

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

```
1  /* Definisi type boolean */
2
3  #ifndef _BOOLEAN13519217_h
4  #define _BOOLEAN13519217_h
5
6  #define boolean unsigned char
7  #define true 1
8  #define false 0
9
10 #endif
```

b. modul13519217.h

```
1  #ifndef modul13519217_H
2  #define modul13519217_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7
8  #define kode(P) (P)->kode
9  #define matkul(P) (P)->matkul
10 #define next(P) (P)->next
11 #define data(P) (P)->data
12 #define source(P) (P)->source
13 #define need(P) (P)->need
14 #define First(L) (L).First
15
16 // Address Mata Kuliah
17 typedef struct AdrMatkul *addressM;
18
19 // Address Prerequisite
20 typedef struct AdrPrereq *addressP;
21
22 // Struct Matakuliah
23 typedef struct AdrMatkul {
24     int kode;
```

```
25     char matkul[10];
26     addressM next;
27 } ElmtMatkul;
28
29 // Struct Prerequisite
30 typedef struct AdrPrereq {
31     int data;
32     addressM source;
33     addressM need;
34     addressP next;
35 } ElmtPrereq;
36
37 // List Matakuliah
38 typedef struct {
39     addressM First;
40 } Matakuliah;
41
42 // List Prerequisite
43 typedef struct {
44     addressP First;
45 } Prerequisite;
46
47 // Alokasi Matakuliah
48 addressM AlokasiM (Matakuliah L, int X, char X1[]);
```

```

50 // Alokasi Prerequisite
51 addressP AlokasiP (Matakuliah L, int X, char X1[], char X2[]);
52
53 // Dealokasi Matakuliah
54 void DealokasiM (addressM *P);
55
56 // Dealokasi Prerequisite
57 void DealokasiP (addressP *P);
58
59 // Search Kode
60 addressM SearchKode (Matakuliah L, int X);
61
62 // Search Matakuliah
63 addressM SearchMatkul (Matakuliah L, char X[]);
64
65 // Search Prerequisite
66 addressP SearchPrereq (Matakuliah L1, Prerequisite L2, char X[]);
67
68 // Add Matakuliah
69 void InsVLastM (Matakuliah *L, int X, char X1[]);
70
71 // Add Prereq
72 void InsVLastP (Matakuliah L1, Prerequisite *L2, int X, char X1[], char X2[]);
73
74 // Fungsi untuk membaca mata kuliah dengan prerequisitenya
75 // Kemudian, memasukkannya ke dalam graf
76 void input(FILE *ptr, Matakuliah L1, Prerequisite L2);
77
78 // Menghapus matakuliah dari list
79 void DeleteM (Matakuliah L, char X[]);
80
81 // Menghapus keterangan prerequisite dari list
82 void DeleteP (Prerequisite L, char X[]);
83
84 // Mencari matakuliah yang tidak memiliki prerequisite
85 // atau
86 // Mencari matakuliah yang prerequisitenya sudah dipenuhi
87 addressP FindP (Matakuliah L1, Prerequisite L2);
88 #endif

```

c. modul13519217.c

```

1  #include "modul13519217.h"
2  #include "boolean13519217.h"
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7
8  // Alokasi Matakuliah
9  addressM AlokasiM (Matakuliah L, int X, char X1[]) {
10     addressM P;
11     P = SearchKode(L, X);
12     // Jika matakuliah sudah pernah dialokasi
13     if (P != NULL) {
14         return P;
15     }
16     // Jika mata kuliah belum dialokasi
17     else {
18         // Membuat alokasi baru
19         P = (addressM) malloc (sizeof (X));
20         if (P == NULL) {
21             return P;
22         }
23         else {
24             kode(P) = X;

```

```

25         strcpy(matkul(P), X1);
26         next(P) = NULL;
27         return P;
28     }
29 }
30 }
31
32 // Alokasi Prerequisite
33 addressP AlokasiP (Matakuliah L, int X, char X1[], char X2[]) {
34     addressP P;
35     P = (addressP) malloc (sizeof (X));
36     if (P == NULL) {
37         return P;
38     }
39     else {
40         data(P) = X;
41         source(P) = SearchMatkul(L, X1);
42         need(P) = SearchMatkul(L, X2);
43         next(P) = NULL;
44         return P;
45     }
46 }
47
48 // Dealokasi Matakuliah

```

```

49 void DealokasiM (addressM *P) {
50     next(*P) = NULL;
51     free (*P);
52 }
53
54 // Dealokasi Prerequisite
55 void DealokasiP (addressP *P) {
56     next(*P) = NULL;
57     free (*P);
58 }
59
60 // Search Kode Matakuliah
61 addressM SearchKode (Matakuliah L, int X) {
62     addressM P;
63     P = First(L);
64     while (P != NULL) {
65         if (kode(P) == X) {
66             return P;
67         }
68         else {
69             P = next(P);
70         }
71     }
72     return P;

```

```

73 }
74
75 // Search Matakuliah
76 addressM SearchMatkul (Matakuliah L, char X[]) {
77     addressM P;
78     P = First(L);
79     while (P != NULL) {
80         if ((strcmp(matkul(P), X)) == 0) {
81             return P;
82         }
83         else {
84             P = next(P);
85         }
86     }
87     return P;
88 }
89
90 // Search Prerequisite
91 addressP SearchPrereq (Matakuliah L1, Prerequisite L2, char X[]) {
92     addressP P, Pc;
93     addressM M;
94     M = SearchMatkul(L1, X);
95     P = First(L2);
96     while (P != NULL) {

```

```

97         if (M == need(P)) {
98             break;
99         }
100     }
101     return Pc;
102 }
103
104 // Add Matakuliah
105 void InsVlastM (Matakuliah * L, int X, char X1[]) {
106     // Cek apakah sudah ada di dalam list
107     addressM P = SearchKode(*L, X);
108     // Jika belum ada
109     if (P == NULL) {
110         // Membuat alokasi baru untuk matakuliah X1
111         addressM P = AlokasiM(*L, X, X1);
112         if (P != NULL) {
113             // Jika list masih kosong
114             if (First(*L) == NULL) {
115                 First(*L) = P;
116             }
117             // Jika list tidak kosong
118             else {
119                 addressM Pc = First(*L);
120                 while (next(Pc) != NULL) {

```

```

121                     Pc = next(Pc);
122                 }
123                 next(Pc) = P;
124             }
125         }
126     }
127 }
128
129 // Add Prereq
130 void InsVlastP (Matakuliah L1, Prerequisite * L2, int X, char X1[], char X2[]) {
131     // Membuat alokasi baru untuk keterangan matakuliah X1 dengan prereq X2
132     addressP P = AlokasiP(L1, X, X1, X2);
133     if (P != NULL) {
134         // Jika list prereq masih kosong
135         if (First(*L2) == NULL) {
136             First(*L2) = P;
137         }
138         // Jika list prereq tidak kosong
139         else {
140             addressP Pc = First(*L2);
141             while (next(Pc) != NULL) {
142                 Pc = next(Pc);
143             }
144             next(Pc) = P;

```

```

145         }
146     }
147 }
148
149 // Fungsi untuk membaca mata kuliah dengan prerequisitenya
150 // Kemudian, memasukkannya ke dalam list
151 void input(FILE *ptr, Matakuliah L1, Prerequisite L2) {
152     // Kamus
153     char CC, sisa[255], kosong[10], hasilbaca[10], matkulsource[10];
154     int i, data, kode;
155     boolean mksource;
156
157     // Keterangan data matakuliah dengan prerequisitenya pada list
158     data = 1;
159     // Keterangan kode matakuliah pada list
160     kode = 1;
161     // Membaca file
162     CC = fgetc(ptr);
163     // Iterasi hingga 'End of File'
164     while (CC != EOF) {
165         i = 0;
166         // Menggabungkan karakter-karakter yang dibaca
167         // Menjadi nama suatu matakuliah
168         hasilbaca[i] = CC; i++;

```

```

169     CC = fgetc(ptr);
170     // Iterasi hingga 'End of Line'
171     // Ket : Titik menjadi mark setiap baris
172     while (CC != '.') {
173         // Mereset nama matakuliah utama setelah disimpan pada list
174         strcpy(matkulsource, kosong);
175         // Menandakan matakuliah utama belum didapatkan
176         mksource = false;
177         // Proses mengambil nama matakuliah utama
178         // Iterasi hingga matakuliah utama sudah didapatkan
179         while (!mksource) {
180             hasilbaca[i] = CC; i++;
181             CC = fgetc(ptr);
182             // Iterasi hingga didapatkan nama mata kuliah utama
183             // Ket : Koma menjadi mark pergantian nama mata kuliah
184             if (CC == ',') {
185                 // Men-skip karakter koma(',') dan spasi(' ')
186                 CC = fgetc(ptr);
187                 CC = fgetc(ptr);
188                 // Memindahkan nama matakuliah yang berhasil dibaca
189                 // ke variabel matakuliah utama
190                 strcpy(matkulsource, hasilbaca);
191                 // Memasukkan nama matakuliah utama ke dalam list matakuliah
192                 InsVlastM(&L1, kode, matkulsource); kode++;
193
194                 // Mengosongkan variabel hasil pembacaan nama matakuliah
195                 strcpy(hasilbaca, kosong);
196                 // Matakuliah utama sudah didapatkan
197                 mksource = true;
198                 i = 0;
199             }
200             // Kasus matakuliah tanpa prerequisite
201             if (CC == '.') {
202                 // Untuk pindah ke nextline
203                 fgetc(sisa, 255, ptr);
204                 // Membaca karakter nama matakuliah selanjutnya
205                 CC = fgetc(ptr);
206                 // Memindahkan nama matakuliah yang berhasil dibaca
207                 // ke variabel matakuliah utama
208                 strcpy(matkulsource, hasilbaca);
209                 // Memasukkan nama matakuliah utama ke dalam list matakuliah
210                 InsVlastM(&L1, kode, matkulsource); kode++;
211                 // Memasukkan keterangan matakuliah tanpa prerequisite ke dalam list pre
212                 InsVlastP(L1, &L2, data, matkulsource, NULL); data++;
213                 // Mengosongkan variabel hasil pembacaan nama mata kuliah
214                 strcpy(hasilbaca, kosong);
215                 // Mengosongkan variabel nama matakuliah utama yang didapatkan
216                 strcpy(matkulsource, kosong);
217                 i = 0;
218             }
219             // Proses pengambilan kebutuhan prerequisite matakuliah utama
220             while ((CC != ',') && (mksource)) {
221                 // Pembacaan nama matakuliah prerequisite
222                 hasilbaca[i] = CC; i++;
223                 CC = fgetc(ptr);
224                 // Prerequisite sudah didapatkan
225                 // Masih ada prerequisite
226                 if (CC == ',') {
227                     // Men-skip karakter koma(',') dan spasi(' ')
228                     CC = fgetc(ptr);
229                     CC = fgetc(ptr);
230                     // Memasukan nama matakuliah prerequisite ke dalam list matakuliah
231                     InsVlastM(&L1, kode, hasilbaca); kode++;
232                     // Memasukkan keterangan nama matakuliah utama dengan prerequisite
233                     // ke dalam list prerequisite
234                     InsVlastP(L1, &L2, data, matkulsource, hasilbaca); data++;
235                     // Mengosongkan variabel hasil pembacaan nama matakuliah
236                     strcpy(hasilbaca, kosong);
237                     i = 0;
238                 }
239                 // Prerequisite sudah didapatkan
240                 // Tidak ada prerequisite yang harus dibaca lagi

```

```

241         if (CC == '.') {
242             // Untuk ke next line
243             fgets(sisa, 255, ptr);
244             // Membaca karakter selanjutnya
245             CC = fgetc(ptr);
246             // Memasukan nama matakuliah prerequisiite ke dalam list matakuliah
247             InsVLastM(&L1, kode, hasilbaca); kode++;
248             // Memasukkan keterangan nama matakuliah utama dengan prerequisitenya
249             // ke dalam list prerequisite
250             InsVLastP(L1, &L2, data, matkulsource, hasilbaca); data++;
251             // Mengosongkan variabel hasil pembacaan nama matakuliah
252             strcpy(hasilbaca, kosong);
253             // Mengosongkan variabel matakuliah utama
254             strcpy(matkulsource, kosong);
255             i = 0;
256             // Matakuliah utama perlu dicari kembali
257             mksource = false;
258         }
259     }
260 }
261 }
262 }
263
264 // Menghapus matakuliah dari list

```

```

265 void DeleteM (Matakuliah L, char X[]) {
266     addressM P, Pc;
267     P = First(L);
268     if (strcmp(matkul(P), X) == 0) {
269         First(L) = next(P);
270         DealokasiM(&P);
271     }
272     else {
273         while (strcmp(matkul(P), X) != 0) {
274             Pc = P;
275             P = next(P);
276         }
277         next(Pc) = next(P);
278         DealokasiM(&P);
279     }
280 }
281
282 // Menghapus keterangan prerequisite dari list
283 void DeleteP (Prerequisite L, char X[]) {
284     addressP P, Pc;
285     char subject[10];
286     P = First(L);
287     if (strcmp(matkul(source(P)), X) == 0) {
288         First(L) = next(P);

```

```

289         DealokasiP(&P);
290     }
291     Pc = P;
292     P = next(P);
293     while (P != NULL) {
294         if (strcmp(matkul(source(P)), X) == 0) {
295             next(Pc) = next(P);
296             DealokasiP(&P);
297         }
298         Pc = P;
299         P = next(P);
300     }
301 }
302
303 // Mencari matakuliah yang tidak memiliki prerequisite
304 // atau
305 // Mencari matakuliah yang prerequisitenya sudah dipenuhi
306 addressP FindP (Matakuliah L1, Prerequisite L2) {
307     return SearchPrereq(L1, L2, NULL);
308 }
309
310

```


d. main13519217.c

```
1  #include "modul13519217.h"
2  #include "boolean13519217.h"
3
4  // KAMUS
5  Matakuliah L1;
6  Prerequisite L2;
7  addressM M;
8  addressP P;
9  int semester = 0;
10
11 int main() {
12     // Directory file
13     char* file = "..\\test\\test1.txt";
14
15     // Membaca file
16     FILE* pFile = fopen(file, "r");
17
18     // Jika file tidak ditemukan
19     if (pFile == NULL) {
20         printf("Error : File tidak dapat ditemukan!\n");
21         // Keluar dari program
22         exit(1);
23     }
24     // Memasukkan semua data input kedalam list matakuliah dan prerequisite
25     input(pFile, L1, L2);
26
27     // Memulai pencarian mata kuliah yang tepat
28     // untuk setiap semester
29     while (semester <= 8) {
30         semester += 1;
31
32         // Jika semua matakuliah sudah diambil
33         if (First(L1) == NULL) {
34             exit(1);
35         }
36         // Matakuliah masih tersedia
37         else {
38             // Mencari matakuliah yang tepat
39             P = FindP(L1, L2);
40             printf("Semester %d : ", semester);
41             while (P != NULL) {
42                 // Print nama matakuliah yang tepat
43                 printf("%s ", semester, matkul(source(P)));
44                 DeleteP(L2, matkul(source(P)));
45             }
46             DeleteM(L1, matkul(source(P)));
47             // Print next line
48             printf("\n");
49         }
50     }
```

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 <i>running</i>		√
3. Program dapat menerima berkas input dan menuliskan output		√
4. Luaran sudah benar untuk semua kasus input		√