Laporan Tugas Kecil II IF2211 Strategi Algoritma

Penyusunan Rencana Kuliah dengan Topological Sort

Nama : Fabian Savero Diaz Pranoto

NIM : 13519140

Kelas : 03

A. Pendahuluan

Salah pendekatan sorting yang menggunakan algoritma Decrease and Conquer adalah metode Topological Sort. Topological Sort merupakan sebuah metode untuk mengurutkan node-node pada sebuah Directed Acrylic Graph (DAG) sehingga untuk setiap tepi uv dari simpul u menuju v, simpul u diurutkan sebelum simpul v. Algoritma topological sort dalam pseudocode sebagai berikut:

```
Algorithm TopoSource(G(V, E))
    make empty Container of vertices, C
    make empty List of vertices, S
    for each vertex v in V do
        incounter(v) ← in-deg(v)
        if incounter(v) == 0 then C.add(v)
    while C is not empty do
        u ← C.remove()
        S.add(u)
        for each edge (u, w) do  // edge is out from u to w
            incounter(w) ← incounter(w)-1
            if incounter(w) == 0 then C.add(w)
    if L.size() == V.size() then return S
    else return "G has a cycle"
```

Terlihat dari pseudocode di atas, pada topological sort dilakukan pengecekan setiap simpul pada graf. Ketika ada simpul yang memiliki derajat masuk sebesar 0, hapus dari graf, masukkan ke list solusi, dan kurangi derajat masuk simpul lain yang mengarah ke simpul yang sudah dihapus. Ulangi langkah-langkah sebelumnya sampai graf tidak memiliki simpul lagi. List S lalu akan memiliki urutan simpul yang benar. Karena dilakukan penghapusan simpul yang perlu diperiksa setiap kali ada simpul yang memenuhi syarat, algoritma ini tergolong Decrease and Conquer.

B. Penjelasan Algoritma

Untuk melakukan Topological Sort terhadap sebuah graf yang berisi mata kuliah untuk menghasilkan rencana kuliah, dibuatlah sebuah algoritma Decrease dan Conquer dengan penerapan fungsi rekursif pada pengecekan simpul. Dalam algoritma ini, dibuat empat fungsi beserta penerapan main program sebagai berikut:

1. **initializeGraph**
   Fungsi initializeGraph merupakan fungsi yang akan mengembalikan sebuah Directed Acrylic Graph (DAG) yang memiliki bentuk python dictionary dengan key berupa nama node dan value berupa sebuah adjacency list. Fungsi ini menerima sebuah argumen berupa nama file txt yang perlu dibuka. Cara kerja fungsi ini sebagai berikut:
   a. Membuka file dengan nama yang telah dimasukkan sebagai parameter.
   b. Menginisialisasi sebuah dictionary bernama grafKuliah.
   c. Mengiterasi setiap baris pada file txt:
      i. Mengeprint baris pada layar.
      ii. Menggantikan karakter koma dan titik dengan karakter kosong.
      iii. Melakukan split string dan hasilnya dinamakan arrayKuliah.

iv. Memasukkan elemen pertama arrayKuliah sebagai key dalam dictionary dan valuenya berupa elemen array sisanya.

d. Mengembalikan sebuah graf berbentuk dictionary bernama grafKuliah.

2. recursiveSortHelper

Fungsi recursiveSortHelper merupakan sebuah fungsi rekursif yang akan membantu untuk melakukan topological sort dalam mengecek setiap simpul dalam graf. Fungsi ini menerima empat argumen berupa sebuah himpunan simpul yang sudah dikunjungi, list solusi, graf, dan simpul yang dicari. Cara kerjanya sebagai berikut:

a. Mengecek simpul yang dimasukkan sebagai parameter.
b. Mengiterasi setiap simpul tetangga yang ada di list tetangga:
    i. Jika simpul tetangga belum pernah dikunjungi, masukkan ke dalam himpunan simpul lalu panggil fungsi recursiveSortHelper untuk mengecek simpul tersebut.
    ii. Jika pernah dikunjungi, lompati.
c. Jika simpul yang dimasukkan sebagai parameter belum di dalam list solusi, masukkan simpul tersebut ke dalam list.

Aksi fungsi ini dalam memasukkan simpul ke dalam himpunan dikunjungi merupakan contoh dari pengurangan masalah sehingga algoritma ini dapat disebut sebagai algoritma Decrease and Conquer.

3. topoSort

Fungsi topoSort merupakan fungsi untuk mendapatkan solusi Topological Sort sebuah graf. Fungsi ini menerima sebuah argumen berupa graf yang berbentuk dictionary. Cara kerjanya sebagai berikut:

a. Menginisialisasi sebuah list solusi kosong.
b. Menginisialisasi sebuah set kunjungan yang kosong.
c. Mengiterasi setiap key pada graf yang dimasukkan sebagai parameter:
    i. Untuk setiap key, cek dengan memanggil fungsi recursiveSortHelper.
d. Mengembalikan sebuah list solusi yang berisi urutan Topological Sort yang benar.

4. printSolution

Fungsi printSolution ini merupakan fungsi untuk mencetak hasil solusi pada list solusi. Fungsi ini menerima sebuah argumen berupa list solusi yang memiliki isi urutan nama simpul hasil Topological Sort. Cara kerjanya sebagai berikut:

a. Menampilkan tulisan “Rencana Kuliah:”.
b. Mengiterasi setiap elemen list solusi dan menampilkan setiap elemennya kepada layar/terminal.

5. Main Program

Main program berisi pemanggilan fungsi-fungsi yang telah dijelaskan sebelumnya. Cara kerjanya sebagai berikut:

a. Meminta input nama file txt yang akan dilakukan Topological Sort.
b. Menginisialisasi graf dengan memanggil fungsi initializeGraph dan menyimpannya dengan nama grafKuliah.
c. Melakukan Topological Sort dengan memanggil fungsi topoSort dan menyimpan hasil sortnya pada list bernama solution.
d. Menampilkan solusi dengan memanggil fungsi printSolution.

C. Source Code Program

```python
# Fungsi untuk inisialisasi graf berbentuk dictionary
def initializeGraph(filename):
    f = open("test/" + filename)

    # Inisialisasi dictionary dengan
    # key kode_kuliah dan value kuliah_prasyarat
    grafKuliah = dict()

    # Baca setiap line, matkul pertama jadi key
    # dan sisanya menjadi value berbentuk array
    print("Isi Berkas:")
    for line in f.read().splitlines():
        print(line)
        arrayKuliah = line.replace(".", "").replace(" ", "").split(",")
        grafKuliah[arrayKuliah[0]] = arrayKuliah[1:]

    f.close()

    return grafKuliah
```

```python
# Fungsi rekursif untuk pengecekan node di toposort
def recursiveSortHelper(visited, solution, graf, node):
    # Loop adjacency list di setiap node/key
    # Jika memiliki adjacency list, loop dan apabila tidak ada di visited maka
    # panggil fungsi rekursif untuk pengecekan node adjacent tersebut
    # Jika node tidak memiliki node adjacent, langsung cek apakah node ada di solusi
    # Jika node tidak ada di solusi, append ke list solution
    for adjacent in graf[node]:
        if adjacent not in visited:
            visited.add(adjacent)
            recursiveSortHelper(visited, solution, graf, adjacent)

    if node not in solution:
        solution.append(node)
```

```python
# Fungsi toposort utama
def topoSort(graf):
    # Solution merupakan list hasil akhir toposort
    # Visited merupakan sebuah set yang berisi node yang sudah
    # dikunjungi
    solution = list()
    visited = set()

    # Loop setiap key di graf dan panggil fungsi rekursif
    for key in graf.keys():
        recursiveSortHelper(visited, solution, graf, key)

    return solution
```

```python
# Fungsi untuk print solusi
def printSolution(solution):
    print("\nRencana Kuliah:")
    for i in range(len(solution)):
        print(str(i+1)+".", solution[i])
```

```python
# Main Program
filename = input("Masukkan nama berkas: ")
grafKuliah = initializeGraph(filename) # Inisialisasi graf
solution = topoSort(grafKuliah) # list solusi
printSolution(solution) # print solusi
```

D. **Screenshot Input/Output**

```
Masukkan nama berkas: no1.txt
Isi Berkas:
MA1101.
MA1201.
F11201.
El1200, MA1101.
IF2120.
IF2110.
IF2210, IF2110.
IF2220, MA1101, MA1201, IF2120.

Rencana Kuliah:
1. MA1101
2. MA1201
3. F11201
4. El1200
5. IF2120
6. IF2110
7. IF2210
8. IF2220
```

```
Masukkan nama berkas: no3.txt
Isi Berkas:
IF1102, IF1011.
IF1103.
IF1105.
IF1107.
IF1108, IF1103, IF1107.
IF1109, IF1108, IF1011.
IF1110, IF1103, IF1011.
IF1011, IF1105, IF1107.

Rencana Kuliah:
1. IF1105
2. IF1107
3. IF1011
4. IF1102
5. IF1103
6. IF1108
7. IF1109
8. IF1110
```

```
Masukkan nama berkas: no4.txt
Isi Berkas:
MA1201.
FI1201.
IF2110.
IF2121, IF2120, IF2110.
IF2211, IF2120, MA1201.
IF2120, MA1201.
IF4020, IF2120, IF2110.

Rencana Kuliah:
1. MA1201
2. FI1201
3. IF2110
4. IF2120
5. IF2121
6. IF2211
7. IF4020
```

```
Masukkan nama berkas: no7.txt
Isi Berkas:
FI1201, PL1210, PI1220.
PI1220, PL1210.
PL1210, CI1000.
CI1000.

Rencana Kuliah:
1. CI1000
2. PL1210
3. PI1220
4. FI1201
```

```
Masukkan nama berkas: no2.txt
Isi Berkas:
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.
MA1201, MA1101.
FI1201, FI1101.
IF1210, KU1102.
KU1202, KU1102.
KI1002, KU1011.
EL1200, FI1101.
IF2121, IF1210, MA1101, MA1201.
IF2110, KU1102, IF1210.
IF2120, MA1201, MA1101.
IF2124, EL1200.
IF2123, MA1201.
IF2130, KU1202.
IF2210, IF2110.
IF2211, IF2110.
IF2220, MA1101, MA1201, IF2120.
IF2230, IF2130.
IF2240, IF2121, IF2120.
IF2250, KU1202, IF2110.
IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140, IF2240.
IF3151, IF2250.
IF3210, IF2110, IF2130, IF3110.
IF3270, IF2210, IF3170.
IF3230, IF3130.
IF3250, IF2250, IF3150.
IF3260, IF2123, IF2110, IF2130, IF3151.
IF3280, IF3151, IF3150.
IF4090, IF3280.
IF4091, IF3280.
IF4092, IF4091.
```

```
Rencana Kuliah:
1. MA1101
2. FI1101
3. KU1001
4. KU1102
5. KU1011
6. KU1024
7. MA1201
8. FI1201
9. IF1210
10. KU1202
11. KI1002
12. EL1200
13. IF2121
14. IF2110
15. IF2120
16. IF2124
17. IF2123
18. IF2130
19. IF2210
20. IF2211
21. IF2220
22. IF2230
23. IF2240
24. IF2250
25. IF3170
26. IF3110
27. IF3130
28. IF3141
29. IF3150
30. IF3140
31. IF3151
32. IF3210
33. IF3270
34. IF3230
35. IF3250
36. IF3260
37. IF3280
38. IF4090
39. IF4091
40. IF4092
```

```
Masukkan nama berkas: no6.txt
Isi Berkas:
MA1101.
FI1101.
KU1001.
KU1102.
KU1011.
KU1024.
MA1201, MA1101.
FI1201, FI1101.
IF1210.
KU1202.
EL1200, MA1101.
IF2121.
IF2110.
IF2120.
IF2124.
IF2123, MA1101.
IF2130.
IF2210, IF2110.
IF2211.
IF2220, MA1101, MA1201, IF2120.
IF2230.
IF2240.
IF2250.
IF3170, IF2121, IF2124, IF2220, IF2211.
IF3110, IF2210, IF2110.
IF3130, IF2230.
IF3141, IF2240, IF2250.
IF3150, IF2250.
IF3140.
IF3151, IF2250.
IF3210, IF2130, IF2110.
IF3270, IF3170, IF2110.
IF3230, IF3130.
IF3250, IF3150, IF2250.
IF3260, IF2130, IF2110, IF2123.
IF3280.
IF4090, IF3280.
IF4091.
KU2071.
IF4092, IF4091.
KU206X.
AS2005.
```

```
Rencana Kuliah:
1. MA1101
2. FI1101
3. KU1001
4. KU1102
5. KU1011
6. KU1024
7. MA1201
8. FI1201
9. IF1210
10. KU1202
11. EL1200
12. IF2121
13. IF2110
14. IF2120
15. IF2124
16. IF2123
17. IF2130
18. IF2210
19. IF2211
20. IF2220
21. IF2230
22. IF2240
23. IF2250
24. IF3170
25. IF3110
26. IF3130
27. IF3141
28. IF3150
29. IF3140
30. IF3151
31. IF3210
32. IF3270
33. IF3230
34. IF3250
35. IF3260
36. IF3280
37. IF4090
38. IF4091
39. KU2071
40. IF4092
41. KU206X
42. AS2005
```

```
Masukkan nama berkas: no5.txt
Isi Berkas:
MA1201, MA1101.
FI1201, FI1101.
IF1210, KU1102.
KU1202, KU1102.
KI1002, KU1011.
EL1200, FI1101.
KU1102.
MA1101.
FI1101.
KU1011.

Rencana Kuliah:
1. MA1101
2. MA1201
3. FI1101
4. FI1201
5. KU1102
6. IF1210
7. KU1202
8. KU1011
9. KI1002
10. EL1200
```

```
Masukkan nama berkas: no8.txt
Isi Berkas:
MA1101.
MA1201, MA1101.
IF2130.
IF2110.
IF2121.
IF2124, IF2120, IF2110.
IF2120.
IF2220, MA1101, MA1201, IF2120.
IF2211.
IF3170, IF2121, IF2124, IF2220, IF2211.
IF3210, IF2130, IF2110.
IF3270, IF3170, IF2110.

Rencana Kuliah:
1. MA1101
2. MA1201
3. IF2130
4. IF2110
5. IF2121
6. IF2120
7. IF2124
8. IF2220
9. IF2211
10. IF3170
11. IF3210
12. IF3270
```

E. Alamat Github

https://github.com/fabiansdp/TucilStima/tree/main/TopoSort

F. Checklist

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.	☒	☐