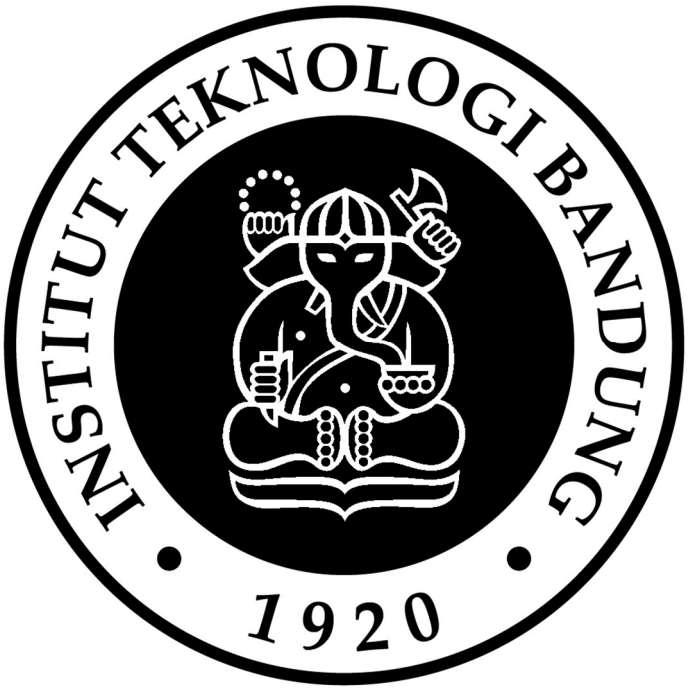
# LAPORAN TUGAS KECIL 2

# IF2211 STRATEGI ALGORITMA

## Penyusunan Rencana Kuliah dengan *Topological Sort*

## (Penerapan *Decrease and Conquer*)




**Nama** : **Christopher Justine William**

**NIM** : **13519006**

**Kelas** : **K-01**

**Bahasa** : **C++**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2021**

## A. Algoritma Topological Sort dan Kaitannya dengan Pendekatan Decrease and Conquer untuk Penyusunan Rencana Kuliah

**Langkah-langkah:**

1. Buatlah Directed Acyclic Graph (DAG) G yang setiap vertex nya terdiri dari kode mata kuliah. Setiap vertex juga menyimpan nilai dari derajat masuk dan derajat keluar.
2. Untuk setiap vertex yang mempunyai derajat masuk = 0, masukkan ke dalam array hasil L.
3. Kumpulan vertex yang terdapat di dalam array L merupakan mata kuliah yang dapat diambil pada suatu semester tertentu. Cetak ke layar untuk mendapakan daftar mata kuliah yang dapat diambil pada suatu semester tertentu.
4. Hapus untuk setiap vertex di dalam array L dari graf G. Hal ini akan mengakibatkan derajat masuk dari vertex successornya dikurangi satu.
5. Ulangi langkah 2 – 4 hingga graf G kosong. Setiap langkah perulangan merepresentasikan satu semester.

**Pendekatan Decrease and Conquer**

Algoritma ini mengurangi sejumlah himpunan persoalan menjadi persoalan yang lebih kecil dari permasalahan yang sama kemudian akan dicari solusi dari persoalan tersebut. Setiap himpunan persoalan akan terus dikurangi menjadi persoalan yang lebih kecil hingga semua solusi ditemukan.

Pada permasalahan penyusunan rencana kuliah dengan topological sort, vertex di dalam himpunan vertex graf G akan terus dikurangi hingga graf G kosong. Vertex yang dikeluarkan/dikurangi dari graf G menandakan bahwa vertex mata kuliah tersebut sudah disusun pada suatu semester tertentu. Sisa vertex graf G merupakan mata kuliah yang akan disusun pada semester yang lebih tinggi dari vertex yang telah dikeluarkan (karena adanya prasyarat). Hal ini mengakibatkan ukuran himpunan vertex graf G menjadi lebih kecil dari sebelumnya sehingga permasalahan ini dapat diselesaikan dengan pendekatan Decrease and Conquer. Karena untuk setiap perulangan jumlah vertex yang mempunyai derajat masuk = 0 dapat bervariasi yang menandakan jumlah mata kuliah yang dapat diambil pada satu semester maka permasalahan ini termasuk dalam decrease by variable size.

## B. Source Code

```
/* TopoSort.cpp */

#include <iostream>
#include <string>
#include <fstream>

#include "Node.hpp"
#include "Graph.hpp"
#include "ListNode.hpp"

using namespace std;

/*
    How to run :
    -   g++ -o TopoSort.exe TopoSort.cpp Graph.cpp ListNode.cpp Node.cpp
    -   ./TopoSort.exe
*/

void TopoSort(Graph G)
/* Melakukan pengurutan mata kuliah yang dapat diambil tiap semester dengan
menggunakan topological sorting */
/* Mata kuliah yang diambil sudah memenuhi prerequisites */
{
    int count = 1;
    while(G.getNumOfVertex() != 0){ /* Cek graph ksosong */
        ListNode L;
        for(int i = 0; i < G.getNumOfVertex(); i++){
            /* Cari graph yang memiliki derajar masuk = 0 dan masukkan ke
ListNode L */
            if(G.vertex[i]->getInDegree() == 0){
                L.add(*G.vertex[i]);
            }
        }
        cout << "Semester " << count << " :";
        for(int i = 0; i < L.getNumOfNode(); i++){
            /* Cetak ID untuk setiap node pada graph yang memiliki derajat
masuk = 0 */
            cout << " " << L.list[i]->getID();
            /* Hapus setiap node yang memiliki derajat masuk = 0 dari graph
*/
            G.delVertex(*L.list[i]);
        }
        count++;
        cout << endl;
    }
}


int main()
{
    cout << "Penyusunan Rencana Kuliah dengan Topological Sort" << endl;
    cout << "(Penerapan Decrease and Conquer)" << endl << endl;;

    Graph G;
```

```
    string namaFile;
    cout << "Masukkan nama file: ";
    cin >> namaFile;
    cout << endl;

    /* Baca graph mata kuliah dari file */
    ifstream File("../test/" + namaFile);
    if(File.fail()) exit(1);

    string s;
    while(getline(File,s)){
        string id = "";
        bool isFirst = true;
        Node* first = NULL;
        for (char x : s){
            if (!(x == ',' || x == '.' || x == ' ')){
                id = id + x;
            }
            else{
                if(x != ' '){
                    if(G.searchVertex(id) == NULL){
                        Node* tmp = new Node(id);
                        G.addVertex(*tmp);
                    }
                    if(isFirst){
                        first = G.searchVertex(id);
                        isFirst = false;
                    }
                    else{
                        G.addEdge(*G.searchVertex(id),*first);
                    }
                }
                id = "";
            }
        }
    }

    /* Melakukan topological sort pada graph yang sudah terbentuk */
    /* Argumen yang di pass ke fungsi TopoSort merupakan hasil copy
constructor dari graph G*/
    TopoSort(G);

    cout << endl;
    system("pause");

    return 0;
}
```

```
/* Node.hpp */

#ifndef NODE_H
#define NODE_H

#include <string>
using namespace std;

class Node{
    private:
        string id; /* Kode mata kuliah */
        Node** prevNode; /* Daftar node predecessor */
        Node** nextNode; /* Daftar node successor */
        int inDegree; /* Derajat masuk */
        int outDegree; /* Derajat keluar */

        friend class Graph;

    public:
        /* Default Constructor */
        Node();

        /* User Define Constructor */
        Node(string);

        /* Destructor */
        ~Node();

        /* Getter */
        string getID();  /* Return kode mata kuliah */
        int getInDegree(); /* Return jumlah derajat masuk */
        int getOutDegree(); /* Return jumlah derjat keluar */

        /* Menambahkan node successsor ke daftar node successsor */
        void addNextNode(Node&);

        /* Menghapus node successsor dari daftar node successsor */
        void delNextNode(Node&);

        /* Menambahkan node predecessor ke daftar node predecessor */
        void addPrevNode(Node&);

        /* Menghapus node predecessor dari daftar node predecessor */
        void delPrevNode(Node&);

        /* Mencetak daftar node predecessor sebagai prerequisites mata kuliah
*/
        void printPrevNode();
};

#endif
```

```
/* Node.cpp */

#include <iostream>
#include <string>
#include "Node.hpp"

Node::Node()
/* Default Constructor */
{

}

Node::Node(string id)
/* Copy Constructor */
{
    this->id = id;
    this->inDegree = 0;
    this->outDegree = 0;
    this->prevNode = NULL;
    this->nextNode = NULL;
}

Node::~Node()
/* Destructor */
{
    delete this->prevNode;
    delete this->nextNode;
}

string Node::getID()
/* Return kode mata kuliah*/
{
    return this->id;
}

int Node::getInDegree()
/* Return jumlah derajat masuk*/
{
    return this->inDegree;
}

int Node::getOutDegree()
/* Return jumlah derjat keluar */
{
    return this->outDegree;
}

void Node::addNextNode(Node& N)
/* Menambahkan node successsor ke daftar node successsor */
{
    this->outDegree++;
    Node** tmp = this->nextNode;
    this->nextNode = new Node*[this->outDegree];
    if(tmp != NULL){
        for(int i = 0; i < this->outDegree-1; i++){
            this->nextNode[i] = tmp[i];
        }
```

```
    }
    delete tmp;
    this->nextNode[this->outDegree-1] = &N;
}

void Node::delNextNode(Node& N)
/* Menghapus node successsor dari daftar node successsor */
{
    for(int i = 0; i < this->outDegree; i++){
        if(this->nextNode[i]->getID() == N.getID()){
            for(int j = i; j < this->outDegree-1; j++){
                this->nextNode[j] = this->nextNode[j+1];
            }
            break;
        }
    }
    this->outDegree--;
    Node** tmp = this->nextNode;
    this->nextNode = new Node*[this->outDegree];
    if(tmp != NULL){
        for(int i = 0; i < this->outDegree; i++){
            this->nextNode[i] = tmp[i];
        }
    }
    delete tmp;
}

void Node::addPrevNode(Node& N)
/* Menambahkan node predecessor ke daftar node predecessor */
{
    this->inDegree++;
    Node** tmp = this->prevNode;
    this->prevNode = new Node*[this->inDegree];
    if(tmp !=  NULL){
        for(int i = 0; i < this->inDegree-1; i++){
            this->prevNode[i] = tmp[i];
        }
    }
    delete tmp;
    this->prevNode[this->inDegree-1] = &N;
}

void Node::delPrevNode(Node& N)
/* Menghapus node predecessor dari daftar node predecessor */
{
    for(int i = 0; i < this->inDegree; i++){
        if(this->prevNode[i]->getID() == N.getID()){
            for(int j = i; j < this->inDegree-1; j++){
                this->prevNode[j] = this->prevNode[j+1];
            }
            break;
        }
    }
    this->inDegree--;
    Node** tmp = this->prevNode;
    this->prevNode = new Node*[this->inDegree];
    if(tmp != NULL){
```

```
        for(int i = 0; i < this->inDegree; i++){
            this->prevNode[i] = tmp[i];
        }
    }
    delete tmp;
}

void Node::printPrevNode()
/* Mencetak daftar node predecessor sebagai prerequisites mata kuliah */
{
    for(int i = 0; i < this->inDegree; i++){
        cout << this->prevNode[i]->getID();
        if(i != this->inDegree-1){
            cout << ", ";
        }
    }
}
```

```
/* Graph.hpp */

#ifndef GRAPH_H
#define GRAPH_H

#include "Node.hpp"

class Graph{
    private:
        Node** vertex; /* Daftar semua mata kuliah */
        int numOfVertex; /* Jumlah semua mata kuliah */

        friend void TopoSort(Graph);

    public:
        /* Default Constructor */
        Graph();

        /* Copy Constructor */
        Graph(const Graph&);

        /* Destructor */
        ~Graph();

        /* Getter */
        int getNumOfVertex(); /* Return jumlah semua mata kuliah */

        /* Menambahkan mata kuliah ke daftar mata kuliah */
        void addVertex(Node&);

        /* Menghapus mata kuliah dari daftar mata kuliah */
        void delVertex(Node&);

        /* Menambahkan edge/prerequisites mata kuliah */
        /* Node src merupakan prerequisites dari node dest */
        void addEdge(Node& src, Node& dest);

        /* Menghapus edge/prerequisites mata kuliah */
        /* Node src merupakan prerequisites dari node dest */
        void delEdge(Node& src, Node& dest);

        /* Mencetak graph ke layar */
        void print();

        /* Return address dari node yang mengandung ID mata kuliah yang
dicari */
        Node* searchVertex(string);

};

#endif
```

```
/* Graph.cpp */

#include <iostream>
#include <string>
#include "Graph.hpp"

Graph::Graph()
/* Default Constructor */
{
    this->numOfVertex = 0;
    this->vertex = NULL;
}

Graph::Graph(const Graph& G)
/* Copy Constructor */
{
    this->numOfVertex = G.numOfVertex;
    this->vertex = new Node*[this->numOfVertex];
    for(int i = 0; i < this->numOfVertex; i++){
        this->vertex[i] = new Node(G.vertex[i]->getID());
    }
    for(int i = 0; i < this->numOfVertex; i++){
        for(int j = 0; j < G.vertex[i]->outDegree; j++){
            this->addEdge(*this->vertex[i], *this->searchVertex(G.vertex[i]-
>nextNode[j]->getID()));
        }
    }
}

Graph::~Graph()
/* Destructor */
{
    for(int i = 0; i < this->numOfVertex; i++)

    {
        delete this->vertex[i];
    }
    delete this->vertex;
}

int Graph::getNumOfVertex()
/* Return jumlah semua mata kuliah */
{
    return this->numOfVertex;
}

void Graph::addVertex(Node& N)
/* Menambahkan mata kuliah ke daftar mata kuliah */
{
    this->numOfVertex++;
    Node** tmp = this->vertex;
    this->vertex = new Node*[this->numOfVertex];
    if(tmp != NULL){
        for(int i = 0; i < this->numOfVertex-1; i++){
            this->vertex[i] = tmp[i];
        }
    }
```

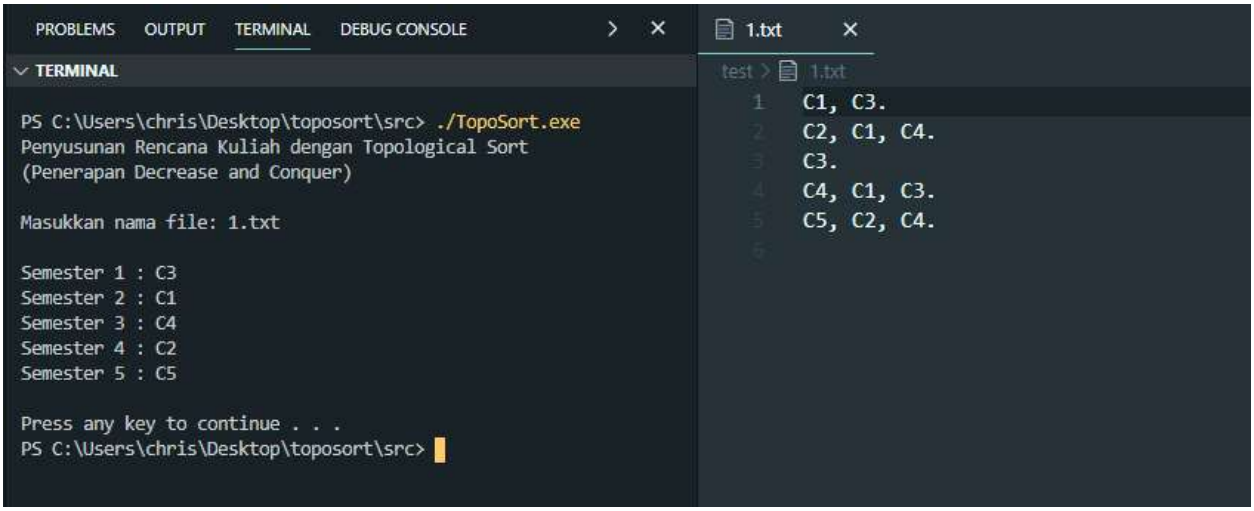
```
    delete tmp;
    this->vertex[this->numOfVertex-1] = &N;
}

void Graph::delVertex(Node& node)
/* Menghapus mata kuliah dari daftar mata kuliah */
{
    Node* N = this->searchVertex(node.getID());
    while(N->outDegree > 0){
        this->delEdge(*N,*N->nextNode[0]);
    }
    while(N->inDegree > 0){
        this->delEdge(*N->prevNode[0], *N);
    }
    for(int i = 0; i < this->numOfVertex; i++){
        if(this->vertex[i] == N){
            for(int j = i; j < this->numOfVertex-1;j++){
                this->vertex[j] = this->vertex[j+1];
            }
            break;
        }
    }
    delete N;
    this->numOfVertex--;
    Node** tmp = this->vertex;
    this->vertex = new Node*[this->numOfVertex];
    for(int i = 0; i < this->numOfVertex; i++){
        this->vertex[i] = tmp[i];
    }
    delete tmp;
}

void Graph::addEdge(Node& src, Node& dest)
/* Menambahkan edge/prerequisites mata kuliah */
/* Node src merupakan prerequisites dari node dest */
{
    src.addNextNode(dest);
    dest.addPrevNode(src);
}

void Graph::delEdge(Node& src, Node& dest)
/* Menghapus edge/prerequisites mata kuliah */
/* Node src merupakan prerequisites dari node dest */
{
    src.delNextNode(dest);
    dest.delPrevNode(src);
}

void Graph::print()
/* Mencetak graph ke layar */
{
    for(int i = 0; i < this->numOfVertex; i++){
        cout << this->vertex[i]->getID() << "(" << this->vertex[i]-
>getInDegree() << "," <<this->vertex[i]->getOutDegree() << ")" << " : ";
        this->vertex[i]->printPrevNode();
        cout << endl;
    }
```

```
}

Node* Graph::searchVertex(string S)
/* Return address dari node yang mengandung ID mata kuliah yang dicari */
{
    for(int i = 0; i < this->numOfVertex; i++){
        if(this->vertex[i]->getID() == S){
            return this->vertex[i];
        }
    }
    return NULL;
}
```

```
/* ListNode.hpp */

#ifndef LIST_NODE_H
#define LIST_NODE_H

#include "Node.hpp"
#include "Graph.hpp"

class ListNode{
    private:
        Node** list; /* Daftar semua node */
        int numOfNode; /* Jumalh semua node */

        friend void TopoSort(Graph);

    public:
        /* Default Construktor */
        ListNode();

        /* Destruktor */
        ~ListNode();

        /* Getter */
        int getNumOfNode(); /* Return jumlah semua node */

        /* Menambahkan node ke daftar node */
        void add(Node&);

};

#endif
```

```
/* ListNode.cpp */

#include <iostream>
#include <string>
#include "ListNode.hpp"

ListNode::ListNode()
/* Default Construktor */
{
    this->numOfNode = 0;
    this->list = NULL;
}

ListNode::~ListNode()
/* Destruktor */
{
    delete list;
}

int ListNode::getNumOfNode()
/* Return jumlah semua node */
{
    return this->numOfNode;
}

void ListNode::add(Node& N)
/* Menambahkan node ke daftar node */
{
    this->numOfNode++;
    Node** tmp = this->list;
    this->list = new Node*[this->numOfNode];
    if(tmp != NULL){
        for(int i = 0; i < this->numOfNode-1; i++){
            this->list[i] = tmp[i];
        }
    }
    delete tmp;
    this->list[this->numOfNode-1] = &N;
}
```

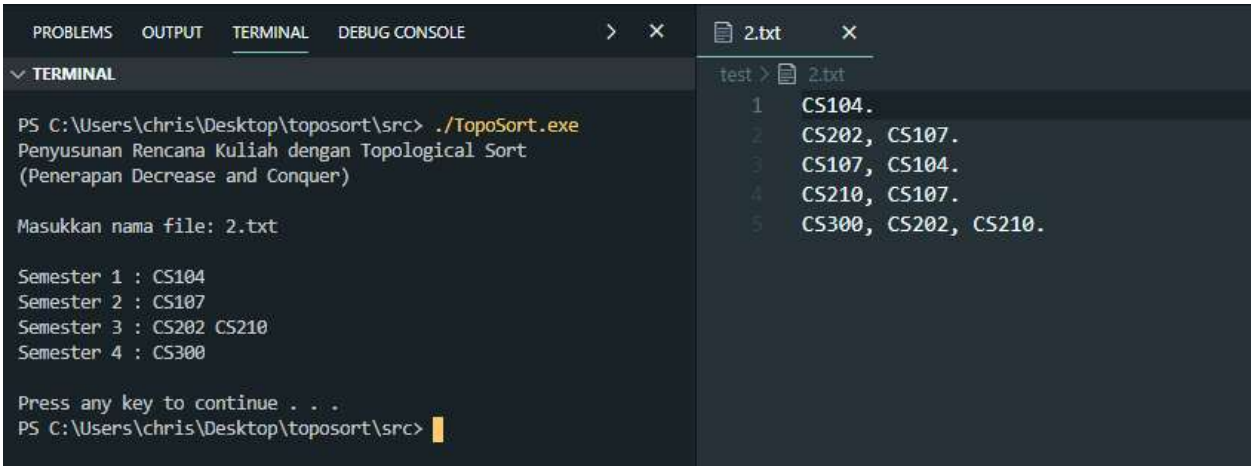
## C. Input dan Output



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
TERMINAL

PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 1.txt

Semester 1 : C3
Semester 2 : C1
Semester 3 : C4
Semester 4 : C2
Semester 5 : C5

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

1.txt

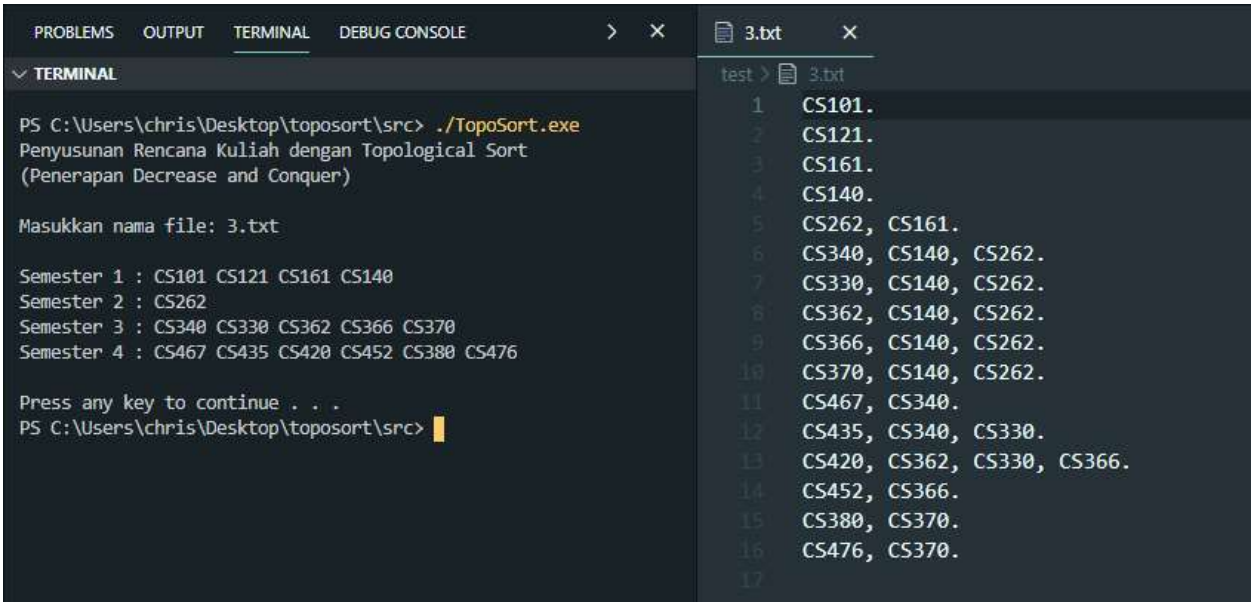
```
C1, C3.
C2, C1, C4.
C3.
C4, C1, C3.
C5, C2, C4.
```



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
TERMINAL

PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 2.txt

Semester 1 : CS104
Semester 2 : CS107
Semester 3 : CS202 CS210
Semester 4 : CS300

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

2.txt

```
CS104.
CS202, CS107.
CS107, CS104.
CS210, CS107.
CS300, CS202, CS210.
```



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
TERMINAL

PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 3.txt

Semester 1 : CS101 CS121 CS161 CS140
Semester 2 : CS262
Semester 3 : CS340 CS330 CS362 CS366 CS370
Semester 4 : CS467 CS435 CS420 CS452 CS380 CS476

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

3.txt

```
CS101.
CS121.
CS161.
CS140.
CS262, CS161.
CS340, CS140, CS262.
CS330, CS140, CS262.
CS362, CS140, CS262.
CS366, CS140, CS262.
CS370, CS140, CS262.
CS467, CS340.
CS435, CS340, CS330.
CS420, CS362, CS330, CS366.
CS452, CS366.
CS380, CS370.
CS476, CS370.
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

TERMINAL

```
PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 4.txt

Semester 1 : CS161 CS225
Semester 2 : CS271 CS162
Semester 3 : CS261 CS290
Semester 4 : CS344 CS325 CS372 CS361 CS362 CS340
Semester 5 : CS492 CS493 CS467

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

4.txt

test > 4.txt

```
CS161.
CS225.
CS271, CS161.
CS162, CS161.
CS261, CS162.
CS290, CS162.
CS344, CS271, CS261.
CS325, CS225, CS261.
CS372, CS271, CS261.
CS361, CS261.
CS362, CS261.
CS340, CS290.
CS492, CS344.
CS493, CS344.
CS467, CS344, CS361, CS362.
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

TERMINAL

```
PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 5.txt

Semester 1 : CS1
Semester 2 : CS2
Semester 3 : CS3 CS4 CS5
Semester 4 : CS6 CS7
Semester 5 : CS8 CS9
Semester 6 : CS10

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

5.txt

test > 5.txt

```
CS1.
CS2, CS1.
CS3, CS2.
CS4, CS2.
CS5, CS2.
CS6, CS4.
CS7, CS4.
CS8, CS6, CS7.
CS9, CS7.
CS10, CS5, CS9.
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

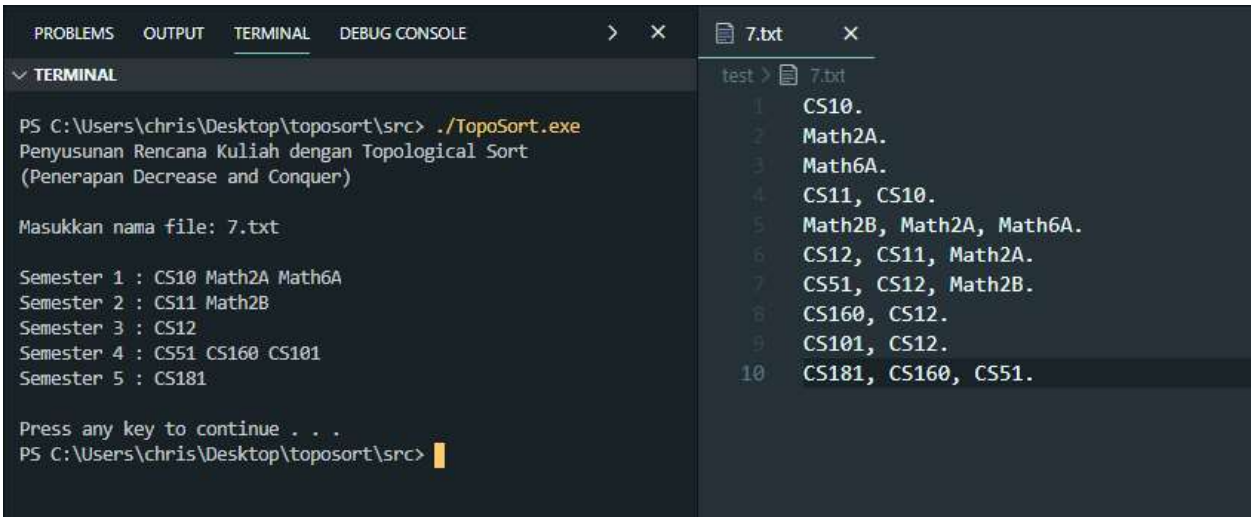
TERMINAL

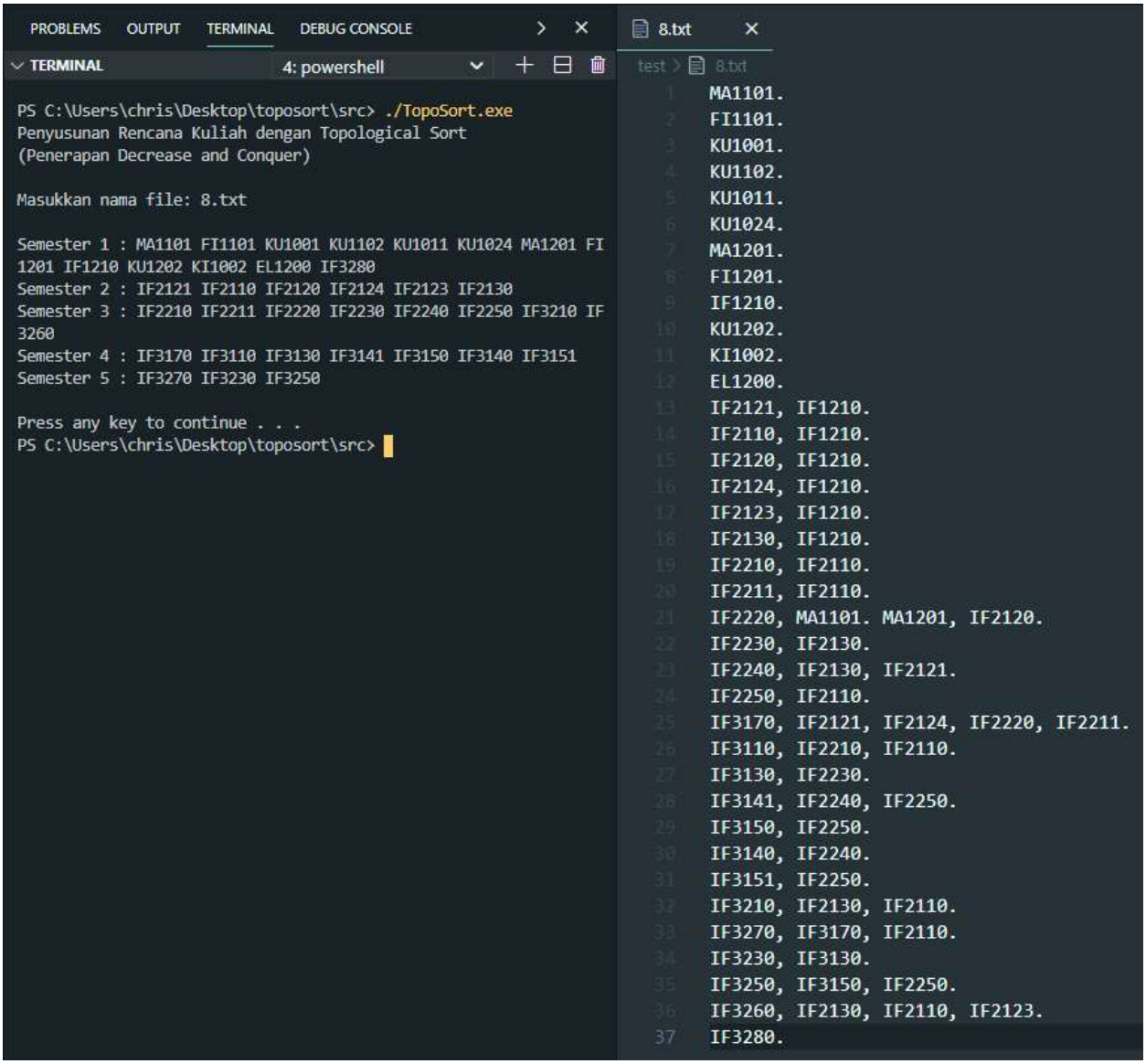
```
PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)

Masukkan nama file: 6.txt

Semester 1 : CS150 CS170
Semester 2 : CS250 CS270
Semester 3 : CS381 CS361
Semester 4 : CS390 CS350 CS355

Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
```

6.txt

test > 6.txt

```
CS150.
CS170.
CS250, CS150.
CS270, CS170.
CS381, CS150, CS270.
CS361, CS250, CS270.
CS390, CS381.
CS350, CS361.
CS355, CS381.
```


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
TERMINAL
PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)
Masukkan nama file: 7.txt
Semester 1 : CS10 Math2A Math6A
Semester 2 : CS11 Math2B
Semester 3 : CS12
Semester 4 : CS51 CS160 CS101
Semester 5 : CS181
Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
7.txt
test > 7.txt
CS10.
Math2A.
Math6A.
CS11, CS10.
Math2B, Math2A, Math6A.
CS12, CS11, Math2A.
CS51, CS12, Math2B.
CS160, CS12.
CS101, CS12.
CS181, CS160, CS51.


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
TERMINAL
4: powershell
PS C:\Users\chris\Desktop\toposort\src> ./TopoSort.exe
Penyusunan Rencana Kuliah dengan Topological Sort
(Penerapan Decrease and Conquer)
Masukkan nama file: 8.txt
Semester 1 : MA1101 FI1101 KU1001 KU1102 KU1011 KU1024 MA1201 FI1201 IF1210 KU1202 KI1002 EL1200 IF3280
Semester 2 : IF2121 IF2110 IF2120 IF2124 IF2123 IF2130
Semester 3 : IF2210 IF2211 IF2220 IF2230 IF2240 IF2250 IF3210 IF3260
Semester 4 : IF3170 IF3110 IF3130 IF3141 IF3150 IF3140 IF3151
Semester 5 : IF3270 IF3230 IF3250
Press any key to continue . . .
PS C:\Users\chris\Desktop\toposort\src>
8.txt
test > 8.txt
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.
MA1201.
FI1201.
IF1210.
KU1202.
KI1002.
EL1200.
IF2121, IF1210.
IF2110, IF1210.
IF2120, IF1210.
IF2124, IF1210.
IF2123, IF1210.
IF2130, IF1210.
IF2210, IF2110.
IF2211, IF2110.
IF2220, MA1101. MA1201, IF2120.
IF2230, IF2130.
IF2240, IF2130, IF2121.
IF2250, IF2110.
IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140, IF2240.
IF3151, IF2250.
IF3210, IF2130, IF2110.
IF3270, IF3170, IF2110.
IF3230, IF3130.
IF3250, IF3150, IF2250.
IF3260, IF2130, IF2110, IF2123.
IF3280.

**D. Drive**

https://drive.google.com/drive/folders/10Ali93URULQ8Hf_AI6-205ieUTJdCsIY?usp=sharing

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