PENYUSUNAN RENCANA KULIAH DENGAN TOPOLOGICAL SORT (PENERAPAN DECREASE AND CONQUER)

LAPORAN TUGAS KECIL

Diajukan Untuk Memenuhi Tugas IF 2211 Strategi Algoritma
Semester II tahun 2020/2021

oleh

KEVIN KATSURA D. SITANGGANG 13519216

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

BAB I

DESKRIPSI MASALAH

1. ALGORITMA TOPOLOGICAL SORT

Topological Sort dalam graf berarah merupakan pengurutan linear pada simpul-simpul graf tersebut di mana jika misalnya pada busur uv yang mengarah dari simpul u ke v, maka simpul u akan terlebih dahulu muncul pada pengurutan dibandingkan simpul v. Pada kasus penyusunan rencana perkuliahan, beberapa mata kuliah lain perlu dipenuhi untuk mengambil suatu mata kuliah sehingga dibutuhkan penyelesaian suatu mata kuliah syarat sebelum pengambilan mata kuliah tujuan. Oleh karena itu, dengan *Topological Sort*, penyusunan rencana perkuliahan dimulai dari mata kuliah yang tidak memiliki *pre-requisites* hingga semua *pre-requisites* dari mata kuliah tujuan dapat terpenuhi.

Semua kasus yang dapat ditangani dengan *Topological Sort* merupakan *Directed Acyclic Graphs (DAG*) atau graf berarah asiklik. Hal ini berarti semua kasus yang direpresentasikan dengan graf dengan siklus melingkar pada *node-node* nya tidak dapat ditangani. Hal ini diakibatkan karena kontradiksi yang terjadi bahwa node-node yang berada pada siklus tersebut saling mempengaruhi sedangkan pada *Topological Sort*, suatu event harus terjadi setelah event yang menunjuknya. Oleh karena itu, tidak semua graf memiliki *Topological Sort*.

2. KAITAN TOPOLOGICAL SORT DENGAN PENDEKATAN DECREASE AND CONQUER

Decrease and Conquer adalah metode perancangan algoritma dengan mereduksi persoalan menjadi dua upa-persoalan (sub-problem) yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja. Pada metode ini, tidak kedua upa-persoalan hasil pembagian diselesaikan. Berbeda hal nya dengan *Devide and Conquer*, metode itu memproses setiap komponen yang dibaginya.

Kaitan antara *Topological Sort* dengan pendekatan *Decrease and Conquer* ini adalah pada *Topological Sort*, terdapat pengurangan *scope* permasalahan dari data input melalui fungsi rekursif. Pada kasus tugas ini, *Topological Sort* akan menampilkan mata kuliah dari semester paling awal yang tidak memiliki prerequisites (simpul masuk = 0) hingga ditampilkan mata kuliah tujuan di semester terakhir. *Topological Sort* melakukan pemanggilan fungsi *Topological Sort* itu sendiri dalam dirinya dengan data input selanjutnya akan dikurang data yang telah diproses. Pada tugas ini, jika simpul A telah diproses, maka input selanjutnya adalah simpul yang ditunjuk A tanpa ada lagi simpul A tersebut (pengurangan scope). Dari hal tersebut, dapat disimpulkan bahawa Topological Sort berkaitan dengan pendekatan *Decrease and Conquer*

BAB II

IMPLEMENTASI PROGRAM

2.1. FUNGSI DAN PROSEDUR

13519216-main.c

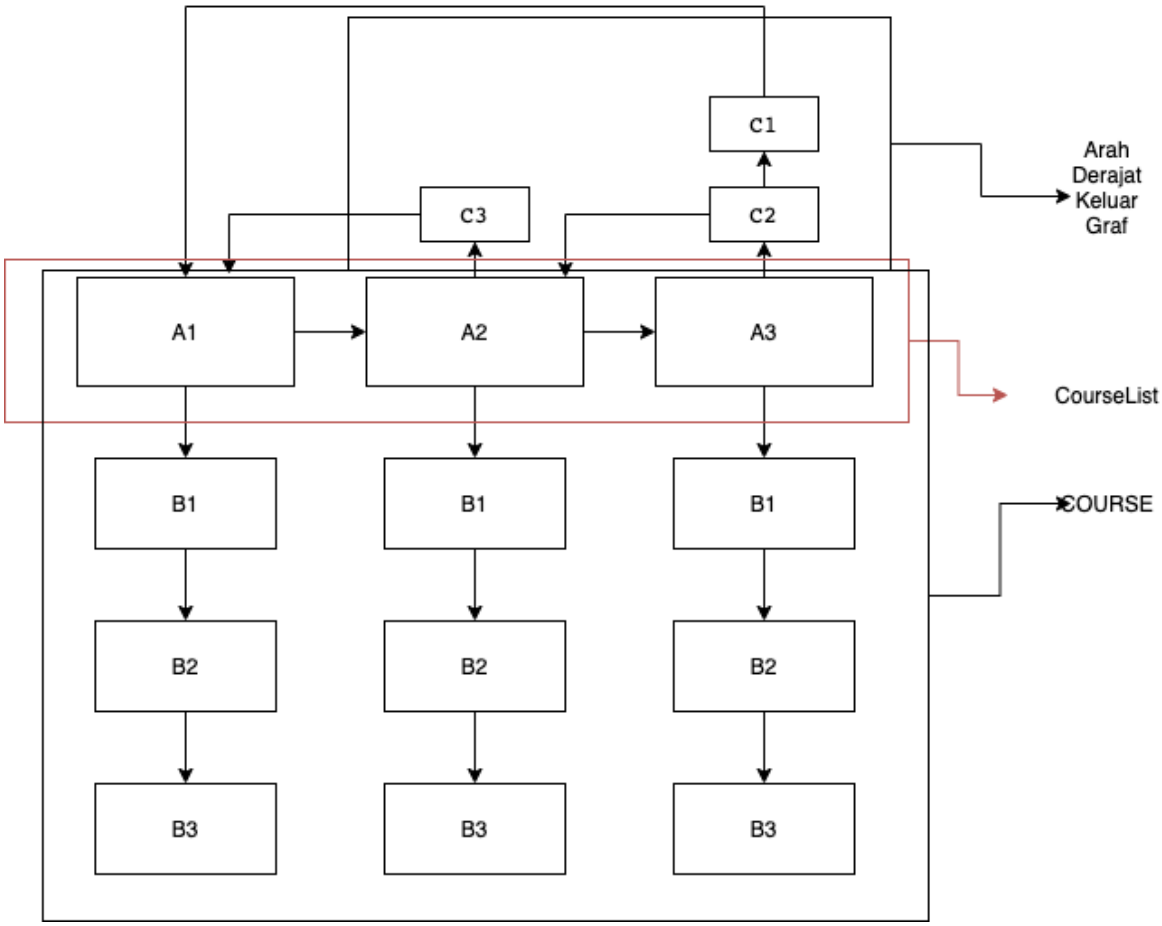
int main()	Proses dari awal hingga akhir program. Terdapat alokasi CourseList, List, pembuatan graph, pemanggilan fungsi DecreaseNConquer hingga Dealokasi *alocated memory*.
void DecreaseNConquer(address1 A1, int i)	Penampilan Semester untuk setiap mata kuliah. Fungsi ini menerapkan *Topological Sort.*

13519216-ADT.c

address1 AlokasiListCourse (address2 theCourse)	Mengalokasi CourseList dengan isi masuk (jumlah simpul masuk), *visitedNode, course, requisites,* dan *next*.
address2 AlokasiCourse (char X[], int panjang)	Mengalokasi COURSE dengan isi array nama, *length*, dan *next*.
void DealokasiListCourse (address1* P)	Membebaskan memori bertipe CourseList yang dialokasikan sebelumnya.
void DealokasiCourse (address2* P)	Membebaskan memori bertipe COURSE yang dialokasi sebelumnya.
void PrintCourse (address2 P)	Menampilkan nama dari mata kuliah yang berada di P.
boolean Compare2Courses (address2 P1, address2 P2)	Menghasilkan apakah kedua nama dari P1 dan P2 adalah sama.

2.2 PENJELASAN SINGKAT KONSEP PROGRAM

Pada program, terdapat 2 ADT yaitu CourseList dan COURSE. ADT COURSE akan menyimpan data dari mata kuliah seperti nama. ADT CourseList akan membentuk pointer ke ADT COURSE. CourseList merepresentasikan setiap kepada/mata kuliah pada tiap baris dari file input sedangkan COURSE merepresentasikan setiap syarat syarat yang ada pada setiap kepala/matakuliah tersebut. Hal ini diperjelas dengan gambar di bawah ini.



CourseList yang dialokasikan tadi bersama-sama dengan pembacaan file input akan digunakan juga dalam merepresentasikan *Directed Acyclic Graph*. Masing-masing CourseList ini merepresentasikan *node-node* yang ada pada graf. Dari gambar di atas, semua bagian "Arah Derajat Keluar Graf" Menunjukkan bahwa CourseList tersebut memiliki simpul keluar sejumlah berapa c_ yang keluar dari CourseList. Pada gambar diatas misalkan A3. A3 akan memiliki 2 simpul keluar yaitu c1 dan c2 yang masing-masing menunjuk ke CourseList A1 dan A2. Masing-masing CourseList tersebut akan mencatat berapa simpul masuk yang ada padanya.

Ketika program dijalankan, yang akan dilakukan program adalah memasukkan data yang ada pada file. Jumlah CourseList yang terurut ini akan merepresentasikan jumlah baris dan setiap Course merupakan prasyarat (dimulai dari COURSE ke-dua/ B2 pada gambar) dari mata kuliah pada setiap barisnya.

Setelah itu, akan dibentuk graf nya, akan dialokasikan CourseList baru yang menunjuk pada CourseList yang dibentuk sebelumnya. Simpul masuk akan dicatat pada setiap CourseList. Oleh karena itu, mata kuliah yang memiliki derajat masuk

adalah 0 akan menjadi start *Topological sort* yang dilakukan nantinya dan simpul yang tidak memiliki simpul keluar merupakan mata kuliah yang ingin diambil.

Setelah graf tersebut terbentuk, maka simpul dengan derajat masuk nol akan dimasukkan dalam fungsi *DecreaseNConque*r yang memprint hasil dari alokasi mata kuliah di masing-masing semester. Fungsi DecreaseNConquer akan berakhir ketika mencapai simpul berderajat keluar nol.

Setelah hasil ditampilkan, program akan mendealokasi semua *allocated memory* yang dilakukan sebelumnya.

2.3. SOURCE CODE PROGRAM

13519216-main.c

```c
#include "stdio.h"
#include "string.h"
#include "13519216-ADT.c"

//DEKLARASI PROCEDURE DecreaseNConquer
void DecreaseNConquer (address1 A1, int i); // Fungsi ini didefinisikan di bawah
fungsi main.
```

```c
// >>>>>>>>>>>>>>>> MAIN FUNCTION <<<<<<<<<<<<<<<<

int main(){
    // Deklarasi array of char dan inputFile
    FILE *inputFile;
    char namaFile[50]; // Penyimpanan nama dari file tanpa path
    char path[100]; // Penyimpanan path ke file ( 'data_uji/' + namaFile )

    // Input Nama File di folder 'data_uji' tanpa menggunakan ekstensi .txt
    printf("Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : ");
    scanf("%123s",namaFile); // input namafile dan assign ke 'namaFile'

    //concat nama file dengan ekstensi .txt
    strcat(namaFile,".txt");

    // assign ('data_uji/' + nama file) ke dalam path
    snprintf(path, sizeof(path), "../test/%s", namaFile);

    //Buka file dan assign ke dalam variabel 'inputFile'
    inputFile=fopen(path,"r");

    // Membaca huruf pertama dari file dan di-assign ke dalam CC

    //Deklarasi array buffer dan List L
    char buffer1[100];
    char buffer2[20];
    List L ;

    // Masukkan data ke dalam list dynamic
    int index,count,baris = 0;
    char CC ; // deklarasi variable CC untuk tempat peny
    int jumlahCourse = 0; //
    address1 A1,A2 ;
    address2 P2,P3,PFirst ;
```

```
// - >>>> KONSEP PROGRAM <<<< -
    /*
    KONSEP DARI LIST YANG DIBUAT ADALAH DENGAN MENGALOKASI LIST COURSE SEBAGAI
PUSAT(ADT CourseList).
    NEXT YANG ADA PADA LIST INI AKAN MERUJUK KE ADT CourseList LAIN.
    JADI SETIAP 'CourseList' INI MEREPRESENTASIKAN BANYAKNYA BARIS DALAM FILE TXT
YANG DIBACA.
    KEMUDIAN DALAM SETIAP 'CourseList' AKAN MERUJUK PADA 'ADT COURSE' DIMANA ADT
COURSE INI MEREPRESENTASIKAN SETIAP HURUF DALAM SETIAP BARIS.
    ELEMEN PERTAMA (ADT COURSE ) DALAM SETIAP 'CourseList' AKAN MERUJUK PADA KODE
KULIAH SEDANGKAAN NEXT DARI 'ADT COURSE' INI SELANJUTNYA ADALAH PRASYARATNYA DARI
KODE KULIAH TERSEBUT
    */

    // Perintah 'getc' akan membaca isi dari file huruf demi huruf
    CC = getc(inputFile); // Pembacaan karakter pertama dalam file
    do{
        if (CC==','){ // Kondisi ketika karakter yang dibaca dari file adalah koma
(,)
            jumlahCourse++ ; // jumlahCourse ini akan digunakan untuk mengecek
apakah dialokasikan 'CourseList' yang baru (ADT CourseList dapat di lihat dalam
'data.h')
            P2 = AlokasiCourse(buffer2,count); // Alokasi P2 dengan pointer ke ADT
COURSE (implementasi ADT COURSE terdapat di 'data.h)
            if (jumlahCourse == 1){
                PFirst =  P2 ; // Jika P2 yang dialokasikan adalah Kode Kuliah,
maka PFirst diassign dengan Alokasi ADT COURSE di atas (P2)
            }
            else{
                Next(P3) = P2 ; //Jika P2 yang dialokasikan adalah prasyarat kode
kuliah, maka Next dari P3 yang diassign sebelumnya akan diset menjadi P2
            }
            P3 = P2 ;
            count = 0 ;
        }
        else if (CC == '.') // Kondisi ketika karakter yang dibaca dari file adalah
titik(.)
        {
            if (jumlahCourse>= 1)
            {
                P2 = AlokasiCourse(buffer2,count);  // Jika bukan elemen pertama
pada COURSE dalam suatu lisCourseListt.
                Next(P3) = P2 ; // P3 yang telah ada sebelumnya di set next nya
menjadi P2 yang baru saja dialokasikan
            }
            else{
                PFirst = AlokasiCourse(buffer2,count); // Jika elemen pertama dalam
COURSE dalam suatu CourseList.
            }

            A1 = AlokasiListCourse(PFirst); // Dialokasikan CourseList dengan
elemen kepada adalah PFirst dan di-assign ke A1
ini akan selesai jika CC adalah EOF
```

```
if (baris==0){
                First(L) = A1 ; // Jika CourseList yang baru saja dialokasikan
masih baris pertama dalam file txt, maka akan diset menjadi First dari List (
penanda elemen pertama CourseList)
            }
            else{
                Next(A2) = A1 ; // Jika telah dialokasikan CourseList lain
sebelumnya, maka CourseList yang baru dialokasikan akan diset menjadi next dari
CourseList sebelumnya
            }
            A2 = A1;
            count = 0 ;
            jumlahCourse = 0 ;
            baris++;
        }
        else{ // Jika yang karakter yang dibaca yang terdapat pada CC bukan titik
maupun koma
            buffer2[count] = CC; // Assign karakter yang terdapat dalam CC ke array
buffer2 karakter demi karakter
            count++ ;
        }
        CC = getc(inputFile); // Membaca karakter selanjutnya dalam file
        if (CC == '\n' || CC == ' ')
        {
            CC = getc(inputFile); // Jika yang dibaca adalah Newline atau SPASI
maka akan diabaikan dan membaca karakter selanjutnya dalam file.
        }
    }while(CC != EOF); // Alokasi ini akan selesai jika CC adalah EOF
```

```
    /*
    SETELAH ALOKASI YANG SEBELUMNYA DILAKUKAN, MAKA AKAN DIALOKASIKAN LAGI LIST
UNTUK MASING-MASING CourseList YANG TELAH DIALOKASIKAN.
    MASING-MASING CourseList INI AKAN MENUNJUK PADA CourseList LAIN YANG MEMILIKI
PRASYARAT CourseList YANG SEDANG DIAKSES (KONSEP SAMA DENGAN GAMBAR GRAPH PADA SPEK
TUGAS).
    */
```

```
//  MEMBENTUK GRAF SEMPURNA
    address1 A3 ;
    A1 = First(L); // ASSIGN FIRST DARI CourseList
    P2 = Next(Course(A1)); // ASSIGN ELEMEN KEDUA(Prasyarat) DALAM CourseList
pertama
    while (A1 != Nil){  // A1 terhadap CourseList
        while (P2 != Nil){  // P2 terhadap setiap Course pada CourseList yang
sedang diakses
            A2 = First(L);
            boolean sama = false;
            while (!sama && A2 != Nil){ // Mengakses CourseList dengan A2
                sama = Compare2Courses(Course(A2),P2); // Mencompare apakah nama
dari Course(A2) dan nama dari P2 adalah sama
                if (sama) {
                    A3 = A2; // Akses CourseList A2
                    while (Requisites(A3) != Nil) // Dicari posisi ujung dari
Requisites dari elemen CourseList yang diakses untuk dialokasikan yang baru
                    {
                        A3 = Requisites(A3);
                    }
                    Requisites(A3) = AlokasiListCourse(Nil); // Mengalokasikan
posisi terujung dari Requisites CourseList yang diakses dengan CourseList kosong
(selanjutnya akan dialokasikan Next yang menunjuk ke CourseList utama yang memiliki
prasyarakat courselist ini)
                    Next(Requisites(A3)) = A1 ; // Set Next dari CourseList yang
baru dialokasikan dengan A1 (CourseList yang memiliki prasyarat A2)
                    Masuk(A1)++ ; // Mengincrement nilai dari simpul masuk
                }
                else{
                    A2 = Next(A2); // Jika tidak sama, maka akan diakses CourseList
selanjutnya
                }
            }
            P2 = Next(P2); // Akses Course selanjutnya dari CourseList yang diakses
        }
        A1 = Next(A1); // Akses CourseList selanjutnya
        if (A1 !=Nil){
            P2 = Next(Course(A1)); // Jika memang A1 sekarang tidak nil, maka akan
dialokasikan lagi P2 dengan COURSE Ke-2 pada A1
        }
    }
```

```
// MAIN SECTION
    // DECREASE AND CONQUER ( TOPOLOGICAL SORT )

    // Mencari Course dengan Busur yang masuk adalah 0
    A1 = First(L); // Akses Elemen pertama pada CourseList
    while (A1 != Nil){ // Akan berhenti sampai A1 == Nil
        if (Masuk(A1) == 0 && !Visited(A1)){
            DecreaseNConquer(A1,1); // Akan diakses semua course yang tidak
memiliki prasyarat dan belum dikunjungi.
        }
        A1 = Next(A1); // Mengakses CourseList selanjutnya
    }
```

```
    // DESTRUKTOR (DEALOKASI SEMUA ALOCATED DATA)
    A1 = First(L);
    P2 = Course(A1);
    while (A1 != Nil)
    {
        A3 = Requisites(A1) ;
        while (A3 != Nil){
            A2 = A3;
            A3 = Requisites(A3);
            DealokasiListCourse(&A2);
        }

        while (P2 != Nil){
            P3 = P2 ;
            P2 = Next(P2);
            DealokasiCourse(&P3);
        }
        A2 = A1 ;
        A1 = Next(A1);
        if (A1 !=Nil)
        {
            P2   = Course(A1);
        }
        DealokasiListCourse(&A2);
    }
```

```
    // Menutup file yang telah dibuka dengan fopen
    fclose(inputFile);
    return 0 ;
}
```

```
// >>>>>>>>>>>  FUNGSI DECREASE AND CONQUER  <<<<<<<<<<<

/*  KONSEP NYA SAMA SEPERTI SPEK.
    DecreaseNConquer akan dipanggil seiring dengan penyempitan lingkup List yang
dikunjungi
*/

void DecreaseNConquer (address1 A1, int i){
    Visited(A1) = true ; // Set bahwa CourseList yang sedang diakses (A1) telah
dikunjungi
    if (Requisites(A1) == Nil ) { // Jika telah menemukan Simpul Ujung
        printf("Semester %d : ",i);
        PrintCourse(Course(A1));
    }
    else{
        int count ;
        address1 A2 ;
        printf("Semester %d : ",i); // Tampilkan terlebih dahulu Semester dari A1
        PrintCourse((Course(A1)));
        A2 = Requisites(A1);
        while (A2 != Nil) // Akses semua node yang ditunjuk oleh node A1
        {
            Masuk(Next(A2))-- ; // Mengurangi jumlah simpul masuk dari masing-
masing node yang ditunjuk A1
            A2 = Requisites(A2); // Akses node yang ditunjuk selanjutnya
        }
        i++ ; // Mengincrement i sebagai indeks semester
        A2 = Requisites(A1);
        while (A2 != Nil) // Akses semua node yang ditunjuk oleh node A1
        {
            if (Masuk(Next(A2)) == 0 && !Visited(Next(A2)))
            {
                DecreaseNConquer(Next(A2),i);   //  Jika jumlah simpul masuk dari
node yang diakses saat ini tidak sama dengan nol dan simpul tersebut belum
dikunjungi),
                                                //  Maka akan rekursif ke simpul
tersebut.
            }
            A2 = Requisites(A2); // Akses simpul yang ditunjuk selanjutnya
        }
    }
}
```

13519216-ADT.h

```c
#ifndef _DATA_
#define _DATA_

#include "stddef.h"
#include "stdlib.h"
#include "stdio.h"
#include "13519216-boolean.h"

#define Nil NULL
```

```c
typedef struct tCOURSE * address2 ;
typedef struct tCourseList * address1 ;
typedef struct tCOURSE
{
    char nama[50] ;
    int length ;
    address2 next ;
} COURSE;

typedef struct tCourseList
{
    int masuk ;
    boolean visitedNode;
    address2 course ;
    address1 requisites;
    address1 next ;
}CourseList;

typedef struct {
    address1 First;
} List;

#define Course(P) (P)->course
#define Next(P) (P)->next
#define Nama(P) (P)->nama
#define panjangKata(P) (P)->length
#define First(L) ((L).First)
#define Requisites(P) ((P)->requisites)
#define Masuk(P) ((P)->masuk)
#define Visited(P) ((P)->visitedNode)
```

```c
/****************** Manajemen Memori ******************/
address1 AlokasiListCourse (address2 theCourse);
address2 AlokasiCourse (char X[], int panjang);
void DealokasiListCourse (address1 *P);
void DealokasiCourse (address2 *P);
void PrintCourse(address2 P);
boolean Compare2Courses(address2 P,address2 P2);
```

```c
#endif
```

13519216-ADT.c

```c
#include "13519216-ADT.h"

// ALOKASI CourseList dan assign nilainya dengan COURSE yang telah dialokasi
address1 AlokasiListCourse (address2 theCourse){
    address1 P1 = (address1) malloc (sizeof(CourseList));
    if (P1 != Nil){
        Course(P1) = theCourse;
        Next(P1) = Nil;
    }
    Requisites(P1) = Nil;
    Masuk(P1) = 0;
    Visited(P1) = false ;
    return P1 ;
};
```

```c
// Alokasi COURSE
address2 AlokasiCourse (char X[], int panjang){
    address2 P = (address2) malloc (sizeof(COURSE));
    if (P != Nil)
    {
        int i;
        panjangKata(P) = panjang ;
        for (i = 0; i < panjang; i++)
        {
            Nama(P)[i] = X[i];
        }
        Next(P) = Nil ;
    }
    return P ;
};
```

```c
// DEALOKASI COURSELIST
void DealokasiListCourse (address1 *P){
    Next(*P) = Nil ;
    Requisites(*P) = Nil ;
    Course(*P) = Nil;
    free(*P);
};

// DEALOKASI COURSE
void DealokasiCourse (address2 *P){
    Next(*P) = Nil ;
    free(*P);
};
```

```
// MENAMPILKAN KE LAYAR nama dari COURSE P
void PrintCourse(address2 P){
    for (int i = 0; i < panjangKata(P); i++)
    {
        printf("%c",Nama(P)[i]);
    }
    printf("\n");
};

// Membandingkan nama dari COURSE P1 dan P2
boolean Compare2Courses(address2 P1,address2 P2){
    boolean sama = true;
    if (panjangKata(P1) == panjangKata(P2)){
        int i = 0 ;
        while (i < panjangKata(P1) && sama){
            if (Nama(P1)[i] == Nama(P2)[i]){
                i++;
            }
            else{   sama = false; }
        }
    }
    else{
        sama = false ;
    }
    return sama ;
};
```

13519216-boolean.h

```
/* File : boolean.h */

#ifndef _BOOLEAN_H
#define _BOOLEAN_H

#define boolean unsigned char
#define true 1
#define false 0

#endif
```

2.4. SCREENSHOT HASIL EKSEKUSI PROGRAM

1. FILE :

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

HASIL :

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : test
Semester 1 : C3
Semester 2 : C1
Semester 3 : C4
Semester 4 : C2
Semester 5 : C5
```

2. FILE :

A, B, C, D, E.
B,C,F,G.
C,D,F.
D,F,E.
E.
F, E.
G,C.

HASIL:

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : 7
Semester 1 : E
Semester 2 : F
Semester 3 : D
Semester 4 : C
Semester 5 : G
Semester 6 : B
Semester 7 : A
```

3. FILE :

A,B,C,D.
B,C,D.
C,E,F,G.
D,C,E,G.
E,F,G.
F.
G,F.

HASIL:

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : 6
Semester 1 : F
Semester 2 : G
Semester 3 : E
Semester 4 : C
Semester 5 : D
Semester 6 : B
Semester 7 : A
```

4. FILE :

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

HASIL :

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : 3
Semester 1 : C5
Semester 2 : C4
Semester 3 : C3
Semester 4 : C2
Semester 5 : C1
```

5. FILE :

A, C, D.
D, B, E.
C, E, Q.
Q, D, B.
B.
E, B.

HASIL:

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : 4
Semester 1 : B
Semester 2 : E
Semester 3 : D
Semester 4 : Q
Semester 5 : C
Semester 6 : A
```

6. FILE :

```
Fisika, Matematika.
Matematika, Kimia, Agama, Seni.
Kimia, Olahraga.
Olahraga, Agama, Seni.
Agama, Biologi, Seni.
Seni, Biologi.
Biologi.
```

HASIL :

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : 5
Semester 1 : Biologi
Semester 2 : Seni
Semester 3 : Agama
Semester 4 : Olahraga
Semester 5 : Kimia
Semester 6 : Matematika
Semester 7 : Fisika
```

7. FILE :

```
Ekonomi, Matematika, Sosiologi.
Matematika, Kimia, Fisika.
Agama.
Kimia, Seni.
Seni.
Sejarah, Agama.
Fisika, Kimia.
Sosiologi, Sejarah, Agama.
```

HASIL :

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : test2
Semester 1 : Agama
Semester 2 : Sejarah
Semester 3 : Sosiologi
Semester 1 : Seni
Semester 2 : Kimia
Semester 3 : Fisika
Semester 4 : Matematika
Semester 5 : Ekonomi
```

8. FILE :

```
Math,Physics.
Chemistry,Physics.
Physics.
Art,Religion,entrepreneurship.
entrepreneurship,Math,Religion.
Religion,Math,Chemistry.
```

HASIL :

```
Masukkan nama file pada folder 'test' (tanpa ekstensi .txt) : test1
Semester 1 : Physics
Semester 2 : Math
Semester 2 : Chemistry
Semester 3 : Religion
Semester 4 : entrepreneurship
Semester 5 : Art
```

2.5. DRIVE ADDRESS OF PROGRAM CODE

https://drive.google.com/drive/folders/1epu0iPY7p_ttz_-JO-TDX8AhBXSbAKh6?usp=sharing

2.6. CEK LIST

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.	✓	

BAB III

PENUTUP

5.1 Kesimpulan

Dari tugas besar berjudul "Penyusunan Rencana Kuliah dengan Topological Sort (Penerapan Decrease and Conquer)", maka program yang saya buat dapat menyusun rencana pengambilan mata kuliah sesuai dengan spesifikasi program. Program penyusunan rencana kuliah ini juga benar untuk semua kasus.

5.2 Saran

Berikut ini adalah saran yang dapat saya berikan untuk tugas besar ini.

1. Sebaiknya dalam penyampaian spesifikasi tugas lebih diperjelas agar meminimumkan pertanyaan diluar hal yang terjadwal karena jika pun terdapat QnA, beberapa orang memungkinkan tidak mengaksesnya.

5.3 Refleksi

1. Dalam pengerjaan tugas sebaiknya tidak dikerjakan mendekati deadline pengumpulan tugas.
2. Dalam pengerjaan tugas sebaiknya dilakukan secara mencicil sehingga tidak membebani aktivitas perkuliahan lainnya.
3. Jika mengerjakan tugas dengan waktu sisa yang terbatas sebaiknya dapat belajar lebih cepat dan efisien.

REFERENSI

WilliamFiset. (2017). Topological Sort Algorithm | Graph Theory. Diakses pada 28 Februari 2021 dari https://www.youtube.com/watch?v=eL-KzMXSXXI

Informatika.stei.itb.ac.id. (2021). Tugas Kecil II IF2211 Strategi Algoritma. Diakses pada 28 Februari 2021 dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Kecil-2-(2021).pdf

Informatika.stei.itb.ac.id. (2021). Algoritma Decrease And Conquer. Diakses pada 28 Februari 2021 dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf

Wikipedia. 2021. Topological Sorting. Diakses pada 28 Februari 2021 dari https://en.wikipedia.org/wiki/Topological_sorting