Laporan Tugas Kecil 2
Mata Kuliah Strategi Algoritma
IF2211 2020/2021

Hughie Alghaniyyu Emiliano
13519217 K-04

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

ALGORITMA TOPOLOGICAL SORT

Algoritma *Topological Sort* adalah algoritma yang digunakan untuk mengurutkan banyak elemen dengan cara membandingkan dua elemen *a* dan *b* sehingga elemen *a* yang mengarah kepada elemen *b*, *a* harus berada sebelum *b*, posisi elemen *a* akan berada pada sebelah kiri elemen *b*. Algoritma *Decrease and Conquer* adalah algoritma yang digunakan untuk menyelesaikan masalah dengan cara membagi permasalahan tersebut menjadi beberapa sub-masalah yang lebih kecil kemudian didapatkan solusi dari permasalahan tersebut tanpa harus menggabungkan semua solusi setiap sub-persoalan. Jadi, keterkaitan algoritma *Topological Sort* dengan pendekatan algoritma *Decrease and Conquer* yaitu kedua algoritma ini menyelesaikan suatu permasalahan dengan cara menyelesaikan tiap sub-masalah yang ada sehingga didapatkan solusi untuk seluruh permasalahan.

SOURCE PROGRAM

Pada tugas ini, bahasa pemrograman yang saya gunakan adalah bahasa C.

a. boolean13519217.h

```c
/* Definisi type boolean */

#ifndef _BOOLEAN13519217_h
#define _BOOLEAN13519217_h

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

#endif
```

b. modul13519217.h

```c
#ifndef modul13519217_H
#define modul13519217_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define kode(P) (P)->kode
#define matkul(P) (P)->matkul
#define next(P) (P)->next
#define data(P) (P)->data
#define source(P) (P)->source
#define need(P) (P)->need
#define First(L) (L).First

// Address Mata Kuliah
typedef struct AdrMatkul *addressM;

// Address Prerequisite
typedef struct AdrPrereq *addressP;

// Struct Matakuliah
typedef struct AdrMatkul {
    int kode;
    char matkul[10];
    addressM next;
} ElmtMatkul;

// Struct Prerequisite
typedef struct AdrPrereq {
    int data;
    addressM source;
    addressM need;
    addressP next;
} ElmtPrereq;

// List Matakuliah
typedef struct {
    addressM First;
} Matakuliah;

// List Prerequisite
typedef struct {
    addressP First;
} Prerequisite;

// Alokasi Matakuliah
addressM AlokasiM (Matakuliah L, int X, char X1[]);
```

```c
// Alokasi Prerequistie
addressP AlokasiP (Matakuliah L, int X, char X1[], char X2[]);

// Dealokasi Matakuliah
void DealokasiM (addressM *P);

// Dealokasi Prerequisite
void DealokasiP (addressP *P);

// Search Kode
addressM SearchKode (Matakuliah L, int X);

// Search Matakuliah
addressM SearchMatkul (Matakuliah L, char X[]);

// Search Prerequisite
addressP SearchPrereq (Matakuliah L1, Prerequisite L2, char X[]);

// Add Matakuliah
void InsVLastM (Matakuliah *L, int X, char X1[]);

// Add Prereq
void InsVLastP (Matakuliah L1, Prerequisite *L2, int X, char X1[], char X2[]);
```

```c
// Fungsi untuk membaca mata kuliah dengan prerequisitenya
// Kemudian, memasukkannya ke dalam graf
void input(FILE *ptr, Matakuliah L1, Prerequisite L2);

// Menghapus matakuliah dari list
void DeleteM (Matakuliah L, char X[]);

// Menghapus keterangan prerequisite dari list
void DeleteP (Prerequisite L, char X[]);

// Mencari matakuliah yang tidak memiliki prerequisite
// atau
// Mencari matakuliah yang prerequisitenya sudah dipenuhi
addressP FindP (Matakuliah L1, Prerequisite L2);
#endif
```

c. modul13519217.c

```c
#include "modul13519217.h"
#include "boolean13519217.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Alokasi Matakuliah
addressM AlokasiM (Matakuliah L, int X, char X1[]) {
    addressM P;
    P = SearchKode(L, X);
    // Jika matakuliah sudah pernah dialokasi
    if (P != NULL) {
        return P;
    }
    // Jika mata kuliah belum dialokasi
    else {
        // Membuat alokasi baru
        P = (addressM) malloc (sizeof (X));
        if (P == NULL) {
            return P;
        }
        else {
            kode(P) = X;
```

```
            strcpy(matkul(P), X1);
            next(P) = NULL;
            return P;
        }
    }
}

// Alokasi Prerequistie
addressP AlokasiP (Matakuliah L, int X, char X1[], char X2[]) {
    addressP P;
    P = (addressP) malloc (sizeof (X));
    if (P == NULL) {
        return P;
    }
    else {
        data(P) = X;
        source(P) = SearchMatkul(L, X1);
        need(P) = SearchMatkul(L, X2);
        next(P) = NULL;
        return P;
    }
}

// Dealokasi Matakuliah
void DealokasiM (addressM *P) {
    next(*P) = NULL;
    free (*P);
}

// Dealokasi Prerequisite
void DealokasiP (addressP *P) {
    next(*P) = NULL;
    free (*P);
}

// Search Kode Matakuliah
addressM SearchKode (Matakuliah L, int X) {
    addressM P;
    P = First(L);
    while (P != NULL) {
        if (kode(P) == X) {
            return P;
        }
        else {
            P = next(P);
        }
    }
    return P;
}

// Search Matakuliah
addressM SearchMatkul (Matakuliah L, char X[]) {
    addressM P;
    P = First(L);
    while (P != NULL) {
        if ((strcmp(matkul(P), X)) == 0) {
            return P;
        }
        else {
            P = next(P);
        }
    }
    return P;
}

// Search Prerequisite
addressP SearchPrereq (Matakuliah L1, Prerequisite L2, char X[]) {
    addressP P, Pc;
    addressM M;
    M = SearchMatkul(L1, X);
    P = First(L2);
    while (P != NULL) {
```

```
        if (M == need(P)) {
            break;
        }
    }
    return Pc;
}

// Add Matakuliah
void InsVLastM (Matakuliah * L, int X, char X1[]) {
    // Cek apakah sudah ada di dalam list
    addressM P = SearchKode(*L, X);
    // Jika belum ada
    if (P == NULL) {
        // Membuat alokasi baru untuk matakuliah X1
        addressM P = AlokasiM(*L, X, X1);
        if (P != NULL) {
            // Jika list masih kosong
            if (First(*L) == NULL) {
                First(*L) = P;
            }
            // Jika list tidak kosong
            else {
                addressM Pc = First(*L);
                while (next(Pc) != NULL) {
                    Pc = next(Pc);
                }
                next(Pc) = P;
            }
        }
    }
}

// Add Prereq
void InsVLastP (Matakuliah L1, Prerequisite * L2, int X, char X1[], char X2[]) {
    // Membuat alokasi baru untuk keterangan matakuliah X1 dengan prereq X2
    addressP P = AlokasiP(L1, X, X1, X2);
    if (P != NULL) {
        // Jika list prereq masih kosong
        if (First(*L2) == NULL) {
            First(*L2) = P;
        }
        // Jika list prereq tidak kosong
        else {
            addressP Pc = First(*L2);
            while (next(Pc) != NULL) {
                Pc = next(Pc);
            }
            next(Pc) = P;
        }
    }
}

// Fungsi untuk membaca mata kuliah dengan prerequisitenya
// Kemudian, memasukkannya ke dalam list
void input(FILE *ptr, Matakuliah L1, Prerequisite L2) {
    // Kamus
    char CC, sisa[255], kosong[10], hasilbaca[10], matkulsource[10];
    int i, data, kode;
    boolean mksource;

    // Keterangan data matakuliah dengan prerequisitenya pada list
    data = 1;
    // Keterangan kode matakuliah pada list
    kode = 1;
    // Membaca file
    CC = fgetc(ptr);
    // Iterasi hingga 'End of File'
    while (CC != EOF) {
        i = 0;
        // Menggabungkan karakter-karakter yang dibaca
        // Menjadi nama suatu matakuliah
        hasilbaca[i] = CC; i++;
```

```
        CC = fgetc(ptr);
        // Iterasi hingga 'End of Line'
        // Ket : Titik menjadi mark setiap baris
        while (CC != '.') {
            // Mereset nama matakuliah utama setelah disimpan pada list
            strcpy(matkulsource, kosong);
            // Menandakan matakuliah utama belum didapatkan
            mksource = false;
            // Proses mengambil nama matakuliah utama
            // Iterasi hingga matakuliah utama sudah didapatkan
            while (!mksource) {
                hasilbaca[i] = CC; i++;
                CC = fgetc(ptr);
                // Iterasi hingga didapatkan nama mata kuliah utama
                // Ket : Koma menjadi mark pergantian nama mata kuliah
                if (CC == ',') {
                    // Men-skip karakter koma(',') dan spasi(' ')
                    CC = fgetc(ptr);
                    CC = fgetc(ptr);
                    // Memindahkan nama matakuliah yang berhasil dibaca
                    // ke variabel matakuliah utama
                    strcpy(matkulsource,hasilbaca);
                    // Memasukkan nama matakuliah utama ke dalam list matakuliah
                    InsVLastM(&L1, kode, matkulsource); kode++;
                    // Mengosongkan variabel hasil pembacaan nama matakuliah
                    strcpy(hasilbaca, kosong);
                    // Matakuliah utama sudah didapatkan
                    mksource = true;
                    i = 0;
                }
                // Kasus matakuliah tanpa prerequisite
                if (CC == '.') {
                    // Untuk pindah ke nextline
                    fgets(sisa, 255, ptr);
                    // Membaca karakter nama matakuliah selanjutnya
                    CC = fgetc(ptr);
                    // Memindahkan nama matakuliah yang berhasil dibaca
                    // ke variabel matakuliah utama
                    strcpy(matkulsource,hasilbaca);
                    // Memasukkan nama matakuliah utama ke dalam list matakuliah
                    InsVLastM(&L1, kode, matkulsource); kode++;
                    // Memasukkan keterangan matakuliah tanpa prerequisite ke dalam list pre
                    InsVLastP(L1, &L2, data, matkulsource, NULL); data++;
                    // Mengosongkan variabel hasil pembacaan nama mata kuliah
                    strcpy(hasilbaca, kosong);
                    // Mengosongkan variabel nama matakuliah utama yang didapatkan
                    strcpy(matkulsource, kosong);
                    i = 0;
                }
            }
            // Proses pemngambilan kebutuhan prerequisite matakuliah utama
            while ((CC != ',') && (mksource)) {
                // Pembacaan nama matakuliah prerequisite
                hasilbaca[i] = CC; i++;
                CC = fgetc(ptr);
                // Prerequisite sudah didapatkan
                // Masih ada prerequisite
                if (CC == ',') {
                    // Men-skip karakter koma(',') dan spasi(' ')
                    CC = fgetc(ptr);
                    CC = fgetc(ptr);
                    // Memasukan nama matakuliah prerequsiite ke dalam list matakuliah
                    InsVLastM(&L1, kode, hasilbaca); kode++;
                    // Memasukkan keterangan nama matakuliah utama dengan prerequisitenya
                    // ke dalam list prerequisite
                    InsVLastP(L1, &L2, data, matkulsource, hasilbaca); data++;
                    // Mengosongkan variabel hasil pembacaan nama matakuliah
                    strcpy(hasilbaca, kosong);
                    i = 0;
                }
                // Prerequisite sudah didapatkan
                // Tidak ada prerequisite yang harus dibaca lagi
```

```
                    if (CC == '.') {
                        // Untuk ke next line
                        fgets(sisa, 255, ptr);
                        // Membaca karakter selanjutnya
                        CC = fgetc(ptr);
                        // Memasukan nama matakuliah prerequsiite ke dalam list matakuliah
                        InsVLastM(&L1, kode, hasilbaca); kode++;
                        // Memasukkan keterangan nama matakuliah utama dengan prerequisitenya
                        // ke dalam list prerequisite
                        InsVLastP(L1, &L2, data, matkulsource, hasilbaca); data++;
                        // Mengosongkan variabel hasil pembacaan nama matakuliah
                        strcpy(hasilbaca, kosong);
                        // Mengosongkan variabel matakuliah utama
                        strcpy(matkulsource,kosong);
                        i = 0;
                        // Matakuliah utama perlu dicari kembali
                        mksource = false;
                    }
                }
            }
        }
    }

    // Menghapus matakuliah dari list
    void DeleteM (Matakuliah L, char X[]) {
        addressM P, Pc;
        P = First(L);
        if (strcmp(matkul(P), X) == 0) {
            First(L) = next(P);
            DealokasiM(&P);
        }
        else {
            while (strcmp(matkul(P), X) != 0) {
                Pc = P;
                P = next(P);
            }
            next(Pc) = next(P);
            DealokasiM(&P);
        }
    }

    // Menghapus keterangan prerequisite dari list
    void DeleteP (Prerequisite L, char X[]) {
        addressP P, Pc;
        char subject[10];
        P = First(L);
        if (strcmp(matkul(source(P)), X) == 0) {
            First(L) = next(P);
            DealokasiP(&P);
        }
        Pc = P;
        P = next(P);
        while (P != NULL) {
            if (strcmp(matkul(source(P)),X) == 0) {
                next(Pc) = next(P);
                DealokasiP(&P);
            }
            Pc = P;
            P = next(P);
        }
    }

    // Mencari matakuliah yang tidak memiliki prerequisite
    // atau
    // Mencari matakuliah yang prerequisitenya sudah dipenuhi
    addressP FindP (Matakuliah L1, Prerequisite L2) {
        return SearchPrereq(L1, L2, NULL);
    }
```

d. main13519217.c

```c
#include "modul13519217.h"
#include "boolean13519217.h"

// KAMUS
Matakuliah L1;
Prerequisite L2;
addressM M;
addressP P;
int semester = 0;

int main() {
    // Directory file
    char* file = "..\\test\\test1.txt";

    // Membaca file
    FILE* pFile = fopen(file, "r");

    // Jika file tidak ditemukan
    if (pFile == NULL) {
        printf("Error : File tidak dapat ditemukan!\n");
        // Keluar dari program
        exit(1);
    }
    // Memasukkan semua data input kedalam list matakuliah dan prerequisite
    input(pFile, L1, L2);

    // Memulai pencarian mata kuliah yang tepat
    // untuk setiap semester
    while (semester <= 8) {
        semester += 1;
        // Jika semua matakuliah sudah diambil
        if (First(L1) == NULL) {
            exit(1);
        }
        // Matakuliah masih tersedia
        else {
            // Mencari matakuliah yang tepat
            P = FindP(L1, L2);
            printf("Semester %d : ", semester);
            while (P != NULL) {
                // Print nama matakuliah yang tepat
                printf("%s ", semester, matkul(source(P)));
                DeleteP(L2, matkul(source(P)));
            }
            DeleteM(L1, matkul(source(P)));
            // Print next line
            printf("\n");
        }
    }
}
```

EKSPERIMEN

Program yang saya buat gagal untuk menemukan solusi dari permasalahan. Ketika dijalankan, program akan langsung *exit* tanpa memberikan *output*. Ketika dilihat dari debugging, program mendapatkan masalah ketika membaca input baris kedua, pada pembacaan matakuliah utama. Program berhasil membaca bahwa terdapat matakuliah utama dan berhasil mendapatkan nama matakuliah tersebut, tetapi program bermasalah ketika memasukkan nama matakuliah tersebut ke dalam list matakuliah. Di dalam proses tersebut, program bermasalah ketika melakukan alokasi *address* untuk matakuliah tersebut sehingga program langsung *exit* ketika melakukan alokasi *address* pada list matakuliah untuk matakuliah tersebut.

REPOSITORY

https://github.com/HughieAlghani/Tucil2Stima2021

Checklist

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		√