# Laporan Tugas Kecil 2

# Pengaturan Jadwal Kelas Dengan *Topological Sort*

Shifa Salsabiila 13519106

Program Studi Teknik Informatika

IF2211: Strategi Algoritma

Dr. Nur Ulfa Maulidevi, S.T., M.Sc.

# Daftar Isi

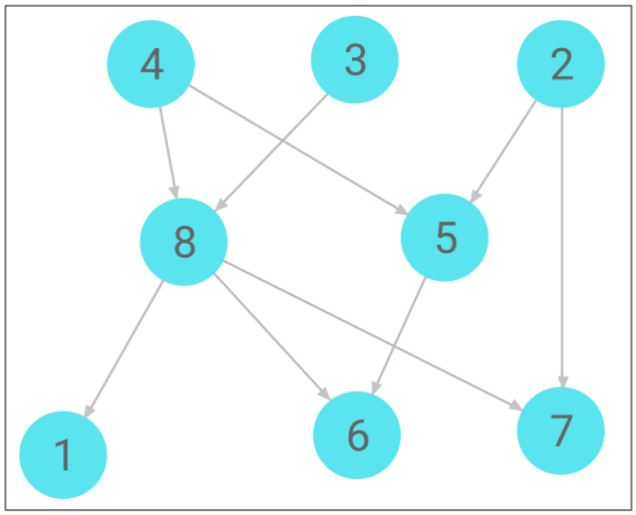
# I. Algoritma

Tujuan dari program ini adalah untuk menyusun rencana pengambilan mata kuliah. Banyak mata kuliah yang memiliki prarsyarat sebelum dapat diambil. Hubungan keterkaitan prasyaray antar mata kuliah ini dapat direpresentasikan dalam bentuk *Directed Acyclic Graph* (DAG). Pada program sederhana ini, diberikan tambahan fitur yaitu untuk memperhitungkan juga SKS suatu mata kuliah dan dapat menerima masukan jumlah SKS maksimum yang bisa diambil oleh seorang mahasiswa.

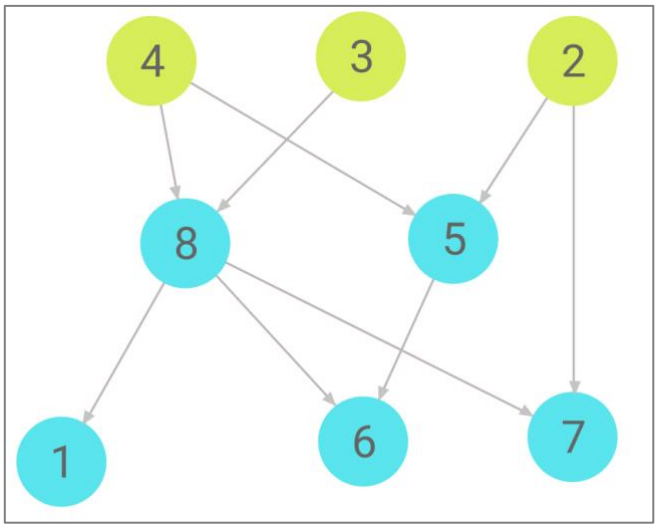
Untuk menyelesaikan permasalahan ini, digunakan pendekatan *topological sort*. Permasalahan dapat digambarkan dalam bentuk graf, dengan *node* merepresentasikan nama/kode mata kuliah yang harus diambil, dan *directed edge* merepresentasikan hubungan *prerequisite* dari satu mata kuliah ke mata kuliah yang lain. Suatu mata kuliah dikatakan bisa diambil pada satu semester (iterasi) tertentu, apabila derajat-masuknya sama dengan nol. Ketika suatu mata kuliah bisa diambil pada semester tertentu, maka mata kuliah tersebut akan dihapuskan dari graf dan derajat-masuk mata kuliah lain yang terhubung padanya akan dikurangi satu.

Pada setiap iterasi, bisa saja terdapat lebih dari satu mata kuliah yang tidak memiliki *prerequisite*, sehingga bisa lebih dari satu mata kuliah yang dihapus. Dalam kondisi tersebut, maka penghapusan seluruh *edge* akan dilakukan secara bersamaan di akhir iterasi, untuk menghindari terjadinya pengurangan *edge* sebelum pengecekan pada semua *node* dalam satu iterasi berakhir.

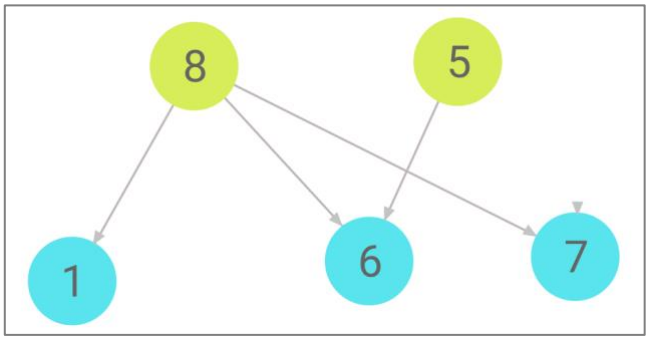
Penerapan algoritma *topological sort* pada penyelesaian masalah pengurutan mata kuliah ini merupakan implementasi algoritma *decrease and conquer* dengan varian *decrease by a variable size*. Ini berarti pada setiap iterasinya, upapersoalan direduksi dengan ukuran yang bisa bervariasi. Hal ini disebabkan oleh adanya ketentuan bahwa dalam satu semester (satu iterasi), jumlah mata kuliah yang diambil boleh lebih dari satu, asalkan mata kuliah yang bersangkutan tidak lagi memiliki *prerequisite* yang belum diambil. Dengan pengertian tersebut, jumlah mata kuliah yang dapat diambil pada setiap semesternya dapat bervariasi, yang artinya node dalam graf pada setiap iterasinya juga dapat berkurang dengan jumlah yang bervariasi. Berikut contoh proses berjalannya program:



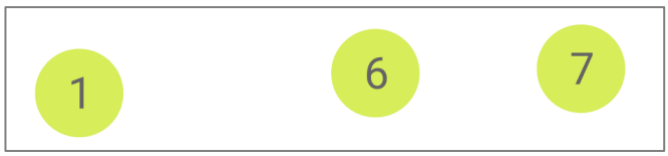

**Gambar 1.1** Contoh DAG persoalan pengambilan mata kuliah




**Gambar 1.2** Iterasi 1




**Gambar 1.3** Iterasi 2




**Gambar 1.4** Iterasi 3

## II. *Source Program*

Bahasa: C++

### ADT Graph

```
#ifndef GRAPH_HPP
#define GRAPH_HPP
#include <iostream>
#include <string>
using namespace std;

typedef struct tadrNode *adrNode;
typedef struct tadrSuccNode *adrSuccNode;
typedef struct tadrNode {
    int NPred;
    string Id;
    int Score;
    adrNode Next;
    adrSuccNode Trail;
} Node;
typedef struct tadrSuccNode {
    adrNode Succ;
    adrSuccNode NextT;
} SuccNode;
typedef struct {
    adrNode First;
} Graph;

#define First(G) (G).First
#define Id(P) (P)->Id
#define Score(P) (P)->Score
#define NPred(P) (P)->NPred
#define Trail(P) (P)->Trail
#define Succ(Pt) (Pt)->Succ
#define NextT(Pt) (Pt)->NextT
#define Next(P) (P)->Next

adrNode AlokNode(string X, int S);
void DealokNode(adrNode P);
adrSuccNode AlokSuccNode(adrNode Pn);
void DealokSuccNode(adrSuccNode Pt);
void CreateGraph (Graph *G, string X, int S);
adrNode SearchNode(Graph G, string X);
adrSuccNode SearchEdge(Graph G, string prec, string succ);
void InsertNode(Graph *G, string X, int S, adrNode *Pn);
void DeleteNode(Graph *G, string X);
void InsertEdge(Graph *G, string prec, string succ);
void DeleteAllEdges(Graph *G, adrNode Pn);

#endif
```

```
adrNode AlokNode(string X, int S) {
    /*KAMUS*/
    Node *P = (Node*) malloc(sizeof(Node));
    /*ALGORITMA*/
    if (P != NULL) {
        Id(P) = X;
        Score(P) = S;
        NPred(P) = 0;
        Trail(P) = NULL;
        Next(P) = NULL;
    }
    return P;
}
```

```
void DealokNode(adrNode P) {
    /*KAMUS*/
    /*ALGORITMA*/
    free(P);
}
```

```
adrSuccNode AlokSuccNode(adrNode Pn) {
    /*KAMUS*/
    adrSuccNode Pt;
    /*ALGORITMA*/
    Pt = (SuccNode*) malloc (sizeof(SuccNode));
    if (Pt != NULL) {
        Succ(Pt) = Pn;
        NextT(Pt) = NULL;
    }
    return Pt;
}
```

```
void DealokSuccNode(adrSuccNode Pt) {
    /*KAMUS*/
    /*ALGORITMA*/
    free(Pt);
}
```

```
void CreateGraph (Graph *G, string X, int S) {
    /*KAMUS*/
    adrNode P;
    /*ALGORITMA*/
    P  = AlokNode(X, S);
    First(*G) = P;
}
```

```
adrNode SearchNode(Graph G, string X) {
    /*KAMUS*/
    adrNode P;
    /*ALGORITMA*/
    P = First(G);
    while (P != NULL && Id(P) != X){
        P = Next(P);
    }
    return P;
}
```

```
adrSuccNode SearchEdge(Graph G, string prec, string succ) {
    /*KAMUS*/
    adrNode P;
    adrSuccNode T;
    /*ALGORITMA*/
    P = SearchNode(G, prec);

    if (P == NULL){
        return NULL;
    } else{
        T = Trail(P);
        if (T == NULL){
            return NULL;
        } else{
            while ((Id(Succ(T)) != succ) && (NextT(T) != NULL)) {
                T = NextT(T);
            }
            if (NextT(T) == NULL){
                return NULL;
            } else{
                return T;
            }
        }
    }
}
```

```
void InsertNode(Graph *G, string X, int S, adrNode *Pn) {
    /*KAMUS*/
    adrNode LastG;
    /*ALGORITMA*/
    LastG = First(*G);
    *Pn = AlokNode(X, S);
    if (*Pn != NULL){
        while (Next(LastG) != NULL){
            LastG = Next(LastG);
        }
        Next(LastG) = *Pn;
    }
}
```

```
void DeleteNode(Graph *G, string X) {
    /*KAMUS*/
    adrNode P;
    adrNode Prev;
    /*ALGORITMA*/
    P = SearchNode(*G, X);
    Prev = First(*G);

    //Removing from main list
    if (P != NULL) {
        if (P == First(*G)) {
            First(*G) = Next(P);
        } else {
            while (Next(Prev)!=P) {
                Prev = Next(Prev);
            }
            Next(Prev) = Next(P);
        }
        //Removing all edges that P is connected to
        DeleteAllEdges(G, P);
    }
}
```

```
void InsertEdge(Graph *G, string prec, string succ) {
    /*KAMUS*/
    adrNode Pprec;
    adrNode Psucc;
    adrSuccNode T;
    /*ALGORITMA*/
    Pprec = SearchNode(*G,prec);
    Psucc = SearchNode(*G,succ);
    if (SearchEdge(*G,prec,succ) == NULL){
        T = Trail(Pprec);
        if (T == NULL){
            Trail(Pprec) = AlokSuccNode(Psucc);
        }
        else{
            while (NextT(T) != NULL){
                T = NextT(T);
            }
            NextT(T) = AlokSuccNode(Psucc);

        }
        NPred(Psucc)++;
    }
}
```

```
void DeleteAllEdges(Graph *G, adrNode Pn) {
    /*KAMUS*/
    adrSuccNode Pt;
```

```
    /*ALGORITMA*/
    Pt = Trail(Pn);
    while (Pt != NULL) {
        NPred(Succ(Pt)) -= 1;
        Pt = NextT(Pt);
    }
}
```

**Pembacaan File**

```
void readFile(string path, Graph *G) {
    /*KAMUS*/
    ifstream fileInput;
    string x;
    string* array;
    int lines_read;

    /*ALGORITMA*/
    lines_read = 0;
    fileInput.open(path);
    array = new string[100];
    if (!fileInput) {
        cerr << "File not found!" << endl;
    }

    while (getline(fileInput, x)) {
        readLine(array, x, G, lines_read);
        lines_read += 1;
    }
}
```

```
void readLine(string* array, string line, Graph *G, int lines_read) {
    /*KAMUS*/
    string delims = ",. ";
    size_t pos = 0;
    string token;
    int i = 0;

    /*ALGORITMA*/
    while ((pos = line.find_first_of(delims)) != string::npos) {
        token = line.substr(0, pos);
        if (token!="") {
            array[i] = token;
            i += 1;
        }
        line.erase(0, pos + 1);
    }
    addLineToGraph(array, i, G, lines_read);
}
```

```
string* separateCourse(string strInput) {
    /*KAMUS*/
    string* result;
    string delims = "-";
    size_t pos = 0;
    string token;
    int i = 0;
    string course;

    /*ALGORITMA*/
    course = addMark(strInput);
    result = new string[2];
    while ((pos = course.find_first_of(delims)) != string::npos) {
        token = course.substr(0, pos);
        if (token!="") {
            //Memisahkan antara kode matkul dan SKS matkul;
            result[i] = token;
            i += 1;
        }
        course.erase(0, pos + 1);
    }
    return result;
}
```

```
string addMark(string x) {
    /*KAMUS*/
    /*ALGORITMA*/
    return x + "-";
}
```

```
int let2num(char c) {
    /*KAMUS*/
    /*ALGORITMA*/
    if (c=='1') {
        return 1;
    } if (c=='2') {
        return 2;
    } if (c=='3') {
        return 3;
    } if (c=='4') {
        return 4;
    } if (c=='5') {
        return 5;
    } if (c=='6') {
        return 6;
    } if (c=='7') {
        return 7;
    } if (c=='8') {
        return 8;
```

```
    } if (c=='9') {
        return 9;
    }
    return 0;
}
```

```
int str2num(string n) {
    /*KAMUS*/
    int result = 0;
    int i = 0;
    /*ALGORITMA*/
    while (n[i]!=' ' && n[i]!='\0') {
        result = let2num(n[i]);
        //cout << n[i];
        i += 1;
    }
    return result;
}
```

**Pembuatan Graph**

```
void addLineToGraph(string* array, int n, Graph *G, int lines_read) {
    /*KAMUS*/
    adrNode P;
    string x  = array[0];
    string* separated = separateCourse(x);
    string code = separated[0];
    int score = str2num(separated[1]);
    /*ALGORITMA*/
    //First node
    if (lines_read == 0) {
        CreateGraph(G, code, score);
    } else {
        if (SearchNode(*G, code) == NULL) {
            InsertNode(G, code, score, &P);
        }
    }
    //Trail node
    for (int i=1; i<n; i++) {
        string y = array[i];
        string* separated2 = separateCourse(y);
        string code2 = separated2[0];
        int score2 = str2num(separated2[1]);
        if (SearchNode(*G, code2) == NULL) {
            InsertNode(G, code2, score2, &P);
        }
        InsertEdge(G, code2, code);
    }
}
```

**Penjadwalan (*Topological Sort*)**

```
void generateJadwal(Graph *G, int sem, bool batasSks, int maxSks) {
    /*KAMUS*/
    adrNode P;
    string* toDelete;
    int i, j;
    int sks;

    /*ALGORITMA*/
    i = 0, j = 0;
    sks = 0;
    P = First(*G);
    toDelete = new string[100];
    cout << "Semester " << sem << ": " ;
    while (P != NULL) {
        if (NPred(P)==0) { //Cek node dengan derajat masuk = 0
            if (!batasSks || (sks+Score(P)<=maxSks)) {
                toDelete[i] = Id(P); //Tambahkan ke list akan hapus
                sks += Score(P); //Tambahkan jumlah SKS yang diambil
                i += 1;
            }
        }
        P = Next(P);
    }
    printJadwal(toDelete, i-1);
    cout << "Total SKS semester ini: " << sks << endl;

    //Menghapus semua mata kuliah yang bisa diambil
    while (j<i) {
        DeleteNode(G, toDelete[j]);
        j += 1;
    }
    delete[] toDelete;
}
```

```
void printJadwal(string* matkul, int n) {
    /*KAMUS*/
    int i;
    /*ALGORITMA*/
    i = 0;
    while (i <= n) {
        cout << matkul[i];
        if (i != n) {
            cout << ", ";
        }
        i += 1;
    }
    cout << endl;
}
```

**Main Program**

```
int main () {
    /*KAMUS*/
    Graph G;
    int sem;
    bool batasSks;
    string denganBatas;
    int maxSks;
    string namaFile;

    /*ALGORITMA*/
    maxSks = 0;

    cout << "Masukkan nama file: "; //Menerima nama file input
    cin >> namaFile;
    namaFile = "../test/" + namaFile; //Membuat path

    cout << "Apakah ada batasan SKS setiap semester? (Y/N)\n";
    cin >> denganBatas; //Menerima apakah ada batasan SKS

    batasSks = denganBatas == "Y" || denganBatas == "y";
    if (batasSks) {
        cout << "Berapa batas maksimum SKS setiap semester?\n";
        cin >> maxSks; //Menerima SKS maksimum per semester
    }


    readFile(namaFile, &G); //Baca file
    sem = 1;
    cout << "\n----------------------------------------\n";
    cout << "Berikut rencana  studi yang dapat diambil:\n";
    if (batasSks) {
        cout << "SKS maksimum setiap semester: " << maxSks << endl;
    }
    cout << endl;
    while (First(G) != NULL) {
        generateJadwal(&G, sem, batasSks, maxSks); //TopSort
        sem += 1;
        cout << endl;
    }

    return 0;

}
```

## III. Hasil dan *Test Case*

Catatan pengeksekusian program:

- Masukkan nama file yang sudah ada pada folder test
- Format suatu kelas pada file test mengikuti bentuk:

  *<nama_kelas>−<SKS>*
- Pengguna dapat memilih untuk menambahkan parameter SKS maksimum untuk setiap semesternya atau tidak pada proses penjadwalan. Harap diperhatikan bahwa SKS maksimum yang dimasukkan tidak boleh lebih kecil dari SKS terbesar suatu kelas yang ada pada file test yang bersangkutan.

**Tabel 3.1** *Test case*

| Input | Output |
|---|---|
| C1-4, C3-4.<br>C2-2, C1-4, C4-3.<br>C3-4.<br>C4-3, C1-4, C3-3.<br>C5-2, C2-2, C4-3.<br><br>File: test.txt | Masukkan nama file: test.txt<br>Apakah ada batasan SKS setiap semester? (Y/N)<br>Y<br>Berapa batas maksimum SKS setiap semester?<br>4<br><br>----------<br>Berikut rencana studi yang dapat diambil:<br>SKS maksimum setiap semester: 4<br><br>Semester 1: C3<br>Total SKS semester ini: 4<br><br>Semester 2: C1<br>Total SKS semester ini: 4<br><br>Semester 3: C4<br>Total SKS semester ini: 3<br><br>Semester 4: C2<br>Total SKS semester ini: 2<br><br>Semester 5: C5<br>Total SKS semester ini: 2 |
| C2-2, C11-2.<br>C3-3.<br>C5-2.<br>C7-2.<br>C8-2, C3-3, C7-2.<br>C9-2, C8-2, C11-2.<br>C10-2, C3-3, C11-2.<br>C11-2, C5-2, C7-2.<br><br>File: test2.txt | Masukkan nama file: test2.txt<br>Apakah ada batasan SKS setiap semester? (Y/N)<br>Y<br>Berapa batas maksimum SKS setiap semester?<br>8<br><br>----------<br>Berikut rencana studi yang dapat diambil:<br>SKS maksimum setiap semester: 8<br><br>Semester 1: C3, C5, C7<br>Total SKS semester ini: 7<br><br>Semester 2: C11, C8<br>Total SKS semester ini: 4<br><br>Semester 3: C2, C9, C10<br>Total SKS semester ini: 6 |

```
C0-4.
C1-3, C0-4.
C2-2.
C3-4, C1-3.
C4-2, C0-4.
C5-4, C1-3, C2-2, C3-4.
C6-3, C5-4.
C7-4, C3-4, C6-3.
```

File: test3.txt

```
Masukkan nama file: test3.txt
Apakah ada batasan SKS setiap semester? (Y/N)
N

________________________________________
Berikut rencana  studi yang dapat diambil:

Semester 1: C0, C2
Total SKS semester ini: 6

Semester 2: C1, C4
Total SKS semester ini: 5

Semester 3: C3
Total SKS semester ini: 4

Semester 4: C5
Total SKS semester ini: 4

Semester 5: C6
Total SKS semester ini: 3

Semester 6: C7
Total SKS semester ini: 4
```

```
SR2101-4, MA1201-4.
MA1201-4, MA1101-4.
TA2102-3, MA1201-4.
SI2201-4, SR2101-4, FI1201-4.
MA1101-4.
FI1201-4, MA1101-4.
MB3101-3, TA2102-3, SI2201-4, EL2202-4.
AE2103-4, FI1201-4, KU1101-3, IF1203-3.
KU1101-3.
EL2202-4, AE2103-4.
IF1203-3, KU1101-3.
```

File: test4.txt

```
Masukkan nama file: test4.txt
Apakah ada batasan SKS setiap semester? (Y/N)
N

________________________________________
Berikut rencana  studi yang dapat diambil:

Semester 1: MA1101, KU1101
Total SKS semester ini: 7

Semester 2: MA1201, FI1201, IF1203
Total SKS semester ini: 11

Semester 3: SR2101, TA2102, AE2103
Total SKS semester ini: 11

Semester 4: SI2201, EL2202
Total SKS semester ini: 8

Semester 5: MB3101
Total SKS semester ini: 3
```

```
IF1101-3.
IF1102-3.
IF1201-2, IF1102-3, IF1103-4.
IF1103-4.
IF1202-4, IF1101-3, IF1102-3.
IF2201-3, IF1101-3, IF1201-2, IF2101-4.
IF2101-4, IF1201-2, IF1103-4.
IF3201-2, IF1202-4, IF3101-4.
IF3101-4, IF1102-3, IF2201-3.
IF3202-4, IF1201-2, IF3102-3.
IF4101-3, IF3202-4.
IF4102-3, IF1101-3, IF3202-4.
IF3102-3, IF2201-3.
IF4201-2, IF1103-4, IF4101-3.
```

File: test5.txt

```
Masukkan nama file: test5.txt
Apakah ada batasan SKS setiap semester? (Y/N)
Y
Berapa batas maksimum SKS setiap semester?
10

_____________________________________________
Berikut rencana  studi yang dapat diambil:
SKS maksimum setiap semester: 10

Semester 1: IF1101, IF1102, IF1103
Total SKS semester ini: 10

Semester 2: IF1201, IF1202
Total SKS semester ini: 6

Semester 3: IF2101
Total SKS semester ini: 4

Semester 4: IF2201
Total SKS semester ini: 3

Semester 5: IF3101, IF3102
Total SKS semester ini: 7

Semester 6: IF3201, IF3202
Total SKS semester ini: 6

Semester 7: IF4101, IF4102
Total SKS semester ini: 6

Semester 8: IF4201
Total SKS semester ini: 2
```

```
MA1101-4.
FI1201-4, MA1101-4.
MA1201-4, MA1101-4.
IF2101-3, MA1101-4, MA1201-4.
IF2201-3, IF2101-3, IF2102-4.
IF2202-4, MA1101-4, IF2101-3.
IF2102-4, MA1101-4, IF1201-3.
IF1201-3, IF1101-3.
IF1101-3.
IF3101-4, IF2201-3, IF2102-4.
IF3201-2, IF3101-4.
IF3202-4, IF3101-4.
IF4101-4, IF3101-4, IF3202-4.
```

File: test6.txt

```
Masukkan nama file: test6.txt
Apakah ada batasan SKS setiap semester? (Y/N)
Y
Berapa batas maksimum SKS setiap semester?
10

_____________________________________________
Berikut rencana  studi yang dapat diambil:
SKS maksimum setiap semester: 10

Semester 1: MA1101, IF1101
Total SKS semester ini: 7

Semester 2: FI1201, MA1201
Total SKS semester ini: 8

Semester 3: IF2101, IF1201
Total SKS semester ini: 6

Semester 4: IF2102, IF2202
Total SKS semester ini: 8

Semester 5: IF2201
Total SKS semester ini: 3

Semester 6: IF3101
Total SKS semester ini: 4

Semester 7: IF3201, IF3202
Total SKS semester ini: 6

Semester 8: IF4101
Total SKS semester ini: 4
```

```
IF001-4, IF002-3, IF008-4, IF006-2.
IF002-3.
IF003-4, IF007-2.
IF004-3, IF001-4, IF003-4, IF007-2.
IF005-1, IF001-4, IF006-2, IF010-4.
IF006-2, IF011-3.
IF007-2, IF001-4, IF011-3.
IF008-4.
IF009-3, IF003-4.
IF010-4.
IF011-3.
```

File: test7.txt

```
Masukkan nama file: test7.txt
Apakah ada batasan SKS setiap semester? (Y/N)
Y
Berapa batas maksimum SKS setiap semester?
12

----------------------------------------
Berikut rencana  studi yang dapat diambil:
SKS maksimum setiap semester: 12

Semester 1: IF002, IF008, IF010
Total SKS semester ini: 11

Semester 2: IF011
Total SKS semester ini: 3

Semester 3: IF006
Total SKS semester ini: 2

Semester 4: IF001
Total SKS semester ini: 4

Semester 5: IF007, IF005
Total SKS semester ini: 3

Semester 6: IF003
Total SKS semester ini: 4

Semester 7: IF004, IF009
Total SKS semester ini: 6
```

```
Masukkan nama file: test7.txt
Apakah ada batasan SKS setiap semester? (Y/N)
N

----------------------------------------
Berikut rencana  studi yang dapat diambil:

Semester 1: IF002, IF008, IF010, IF011
Total SKS semester ini: 14

Semester 2: IF006
Total SKS semester ini: 2

Semester 3: IF001
Total SKS semester ini: 4

Semester 4: IF007, IF005
Total SKS semester ini: 3

Semester 5: IF003
Total SKS semester ini: 4

Semester 6: IF004, IF009
Total SKS semester ini: 6
```

```
KU1110-2.
MA1101-4.
KI1102-4.
FI1103-4.
KI1202-4, KU1110-2, MA1101-4, KI1102-4.
KI1210-3, KU1110-2, KI1102-4, FI1103-4.
KI1211-3, KI1102-4.
KI12112-2, KU1110-2, KI1102-4.
KI2101-4, MA1101-4, KI1210-3.
KI2102-4, KI1210-3, KI1211-3.
KI2201-3, MA1101-4, KI2102-4.
```

File: test8.txt

```
Masukkan nama file: test8.txt
Apakah ada batasan SKS setiap semester? (Y/N)
Y
Berapa batas maksimum SKS setiap semester?
13

----------------------------------------
Berikut rencana  studi yang dapat diambil:
SKS maksimum setiap semester: 13

Semester 1: KU1110, MA1101, KI1102
Total SKS semester ini: 10

Semester 2: FI1103, KI1202, KI1211, KI12112
Total SKS semester ini: 13

Semester 3: KI1210
Total SKS semester ini: 3

Semester 4: KI2101, KI2102
Total SKS semester ini: 8

Semester 5: KI2201
Total SKS semester ini: 3
```

```
Masukkan nama file: test8.txt
Apakah ada batasan SKS setiap semester? (Y/N)
N

----------------------------------------
Berikut rencana  studi yang dapat diambil:

Semester 1: KU1110, MA1101, KI1102, FI1103
Total SKS semester ini: 14

Semester 2: KI1202, KI1210, KI1211, KI12112
Total SKS semester ini: 12

Semester 3: KI2101, KI2102
Total SKS semester ini: 8

Semester 4: KI2201
Total SKS semester ini: 3
```

**Tabel 3.2** Form penilaian mandiri

| No | 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. | √ | |

## IV. Alamat *Source Code*

Repository: https://github.com/salsabiilashifa11/TopSort