Penerapan Algoritma BFS dan DFS untuk Penjadwalan Rencana Studi

Muhamad Arif Adiputra (13516114)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hotaru@itb.ac.id

***Abstraksi* — Makalah ini berisi tentang penjelasan awal mengenai definisi algoritma, algoritma BFS dan DFS, dan penerapan algoritma BFS dan DFS untuk mencari banyaknya semester yang dapat diambil seminimal mungkin untuk beberapa mata kuliah dengan prasyarat-prasyarat mata kuliah tersebut sudah dipenuhi**

***Kata Kunci* — Algoritma, Algoritma BFS (Breadth First Search), DFS (Depth First Search)**

I. PENDAHULUAN

Dalam matematika dan ilmu komputer, algoritma adalah prosedur langkah-demi-langkah untuk penghitungan. Algoritma digunakan untuk penghitungan, pemrosesan data, dan penalaran otomatis.

Algoritma adalah metode efektif diekspresikan sebagai rangkaian terbatas dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah fungsi. Dimulai dari sebuah kondisi awal dan input awal (mungkin kosong), instruksi-instruksi tersebut menjelaskan sebuah komputasi yang, bila dieksekusi, diproses lewat sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, yang pada akhirnya menghasilkan "keluaran" dan berhenti di kondisi akhir. Transisi dari satu kondisi ke kondisi selanjutnya tidak harus deterministik; beberapa algoritme, dikenal dengan algoritme pengacakan, menggunakan masukan acak.

Walaupun algorism-nya al-Khawarizmi dirujuk sebagai aturan-aturan melakukan aritmetika menggunakan bilangan Hindu-Arab dan solusi sistematis dan persamaan kuadrat, sebagian formalisasi yang nantinya menjadi algoritme modern dimulai dengan usaha untuk memecahkan permasalahan keputusan (Entscheidungsproblem) yang diajukan oleh David Hilbert pada tahun 1928. Formalisasi selanjutnya dilihat sebagai usaha untuk menentukan "penghitungan efektif" atau "metode efektif" formalisasi tersebut mengikutkan Godel-Herbrand-Kleene fungsi rekursif-nya Kurt Godel - Jacques Herbrand - Stephen Cole Kleene pada tahun 1930, 1934, dan 1935, kalkulus lambda-nya Alonzo Church pada tahun 1936, "Formulasi 1"-nya Emil Post pada tahun 1936, dan Mesin Turing-nya Alan Turing pada tahun 1936-7 dan 1939. Dari definisi formal dari algoritme di atas, berkaitan dengan konsep intuituf, masih tetap ada masalah yang menantang.

Dalam makalah ini, algoritma BFS (Breadth First Search) dan DFS (Depth First Search) akan digunakan untuk menjadwalkan rencana studi para mahasiswa yang sedang mengikuti perkuliahan. Penggunaan algoritma penting karena umumnya para mahasiswa tidak teliti dalam memilih mata kuliah dengan prasyarat-prasyarat yang harus dipenuhi untuk mengambil mata kuliah tersebut

Dalam pemilihan mata kuliah, para mahasiswa dalam beberapa program studi atau fakultas tertentu sudah diberikan panduan mata kuliah yang wajib diambil oleh para mahasiswa tersebut agar dapat lulus. Banyak dari para mahasiswa tersebut terkadang tidak memerhatikan prasyarat yang terdapat dalam setiap mata kuliah sehingga menjadikan mata kuliah yang akan diambil oleh mahasiswa tersebut tidak dapat diambil dan menjadi beban mahasiswa tersebut dalam perencanaan rencana studi mahasiswa tersebut. Disini, algoritma BFS dan DFS sangat berguna untuk mencari tahu apakah rencana studi para mahasiswa tersebut sudah baik atau tidak dilihat dari prasyarat dari mata kuliahnya telah diambil di perkuliahan sebelumnya.

Tujuan utama dari penerapan algoritma BFS dan DFS ini adalah menjadikan para mahasiswa dapat melihat rencana studi mereka apakah sudah baik (sudah memenuhi prasyaratnya) untuk seluruh mata kuliah yang diambilnya dengan susunannya atau tidak. Dengan pemakaian algoritma ini juga para mahasiswa dapat melihat jumlah minimum semester yang dapat dia ambil pada saat perkuliahan dengan batas SKS yang dapat diambilnya.

II. ALGORITMA BFS DAN DFS

1. BFS (BREADTH FIRST SEARCH)

Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum

dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi , demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada arah d dikunjungi lebih dahulu sebelum simpul-simpul pada arah d+1.

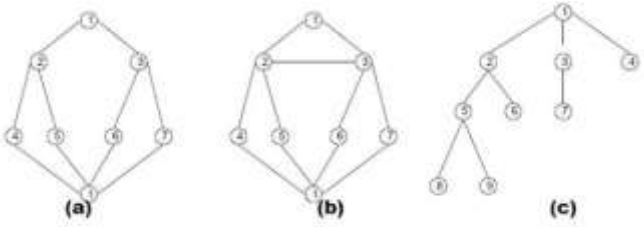
Algoritma ini memerlukan sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpul-simpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungu masuk ke dalam antrian hanya satu kali. Algoritma ini juga membutuhkan table Boolean untuk menyimpan simpul yang telah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali.

a. CARA KERJA ALGORITMA BFS

Dalam algoritma BFS, simpul anak yang telah dikunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian.
Untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya, berikut langkah-langkah algoritma BFS:

- Masukkan simpul ujung (akar) ke dalam antrian.
- Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi.
- Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan..
- Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian.
- Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan.
- Ulangi pencarian dari langkah kedua.

b. Contoh Metode Pencarian BFS (Breadth First Search)



Gambar 1, diambil dari saungkode.wordpress.com

Maka penyelesaiannya adalah:

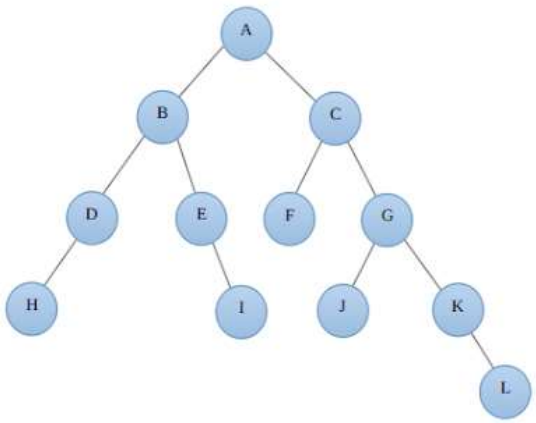
Gambar (a) BFS(1): 1, 2, 3, 4, 5, 6, 7, 1.
Gambar (b) BFS(1): 1, 2, 3, 4, 5, 6, 7, 1
Gambar (c) BFS(1): 1, 2, 3, 4, 5, 6, 7, 8, 9

2. DFS (DEPTH FIRST SEARCH)

DFS (Depth First Search) adalah salah satu algoritma penelusuran struktur graf atau pohon berdasarkan kedalaman. Simpul ditelusuri dari root kemudian ke salah satu simpul anaknya (misal nya prioritas penelusuran berdasarkan anak pertama [simpul sebelah kiri]), maka penelusuran dilakukan terus melalui simpul anak pertama dari simpul anak pertama level sebelumnya hingga mencapai level terdalam.

Setelah sampai di level terdalam, penelusuran akan kembali ke 1 level sebelumnya untuk menelusuri simpul anak kedua pada pohon biner [simpul sebelah kanan] lalu kembali ke langkah sebelumnya dengan menelusuri simpul anak pertama lagi sampai level terdalam dan seterusnya.

a. Contoh pohon biner Depth First Search :



Gambar 2, diambil dari saungkode.wordpress.com

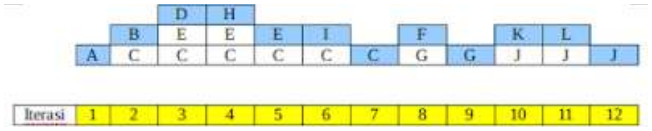
Maka, urutan penelusurannya adalah : A – B – D – H – E – I – C – F – G – J – K – L

Dalam implementasinya DFS dapat diselesaikan dengan cara rekursif atau dengan bantuan struktur data stack. Kita akan membahas dengan cara yang menggunakan stack. Stack yang digunakan adalah stack yang isi elemennya adalah simpul pohon / tree. Bagaimana cara kerjanya ? Berikut ini adalah urutan algoritmanya :

- Masukkan simpul root ke dalam tumpukan dengan push.
- Ambil dan simpan isi elemen (berupa simpul pohon) dari tumpukan teratas.
- Hapus isi stack teratas dengan prosedur pop.
- Periksa apakah simpul pohon yang disimpan tadi memiliki anak simpul.
- Jika ya, push semua anak simpul yang dibangkitkan ke dalam stack.

- Jika tumpukan kosong berhenti, tapi jika tidak kembali ke langkah dua.

Jadi, untuk gambar pohon biner di atas urutan langkah dan kondisi stack-nya setiap iterasi adalah :



Gambar 3, diambil dari saungkode.wordpress.com

Contoh diatas menggunakan prioritas untuk memasukkan anak simpul dari sebelah kanan terlebih dahulu ke dalam stack. Sehingga, pada iterasi 2 elemen A dihapus lalu memasukkan anak simpulnya yaitu C dulu, baru B ke dalam stack. Selain itu bisa dilihat stack teratas (yang diwarna biru) pada tiap iterasi memiliki urutan A – B – D – H – E – I – C – F – G – J – K – L. Pada iterasi ke 13 itu kondisi stack sudah kosong karena ketika simpul J dibangkitkan tidak ada anak simpul yang dimasukkan ke stack.

III. PENGGUNAAN ALGORITMA BFS DAN DFS UNTUK RENCANA STUDI

A. Penggunaan DFS untuk verifikasi rencana studi

1. Pertama-tama, buatlah program yang dapat menyimpan data tiap semester yang ada
2. Cek mata kuliah di semester 1 apakah tidak ada prerequisite yang belum terpenuhi atau tidak, jika terpenuhi maka lanjut ke tahap selanjutnya jika tidak maka rencana studi tidak baik (terdapat prerequisite yang belum terpenuhi).
3. Lakukan penghapusan prerequisite mata kuliah tersebut di mata kuliah lainnya di semester semester selanjutnya.
4. Lakukan hal yang sama dengan tahap 2 sampai semua semester habis.
5. Jika sampai tahap ini dan semua semester telah dicek, maka rencana studi yang dibuat baik. (semua prerequisite sudah dipenuhi)

Contoh sederhana penerapannya yaitu diberikan data berikut ini :

Data Prerequisite Mata Kuliah :

```
FI1201 : FI1101
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

Data Semester :

```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

Pengecekan dilakukan seperti berikut :

1. Semester 1 : FI1101 , KI1102, KU1072 prerequisite telah dipenuhi semua (prerequisite tidak ada), lanjut ke tahap berikutnya.
2. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : IF1210, -
IF2210 : IF2110, IF1210, -

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

3. Semester 2 : FI1201 , KI1202, IF1210 prerequisite telah dipenuhi semua lanjut ke tahap berikutnya.
4. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : IF2110, -, -

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

5. Semester 3 : IF2110 prerequisite telah dipenuhi semua lanjut ke tahap berikutnya.
6. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : -, -, -

(Strip menandakan sudah dihapus)
```

7. Semester 4 : IF2210 prerequisite telah dipenuhi semua lanjut ke tahap berikutnya.
8. Semua semester telah dicek. Berarti rencana studi baik (tidak ada prerequisite yang belum diambil atau terpenuhi)

B. Penggunaan BFS untuk verifikasi rencana studi

Algoritma yang digunakan yaitu sebagai berikut :

1. Pertama-tama, buatlah program yang dapat menyimpan data tiap semester yang ada
2. Cek mata kuliah di semester 1 apakah tidak ada prerequisite yang belum terpenuhi atau tidak, jika terpenuhi maka lanjut ke tahap selanjutnya jika tidak maka rencana studi tidak baik (terdapat prerequisite yang belum terpenuhi)
3. Pengecekan dilanjut untuk mata kuliah tersebut dengan menghapus prerequisite mata kuliah tersebut di mata kuliah lainnya di semester selanjutnya sampai tidak ada lagi mata kuliah yang memiliki prerequisite tersebut.
4. Lanjutkan pengulangan sampai seluruh mata kuliah telah tercek.

Contoh sederhana penerapannya yaitu diberikan data berikut ini :

Data Prerequisite Mata Kuliah :
```
FI1201 : FI1101
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

Data Semester :
```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

Pengecekan dilakukan seperti berikut :

1. Semester 1 : FI1101 , KI1102, KU1072 dilakukan BFS masing-masing. Pertama tama FI1101 dahulu
2. Mata kuliah FI1101 tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
3. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

4. Mata kuliah `FI1201` sudah tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
5. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072

(Strip menandakan sudah dihapus)
```

Karena tidak ada mata kuliah yang memiliki prerequisite FI1201, maka kembali ke list awal (KI1102, KU1072)

6. Mata kuliah KI1102tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
7. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

8. Mata kuliah `KI1202` sudah tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
9. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072

(Strip menandakan sudah dihapus)
```

Karena tidak ada mata kuliah yang memiliki prerequisite FI1201, maka kembali ke list awal yang belum dicek (KU1072)

10. Mata kuliah KU1072 tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
11. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
```

```
IF2110 : IF1210, -
IF2210 : IF2110, IF1210, -

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

12. Mata kuliah `IF1210` sudah tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
13. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : IF2110, -, -

(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

14. Mata kuliah `IF2210` tidak memiliki prerequisite, hapus prerequisite yang memiliki mata kuliah ini di table dan masuk ke mata kuliah tersebut untuk pencariannya.
15. Penghapusan data prerequisite dalam tabel yang ada :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : -, -, -
(Strip menandakan sudah dihapus)
```

Lalu lanjut ke tahap berikutnya

16. Karena semua mata kuliah yang diambil telah dicek, maka rencana kuliah tersebut valid

C. Penggunaan BFS untuk pembuatan rencana studi dengan semester seminimal mungkin

1. Pertama-tama, buatlah program yang dapat menyimpan data tiap semester yang ada
2. Ambil semua mata kuliah dengan prerequisite nol di semester 1, lalu hapus seluruh prerequisite mata kuliah yang telah diambil di data prerequisite
3. Tambah semester, lakukan pengulangan tahap 2 sampai seluruh mata kuliah terambil

Contoh sederhana penerapannya yaitu diberikan data berikut ini :

Data Prerequisite Mata Kuliah :

```
FI1201 : FI1101
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

Pembuatan rencana studi dilakukan seperti berikut :

1. Ambil seluruh mata kuliah yang prerequisite nya nol menghasilkan data sebagai berikut :

Data Semester :

```
Semester 1 : FI1101 , KI1102, KU1072
```

2. Lakukan penghapusan prerequisite mata kuliah yang telah diambil sehingga menjadi seperti berikut :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : IF1210, -
IF2210 : IF2110, IF1210, -
```

3. Masuk ke semester 2 (1+1), Ambil seluruh mata kuliah yang prerequisite nya sudah nol menghasilkan data sebagai berikut :

Data Semester :

```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
```

4. Lakukan penghapusan prerequisite mata kuliah yang telah diambil sehingga menjadi seperti berikut :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : IF2110, -, -
```

5. Masuk ke semester 3 (2+1), Ambil seluruh mata kuliah yang prerequisite nya sudah nol menghasilkan data sebagai berikut :

Data Semester :

```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
Semester 3 : IF2110
```

6. Lakukan penghapusan prerequisite mata kuliah yang telah diambil sehingga menjadi seperti berikut :

Data Prerequisite Mata Kuliah :

```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : -, -, -
```

7. Masuk ke semester 4 (3+1), Ambil seluruh mata kuliah yang prerequisite nya sudah nol menghasilkan data sebagai berikut :

Data Semester :
```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

8. Lakukan penghapusan prerequisite mata kuliah yang telah diambil sehingga menjadi seperti berikut :

Data Prerequisite Mata Kuliah :
```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : -, -
IF2210 : -, -, -
```

9. Selesai, berikut merupakan jumlah semester minimal dengan rencana studinya yang dapat diambil :

Data Semester :
```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201 , KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

D. Penggunaan DFS untuk pembuatan rencana studi dengan semester seminimal mungkin

1. Pertama-tama, buatlah program yang dapat menyimpan data tiap semester yang ada
2. Ambil semua mata kuliah dengan prerequisite nol di semester 1, lalu lakukan penghapusan prerequisite dari satu-per-satu mata kuliah dengan semua prerequisite dari mata kuliah tersebut dimasukan di semester 1+x dengan nilai x merupakan banyaknya prerequisite yang ada sebelumnya
3. Lakukan pengulangan sampai seluruh mata kuliah terpenuhi.

Contoh sederhana penerapannya yaitu diberikan data berikut ini :

Data Prerequisite Mata Kuliah :
```
FI1201 : FI1101
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

Pembuatan rencana studi dilakukan seperti berikut :

1. Ambil seluruh mata kuliah yang prerequisite nya nol menghasilkan data sebagai berikut :

Data Semester :
```
Semester 1 : FI1101 , KI1102, KU1072
```

2. Lakukan penghapusan prerequisite mata kuliah yang telah diambil sehingga menjadi seperti berikut dan masuk ke mata kuliah yang memiliki prerequisite tersebut sampai habis.
3. Pertama-tama, dari list yang ada (`FI1101 , KI1102, KU1072`) FI1101 dahulu dicari seluruh mata kuliah yang prerequisite nya mata kuliah FI1101

Data Prerequisite Mata Kuliah :
```
FI1201 : -
KI1202 : KI1102
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

4. FI1201 diambil di semester selanjutnya setelah FI1101 menjadikan data semester seperti berikut :

Data Semester :
```
Semester 1 : FI1101 , KI1102, KU1072
Semester 2 : FI1201
```

5. Karena semua yang memiliki prerequisite FI1101 sudah habis, maka lanjut ke mata kuliah berikutnya di list (KI1102, KU1072) yaitu K1102 dicari seluruh mata kuliah yang prerequisite nya mata kuliah K1102

Data Prerequisite Mata Kuliah :
```
FI1201 : -
KI1202 : -
IF1210 : KU1072
IF2110 : IF1210, KU1072
IF2210 : IF2110, IF1210, KU1072
```

6. `KI1202` diambil di semester selanjutnya setelah K1102 menjadikan data semester seperti berikut :

Data Semester :
```
Semester 1 : FI1101, KI1102, KU1072
Semester 2 : FI1201, KI1202
```

7. Karena semua yang memiliki prerequisite FI1101 sudah habis, maka lanjut ke mata kuliah berikutnya di list (KU1072) yaitu KU1072dicari seluruh mata kuliah yang prerequisite nya mata kuliah KU1072

Data Prerequisite Mata Kuliah :
```
FI1201 : -
KI1202 : -
IF1210 : -
IF2110 : IF1210, -
IF2210 : IF2110, IF1210, -
```

8. `IF1210` diambil di semester selanjutnya setelah KU1072 menjadikan data semester seperti berikut :

Data Semester :
```
Semester 1 : FI1101, KI1102, KU1072
Semester 2 : FI1201, KI1202, IF1210
```

9. IF2110 terdapat di prerequisite yang lain sehingga dapat dilakukan pencarian ke dalam lagi sehingga data prerequisite mata kuliah menjadi :
   Data Prerequisite Mata Kuliah :
   ```
   FI1201 : -
   KI1202 : -
   IF1210 : -
   IF2110 : -, -
   IF2210 : IF2110, -, -
   ```

10. `IF2110` diambil di semester selanjutnya setelah `IF1210` menjadikan data semester seperti berikut :

Data Semester :
```
Semester 1 : FI1101, KI1102, KU1072
Semester 2 : FI1201, KI1202, IF1210
Semester 3 : IF2110
```

11. `IF2110` terdapat di prerequisite yang lain sehingga dapat dilakukan pencarian ke dalam lagi sehingga data prerequisite mata kuliah menjadi :

    Data Prerequisite Mata Kuliah :
    ```
    FI1201 : -
    KI1202 : -
    IF1210 : -
    IF2110 : -, -
    IF2210 : -, -, -
    ```

12. `IF2210` diambil di semester selanjutnya setelah `IF2110` menjadikan data semester seperti berikut :

Data Semester :
```
Semester 1 : FI1101, KI1102, KU1072
Semester 2 : FI1201, KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

13. Selesai karena semua mata kuliah telah diambil. Berikut rencana studi yang didapatkan :

Data Semester :
```
Semester 1 : FI1101, KI1102, KU1072
Semester 2 : FI1201, KI1202, IF1210
Semester 3 : IF2110
Semester 4 : IF2210
```

V. KESIMPULAN

Algoritma BFS dan DFS ini dapat digunakan untuk memvalidasi rencana studi para mahasiswa yang sudah diberikan jadwal studi yang akan dilakukan di masa perkuliahan. Program ini dapat mengecek data rencana studi mahasiswa dengan pasti sehingga mahasiswa dapat merencanakan kegiatan perkuliahan sebelumnya sehingga dapat mengambil kuliahnya dengan baik.

Algoritma ini memiliki kelebihan dan kekuarangan yang pasti ada ketika algoritma ini dipakai. Algoritma DFS memiliki kelebihan di kecepatannya sedangkan kekurangannya yaitu kita harus memberi limit kedalaman yang harus di cek sehingga tidak looping programnya dan mungkin tidak memberikan solusi keseluruhan. Algoritma BFS di lain sisi tidak perlu limit apapun dan pasti memberikan solusi keseluruhan tetapi tidak secepat algoritma DFS dalam perbandingan kecepatannya.

VII. PENGHARGAAN

Alhamdulillah, penulis mengucapkan terima kasih kepada Allah subhanahu wa Ta'ala karena berkat-Nya penulis telah dapat menyelesaikan makalah ini sebagaimana mestinya. Ucapan terima kasih juga saya sampaikan kepada orang tua saya karena dukungan merekalah saya dapat menggenggam kuliah ini.

Terima kasih khususnya kepada dosen pengajar mata kuliah Matematika Diskrit kelas 3 Bapak Rinaldi Munir yang telah mengajar di kelas kami dan umumnya tim pengajar dosen mata kuliah Strategi Algoritma di kuliah IF semester 4 ini.

REFERENSI

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp. 540–549.
[2] Goodrich, Michael T.; Tamassia, Roberto (2001), Algorithm Design: Foundations, Analysis, and Internet Examples, Wiley, ISBN 0-471-38365-1
[3] Kleinberg, Jon; Tardos, Éva (2006), Algorithm Design, Addison Wesley, pp. 92–94
[4] Knuth, Donald E. (1997), The Art of Computer Programming Vol 1. 3rd ed, Boston: Addison-Wesley, ISBN 0-201-89683-4, OCLC 155842391
[5] "Graph500 benchmark specification (supercomputer performance evaluation)". Graph500.org, 2010.
[6] Zuse, Konrad (1972), Der Plankalkül (in German), Konrad Zuse Internet Archive. See pp. 96–105 of the linked pdf file (internal numbering 2.47–2.56).
[7] Moore, Edward F. (1959). "The shortest path through a maze". Proceedings of the International Symposium on the Theory of Switching. Harvard University Press. pp. 285–292. As cited by Cormen, Leiserson, Rivest, and Stein.
[8] Skiena, Steven (2008). The Algorithm Design Manual. Springer. p. 480. doi:10.1007/978-1-84800-070-4_4.
[9] Lee, C. Y. (1961). "An Algorithm for Path Connections and Its Applications". IRE Transactions on Electronic Computers.
[10] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. "22.2 Breadth-first search". Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill. pp. 531–539. ISBN 0-262-03293-7.
[11] Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall. ISBN 978-0137903955.
[12] Coppin, B. (2004). Artificial intelligence illuminated. Jones & Bartlett Learning. pp. 79–80.
[13] Aziz, Adnan; Prakash, Amit (2010). "4. Algorithms on Graphs". Algorithms for Interviews. p. 144. ISBN 1453792996.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandumg, 14 Mei 2018

Muhamad Arif Adiputra
13516114