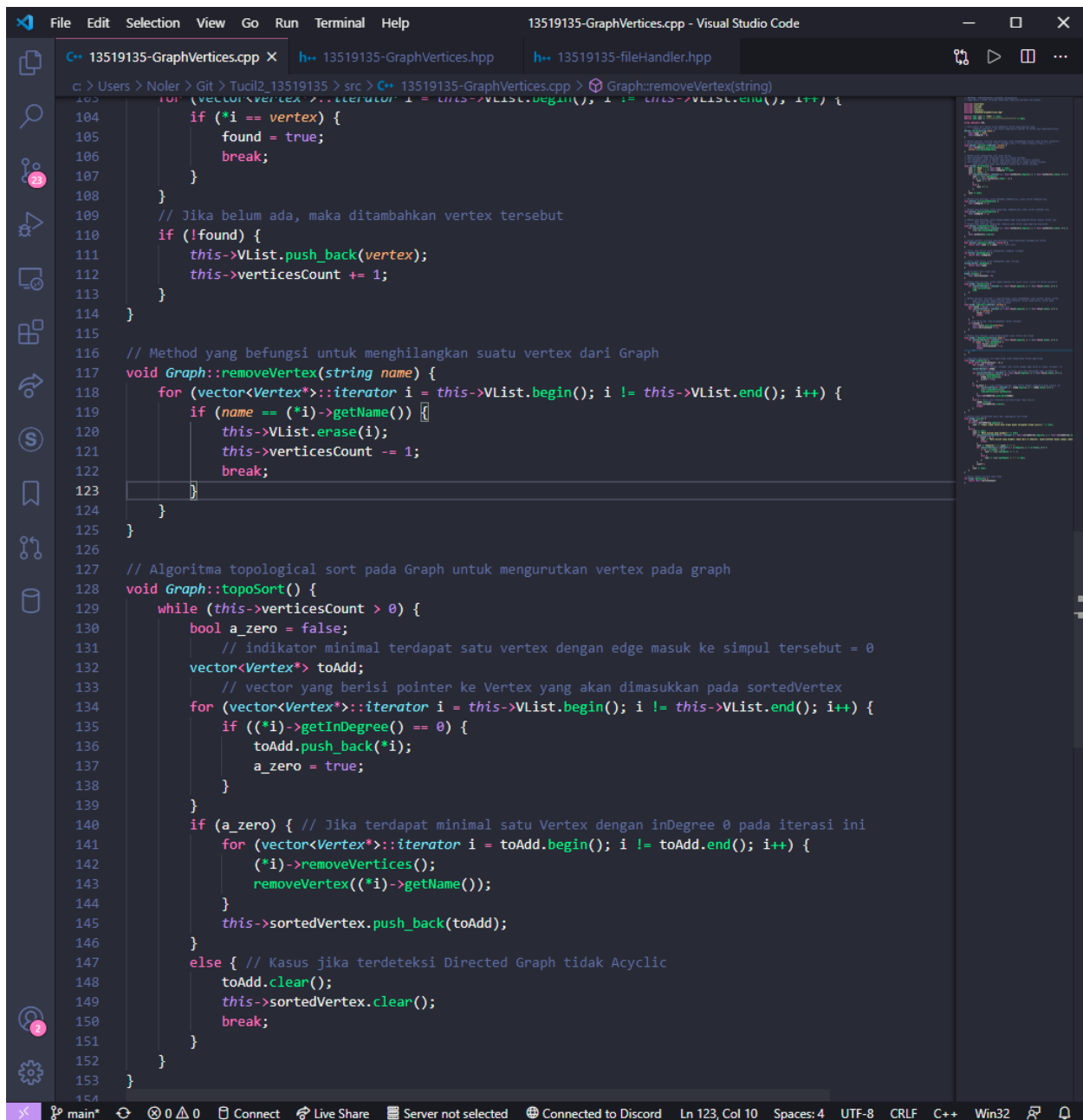
Gambar 2.2 GraphVertices.cpp - 1



```
11413519135-GraphVertices.cpp X 113519135-GraphVertices.hpp 113519135-fileHandler.hpp
c:\Users> Noler> Git> Tuci2_13519135> src> C++ 13519135-GraphVertices.cpp> Graph::topoSort()

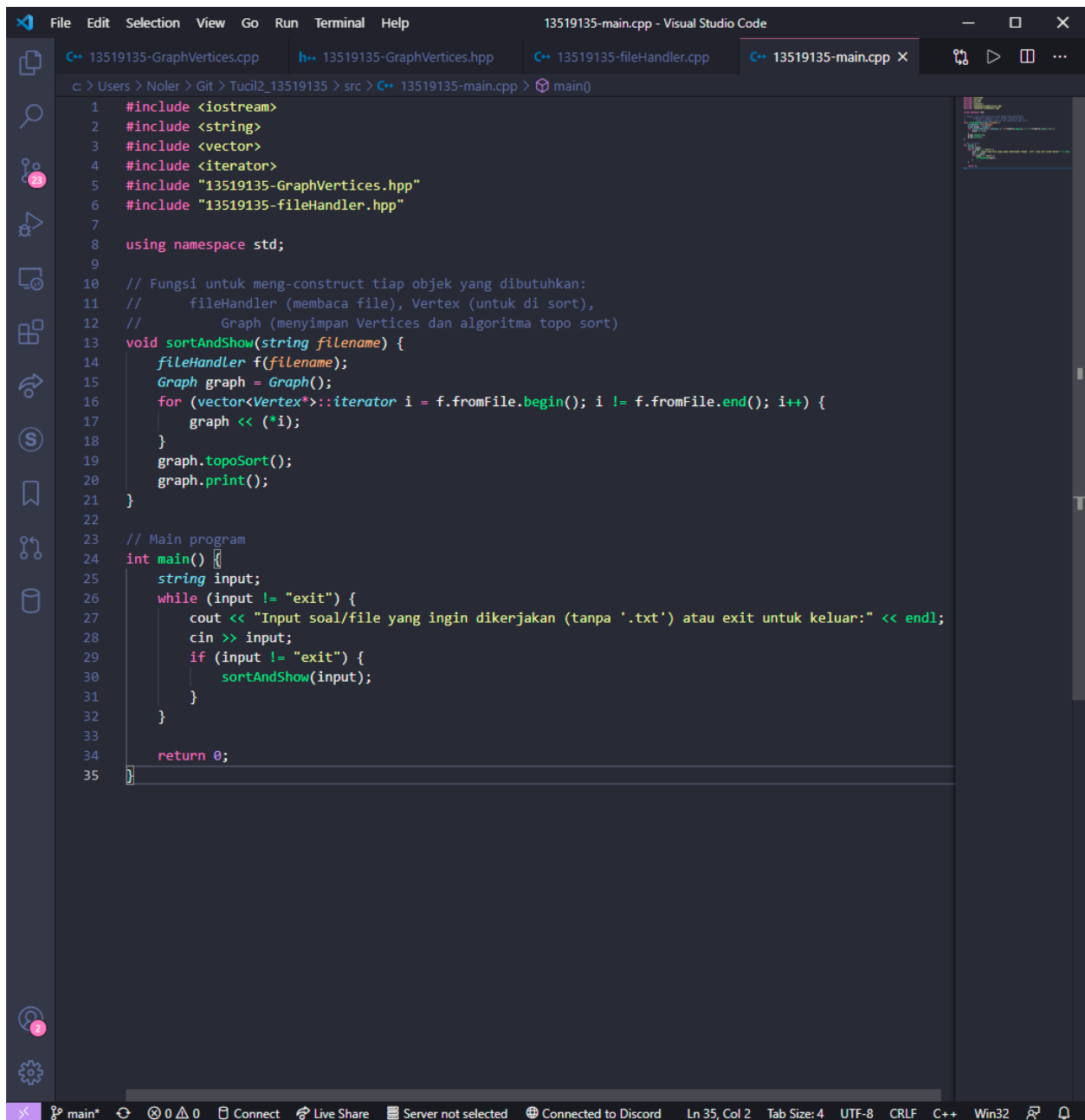
51 void Vertex::increaseInDegree() {
52     this->inDegree += 1;
53 }
54
55 // Method yang berfungsi untuk mengurangi inDegree dari suatu vertex sebanyak satu
56 void Vertex::decreaseInDegree() {
57     this->inDegree -= 1;
58 }
59
60 // Method yang berfungsi untuk menghilangkan edge yang mengarah keluar menuju vertex lain
61 // pada suatu vertex
62 // Method juga sekaligus mengurangi inDegree pada vertex yang edge-nya diputuskan
63 void Vertex::removeVertices() {
64     for (vector<Vertex*>::iterator i = this->outVectors.begin(); i != this->outVectors.end(); i++) {
65         (*i)->decreaseInDegree();
66     }
67     this->outVectors.clear();
68 }
69
70 // Method operator overload yang berfungsi untuk menentukan kesamaan dua vertex
71 bool Vertex::operator==(Vertex const& V) {
72     return this->name == V.name; //// check later
73 }
74
75 // Getter pada Vertex untuk mendapatkan inDegree (integer)
76 int Vertex::getInDegree() {
77     return this->inDegree;
78 }
79
80 // Getter pada Vertex untuk mendapatkan nama (string)
81 string Vertex::getName() {
82     return this->name;
83 }
84
85 // Konstruktor dari Graph yang
86 Graph::Graph() {
87     this->verticesCount = 0;
88 }
89
90 // Method yang berfungsi untuk memperlihatkan isi vector VList (vector of Vertex pointers)
91 void Graph::checkVList() {
92     for (vector<Vertex*>::iterator i = this->VList.begin(); i != this->VList.end(); i++) {
93         (*i)->printInfo();
94         LINE
95     }
96 }
97
98 // Method operator overload << yang berfungsi untuk menambahkan suatu pointer menuju vertex
99 // pada suatu Graph (dalam atribut vector<Vertex*> VList) yang berisi vertex yang
100 // terdapat pada Directed Graph tersebut
101 void Graph::operator<<(Vertex* vertex) {
```

Gambar 2.3 GraphVertices.cpp - 2



```
103 for (vector<Vertex*>::iterator i = this->VList.begin(); i != this->VList.end(); i++) {
104     if (*i == vertex) {
105         found = true;
106         break;
107     }
108 }
109 // Jika belum ada, maka ditambahkan vertex tersebut
110 if (!found) {
111     this->VList.push_back(vertex);
112     this->verticesCount += 1;
113 }
114 }
115
116 // Method yang berfungsi untuk menghilangkan suatu vertex dari Graph
117 void Graph::removeVertex(string name) {
118     for (vector<Vertex*>::iterator i = this->VList.begin(); i != this->VList.end(); i++) {
119         if (name == (*i)->getName()) {
120             this->VList.erase(i);
121             this->verticesCount -= 1;
122             break;
123         }
124     }
125 }
126
127 // Algoritma topological sort pada Graph untuk mengurutkan vertex pada graph
128 void Graph::topoSort() {
129     while (this->verticesCount > 0) {
130         bool a_zero = false;
131         // indikator minimal terdapat satu vertex dengan edge masuk ke simpul tersebut = 0
132         vector<Vertex*> toAdd;
133         // vector yang berisi pointer ke Vertex yang akan dimasukkan pada sortedVertex
134         for (vector<Vertex*>::iterator i = this->VList.begin(); i != this->VList.end(); i++) {
135             if ((*i)->getInDegree() == 0) {
136                 toAdd.push_back(*i);
137                 a_zero = true;
138             }
139         }
140         if (a_zero) { // Jika terdapat minimal satu Vertex dengan inDegree 0 pada iterasi ini
141             for (vector<Vertex*>::iterator i = toAdd.begin(); i != toAdd.end(); i++) {
142                 (*i)->removeVertices();
143                 removeVertex((*i)->getName());
144             }
145             this->sortedVertex.push_back(toAdd);
146         }
147         else { // Kasus jika terdeteksi Directed Graph tidak Acyclic
148             toAdd.clear();
149             this->sortedVertex.clear();
150             break;
151         }
152     }
153 }
```

Gambar 2.4 GraphVertices.cpp - 3



```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <iterator>
5 #include "13519135-GraphVertices.hpp"
6 #include "13519135-fileHandler.hpp"
7
8 using namespace std;
9
10 // Fungsi untuk meng-construct tiap objek yang dibutuhkan:
11 //   fileHandler (membaca file), Vertex (untuk di sort),
12 //   Graph (menyimpan Vertices dan algoritma topo sort)
13 void sortAndShow(string filename) {
14     fileHandler f(filename);
15     Graph graph = Graph();
16     for (vector<Vertex*>::iterator i = f.fromFile.begin(); i != f.fromFile.end(); i++) {
17         graph << (*i);
18     }
19     graph.topoSort();
20     graph.print();
21 }
22
23 // Main program
24 int main() {
25     string input;
26     while (input != "exit") {
27         cout << "Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:" << endl;
28         cin >> input;
29         if (input != "exit") {
30             sortAndShow(input);
31         }
32     }
33
34     return 0;
35 }
```

Gambar 2.5 main.cpp

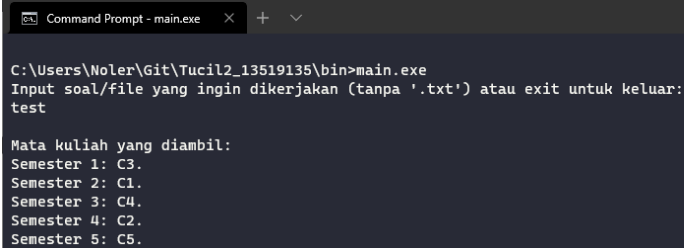
Tangkapan Layar dari Input dan Output

1. test.txt

Input:

```
1 C1, C3.
2 C2, C1, C4.
3 C3.
4 C4, C1, C3.
5 C5, C2, C4.
```

Output:



```
Command Prompt - main.exe
C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
test

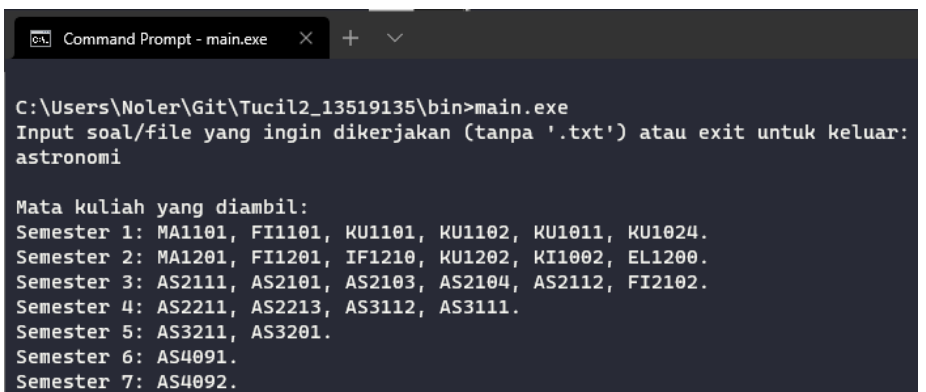
Mata kuliah yang diambil:
Semester 1: C3.
Semester 2: C1.
Semester 3: C4.
Semester 4: C2.
Semester 5: C5.
```

2. astronomi.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1101.
4 KU1102.
5 KU1011.
6 KU1024.
7
8 MA1201, MA1101.
9 FI1201, FI1101.
10 IF1210, KU1102.
11 KU1202, KU1102, KU1011.
12 KI1002, MA1101, FI1101.
13 EL1200, MA1101.
14
15 AS2111, FI1201, MA1201.
16 AS2101, MA1201.
17 AS2103, MA1201.
18 AS2104, MA1201, MA1101.
19 AS2112, KU1102, MA1201.
20 FI2102, FI1201.
21
22 AS2211, AS2111, MA1201.
23 AS2213, AS2111.
24
25 AS3112, FI1201, FI2102.
26 AS3111, FI2102, AS2111.
27
28 AS3211, AS2211, AS3112.
29 AS3201, MA1101, MA1201, FI2102, AS3112.
30
31 AS4091, AS3211.
32
33 AS4092, AS4091.
```

Output:



```
Command Prompt - main.exe
C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
astronomi

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1101, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
Semester 3: AS2111, AS2101, AS2103, AS2104, AS2112, FI2102.
Semester 4: AS2211, AS2213, AS3112, AS3111.
Semester 5: AS3211, AS3201.
Semester 6: AS4091.
Semester 7: AS4092.
```

3. biomedik.txt

Input:

```
1 EB2102.
2 EB2103.
3 EB2101.
4 MA2072.
5 KI2162.
6 EL2142.
7 EL2101.
8
9 EB2206, EB2102, EB2103.
10 EB2205, EB2101.
11 EB2207, EB2101.
12 EB2200, KI2162, EL2101.
13 MA2074, MA2072, KI2162.
14 EL2008, EL2142, EL2101.
15 EL2208, EL2101, EB2101.
```

Output:

```
Command Prompt - main.exe
C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
biomedik

Mata kuliah yang diambil:
Semester 1: EB2102, EB2103, EB2101, MA2072, KI2162, EL2142, EL2101.
Semester 2: EB2206, EB2205, EB2207, EB2200, MA2074, EL2008, EL2208.
```

4. elektro.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1101.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210, KU1102.
10 KU1202, KU1102, KU1011.
11 KI1002, MA1101, FI1101.
12 EL1200, MA1101.
13 EL2001, MA1201, EL1200.
14 EL2101, EL1200, FI1201.
15 EL2002, EL1200, FI1201.
16 EL2102, EL1200, FI1201.
17 EL2003, EL1200, FI1201.
18 EL2004, EL1200, FI1201.
19 MA2072, EL1200, FI1201.
20 EL2005, EL2001.
21 EL2205, EL2101.
22 EL2006, EL2002.
23 EL2007, EL2102.
24 EL2008, EL2003.
25 EL2208, EL2004.
26 MA2074, MA2072.
27 EL3009, EL2005.
28 EL3109, EL2205.
29 EL3010, EL2006.
30 EL3110, EL2007.
31 EL3011, EL2008.
32 EL3111, EL2208.
33 EL3012, EL2005.
34 EL3013, EL2208.
35 EL3014, EL3009.
36 EL3214, EL3109.
37 EL3015, EL3010.
38 EL3215, EL3110.
39 EL3016, EL3011.
40 EL3216, EL3111.
41 EL3017, EL3012.
42 EL3217, EL3013.
43 EL4018, EL3013.
44 EL4092, EL3014.
45 KU206X, EL3214.
46 EL4090, EL3015.
47 XXMANJ, EL3215.
48 XX4000, XXMANJ.
49 XXLING, XXMANJ.
50 EL4091, EL4090.
51 KU2071, KU206X.
```

Output:

```
Command Prompt - main.exe
C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
elektro

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1101, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
Semester 3: EL2001, EL2101, EL2002, EL2102, EL2003, EL2004, MA2072.
Semester 4: EL2005, EL2205, EL2006, EL2007, EL2008, EL2208, MA2074.
Semester 5: EL3009, EL3109, EL3010, EL3110, EL3011, EL3111, EL3012, EL3013.
Semester 6: EL3014, EL3214, EL3015, EL3215, EL3016, EL3216, EL3017, EL3217, EL4018.
Semester 7: EL4092, KU206X, EL4090, XXMANJ.
Semester 8: XX4000, XXLING, EL4091, KU2071.
```

5. informatika.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1001.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210, KU1102.
10 KU1202, KU1102.
11 KI1002, KU1011.
12 EL1200, FI1101.
13 IF2121, IF1210, MA1201.
14 IF2110, KU1102, IF1210.
15 IF2120, MA1201.
16 IF2124, EL1200.
17 IF2123, MA1201.
18 IF2130, KU1202.
19 IF2210, IF2110.
20 IF2211, IF2110.
21 IF2220, MA1201, IF2120.
22 IF2230, IF2130.
23 IF2240, IF2121, IF2120.
24 IF2250, IF2110.
25 IF3170, IF2121, IF2211.
26 IF3110, IF2210, IF2110.
27 IF3130, IF2230.
28 IF3141, IF2240, IF2250.
29 IF3150, IF2250.
30 IF3140, IF2240.
31 IF3151, IF2250.
32 IF3210, IF2110, IF3110.
33 IF3270, IF2210, IF3170.
34 IF3230, IF3130.
35 IF3250, IF2250, IF3150.
36 IF3260, IF2123, IF3151.
37 IF3280, IF3151, IF3150.
38 IF4090, IF3280.
39 IF4091, IF3280.
40 KU2071, IF3280.
41 IF4092, IF4091.
42 XXLING, IF4090.
43 KU206X, IF4091.
```

Output:

```
Command Prompt - main.exe

C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
informatika

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1001, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
Semester 3: IF2121, IF2110, IF2120, IF2124, IF2123, IF2130.
Semester 4: IF2210, IF2211, IF2220, IF2230, IF2240, IF2250.
Semester 5: IF3170, IF3110, IF3130, IF3141, IF3150, IF3140, IF3151.
Semester 6: IF3210, IF3270, IF3230, IF3250, IF3260, IF3280.
Semester 7: IF4090, IF4091, KU2071.
Semester 8: IF4092, XXLING, KU206X.
```

6. matematika.txt

Input:

```
1 MA2111.
2 KU206X.
3 MA2121.
4 MA2151.
5 MA2181.
6 KU2071, KU206X.
7 MA2231, MA2151, MA2121.
8 MA2271, MA2151, MA2181.
9 MA2251, MA2181, MA2121.
10 MA3131, MA2231.
11 XXLING, KU2071.
12 MA3171, MA2271.
13 MA3181, MA2251.
14 MA3231, MA3171.
15 XXMANJ, XXLING.
16 MA3011, MA3181.
17 MA3271, MA3171.
```

Output:

```
Command Prompt - main.exe

C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
matematika

Mata kuliah yang diambil:
Semester 1: MA2111, KU206X, MA2121, MA2151, MA2181.
Semester 2: KU2071, MA2231, MA2271, MA2251.
Semester 3: MA3131, XXLING, MA3171, MA3181.
Semester 4: MA3231, XXMANJ, MA3011, MA3271.
```

7. power.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1101.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210, KU1102.
10 KU1202, KU1102, KU1011.
11 KI1002, MA1101, FI1101.
12 EL1200, MA1101.
13 EP2091, MA1101, MA1201.
14 MA2072, MA1101, MA1201.
15 EL2001, EL1200.
16 EL2101, EL1200.
17 EL2142, EL1200.
18 MS2060, MA1101, MA1201, EL1200.
19 EP2076, EL2001.
20 EP2094, MA2072, EL2001.
21 KU2071, EL2001, EL2101.
22 MA2074, MA1101, MA 1201, MS2060.
23 EL2005, EL2001, EL2101.
24 EL2205, EL2101, EL2142.
25 EL2006, MA2072, FI1201, EL2142.
26 EP3071, EL2001, EL2006.
27 EP3073, MA2072, MA2074.
28 EP3075, EL2001, MA2074.
29 EP3095, EL2006, EL2205.
30 EP3171, EL2005, EL2205, EL2006.
31 TI3004, MA1101, MA1201, EL2205, EL2006.
32 EL3015, EL2006, EL2005.
33 EP3272, EL3015, TI3004.
34 EP3070, EP3071, EP3073.
35 KU206X, EP3071, TI3004.
36 EP3072, EL2001, EL2005, EP3071.
37 EP3074, EL2001, EL2006, EP3071, EP3073.
38 EP3076, EP3075, EL2006.
39 EP4096, KU1011, EP3076, EP3074.
40 XXLING, EP3076.
41 EP4071, EP3071, MS2060, EP3272.
42 EP4073, KU206X.
43 RP4077, EP3075, EL2001, EP3076.
44 EP4091, EP4096.
45 XXMANJ, EP4096.
46 EP4099, EP4096.
47 EP4070, EP4096.
```

Output:

```
Command Prompt - main.exe X + v

C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
power

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1101, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
Semester 3: EP2091, MA2072, EL2001, EL2101, EL2142, MS2060.
Semester 4: EP2076, EP2094, KU2071, MA2074, EL2005, EL2205, EL2006.
Semester 5: EP3071, EP3073, EP3075, EP3095, EP3171, TI3004, EL3015.
Semester 6: EP3272, EP3070, KU206X, EP3072, EP3074, EP3076.
Semester 7: EP4096, XXLING, EP4071, EP4073, RP4077.
Semester 8: EP4091, XXMANJ, EP4099, EP4070.
```

8. telekomunikasi.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1101.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210, KU1102.
10 KU1202, KU1102, KU1011.
11 KI1002, MA1101, FI1101.
12 EL1200, MA1101.
13 ET2101, MA1201.
14 ET2103, EL1200.
15 ET2105, EL1200, MA1201.
16 ET2107, IF1210.
17 ET2109, MA1201.
18 ET2111, EL1200.
19 MA2072, MA1201.
20 ET2200, ET2101.
21 ET2201, ET2103.
22 ET2204, ET2105.
23 ET2206, ET2107.
24 ET2208, ET2109.
25 ET2212, ET2111.
26 ET2214, ET2111.
27 MA2074, MA2072.
```

Output:

```
Command Prompt - main.exe
C:\Users\Noler\Git\Tucil2_13519135\bin>main.exe
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
telekomunikasi

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1101, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
Semester 3: ET2101, ET2103, ET2105, ET2107, ET2109, ET2111, MA2072.
Semester 4: ET2200, ET2201, ET2204, ET2206, ET2208, ET2212, ET2214, MA2074.
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
```

9. tpb.txt

Input:

```
1 MA1101.
2 FI1101.
3 KU1101.
4 KU1102.
5 KU1011.
6 KU1024.
7 MA1201, MA1101.
8 FI1201, FI1101.
9 IF1210, KU1102.
10 KU1202, KU1102, KU1011.
11 KI1002, MA1101, FI1101.
12 EL1200, MA1101.
```

Output:

```
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
tpb

Mata kuliah yang diambil:
Semester 1: MA1101, FI1101, KU1101, KU1102, KU1011, KU1024.
Semester 2: MA1201, FI1201, IF1210, KU1202, KI1002, EL1200.
```

10. cyclic.txt (kasus graph dengan siklus)

Input:

```
1 MA1101, FI1101.
2 FI1101, MA1201.
3 MA1201, MA1101.
```

Output:

```
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
cyclic
Input tidak valid atau Graph bukan merupakan Graph acyclic.
Input soal/file yang ingin dikerjakan (tanpa '.txt') atau exit untuk keluar:
```

Alamat *Source Code* program

https://github.com/naufalsuryasumirat/Tucil2_13519135

Referensi

<https://www.geeksforgeeks.org/decrease-and-conquer/>

<https://algorithms.tutorialhorizon.com/topological-sort/>

https://en.wikipedia.org/wiki/Topological_sorting

<http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%2010%20-%20Decrease%20and%20Conquer%20Sorts%20and%20Graph%20Searches.htm>