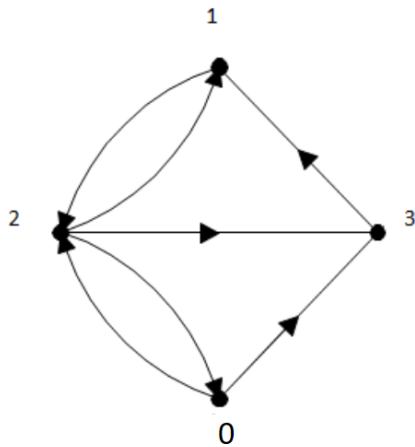


TUGAS PRAKTIKUM STRUKDAT PERTEMUAN 11

Jumat, 12 Mei 2023

1. Notasikan graph berikut dalam bentuk Adjacency list, edge list, dan adjacency matriks.

a.



Jawab:

- Adjacency list

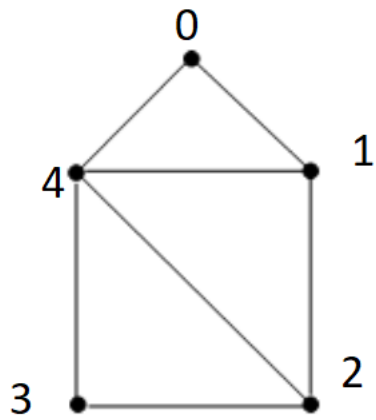
0	2	3	
1	2	3	
2	0	1	3
3	0	1	2

- Edge list
[(0,3),(0,2),(2,3),(2,0),(2,1),(1,2),(3,1)]

- Adjacency matrix

	0	1	2	3
0	0	0	1	1
1	0	0	1	1
2	1	1	0	1
3	1	1	0	0

b.



Jawab:

- Adjacency list

0	1	4		
1	0	2	4	
2	1	3	4	
3	2	4		
4	0	1	2	3

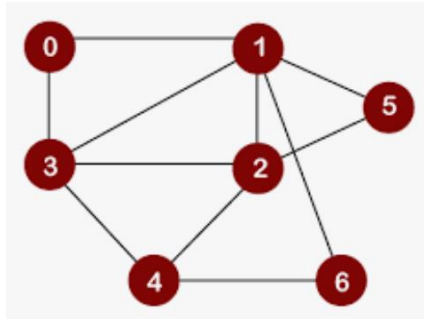
- Edge list

[(0,1),(0,4),(1,2),(1,4),(2,1),(2,3),(2,4),(3,2),(3,4),(4,0),(4,1),(4,2),(4,3)]

- Adjacency list

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	0	1
2	0	1	0	1	1
3	0	0	1	0	1
4	1	1	1	1	0

2. Diberikan graph berikut, telusurilah setiap verteknya menggunakan algoritma: (a) DFS dan (b) BFS (starting vertexnya vertex 0). Sebagai catatan, jika pada satu titik ada dua opsi yang setara maka dahulukanlah penelusuran vertex dengan no ID yang lebih kecil.



Jawab :

- DFS
 1. Mulai dari node 0, tandai node 0 sebagai visited (dikunjungi).
 2. Pilih node tetangga dengan ID terkecil dari node 0 yang belum dikunjungi, yaitu node 1.
 3. Kunjungi node 1 dan tandai sebagai visited.
 4. Pilih node tetangga dengan ID terkecil dari node 1 yang belum dikunjungi, yaitu node 3.
 5. Kunjungi node 3 dan tandai sebagai visited.
 6. Tidak ada node tetangga dari node 3 yang belum dikunjungi, kembali ke node 1.
 7. Pilih node tetangga dengan ID terkecil dari node 1 yang belum dikunjungi, yaitu node 4.
 8. Kunjungi node 4 dan tandai sebagai visited.
 9. Tidak ada node tetangga dari node 4 yang belum dikunjungi, kembali ke node 1.
 10. Pilih node tetangga dengan ID terkecil dari node 1 yang belum dikunjungi, yaitu node 2.
 11. Kunjungi node 2 dan tandai sebagai visited.
 12. Pilih node tetangga dengan ID terkecil dari node 2 yang belum dikunjungi, yaitu node 6.
 13. Kunjungi node 6 dan tandai sebagai visited.
 14. Tidak ada node tetangga dari node 6 yang belum dikunjungi, kembali ke node 2.
 15. Pilih node tetangga dengan ID terkecil dari node 2 yang belum dikunjungi, yaitu node 5.
 16. Kunjungi node 5 dan tandai sebagai visited.
 17. Tidak ada node tetangga dari node 5 yang belum dikunjungi, kembali ke node 2.
 18. Tidak ada node tetangga dari node 2 yang belum dikunjungi, kembali ke node 0.
 19. Selesai.

Hasilnya (0,1,2,3,4,6,5)

```
graph = {
    0:[1,3],
    1:[0,2,3,4,5],
    2:[1,3,4,6],
    3:[0,1,2,4],
    4:[2,3,6],
    5:[1,2],
    6:[1,4]
}

def dfs(graph, start):
    visited = set()

    def dfs_helper(node):
        visited.add(node)
        print(node, end=" ")

        for neighbor in graph[node]:
            if neighbor not in visited:
                dfs_helper(neighbor)

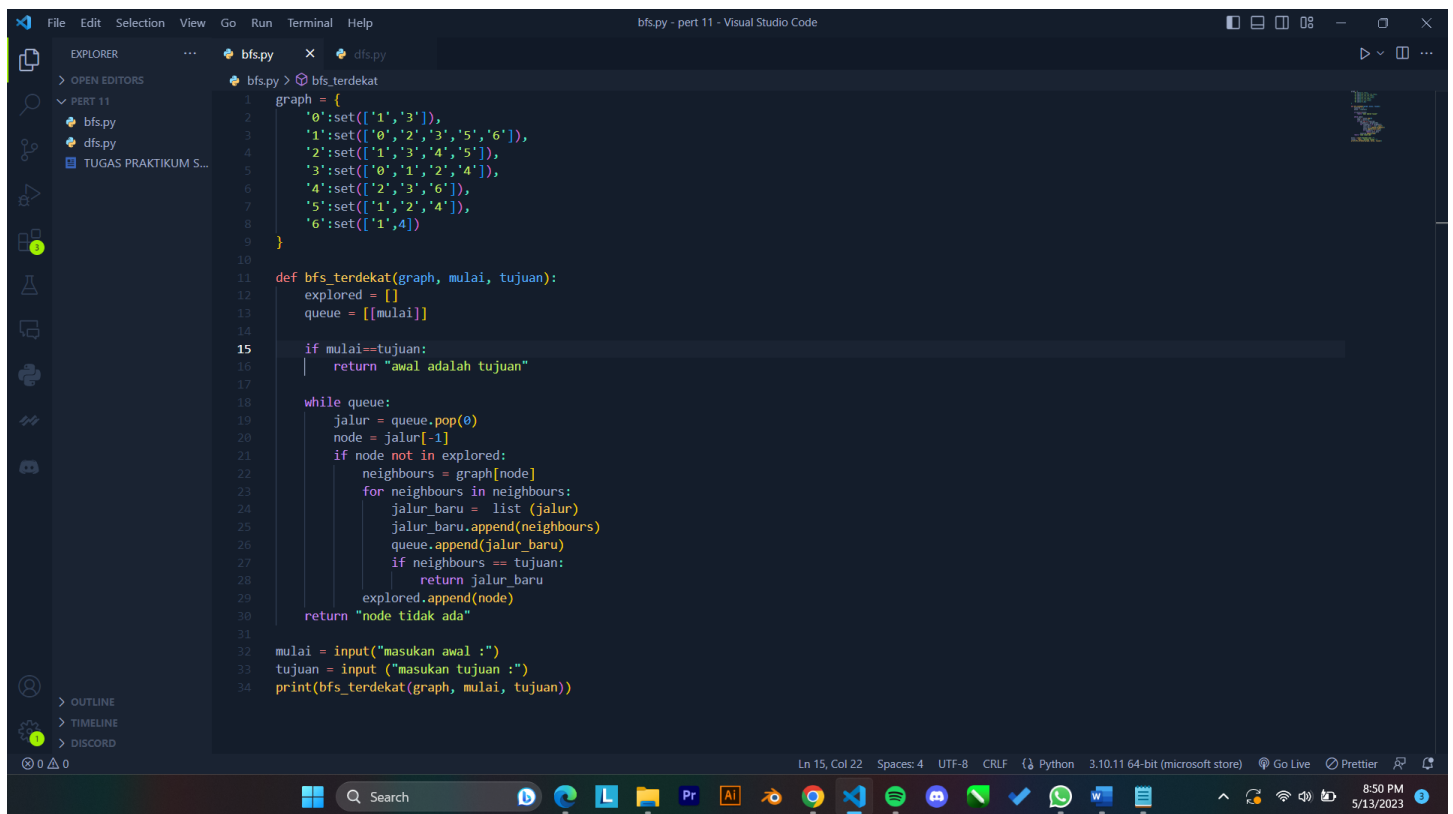
    dfs_helper(start)

dfs(graph, 0)
```

- BFS

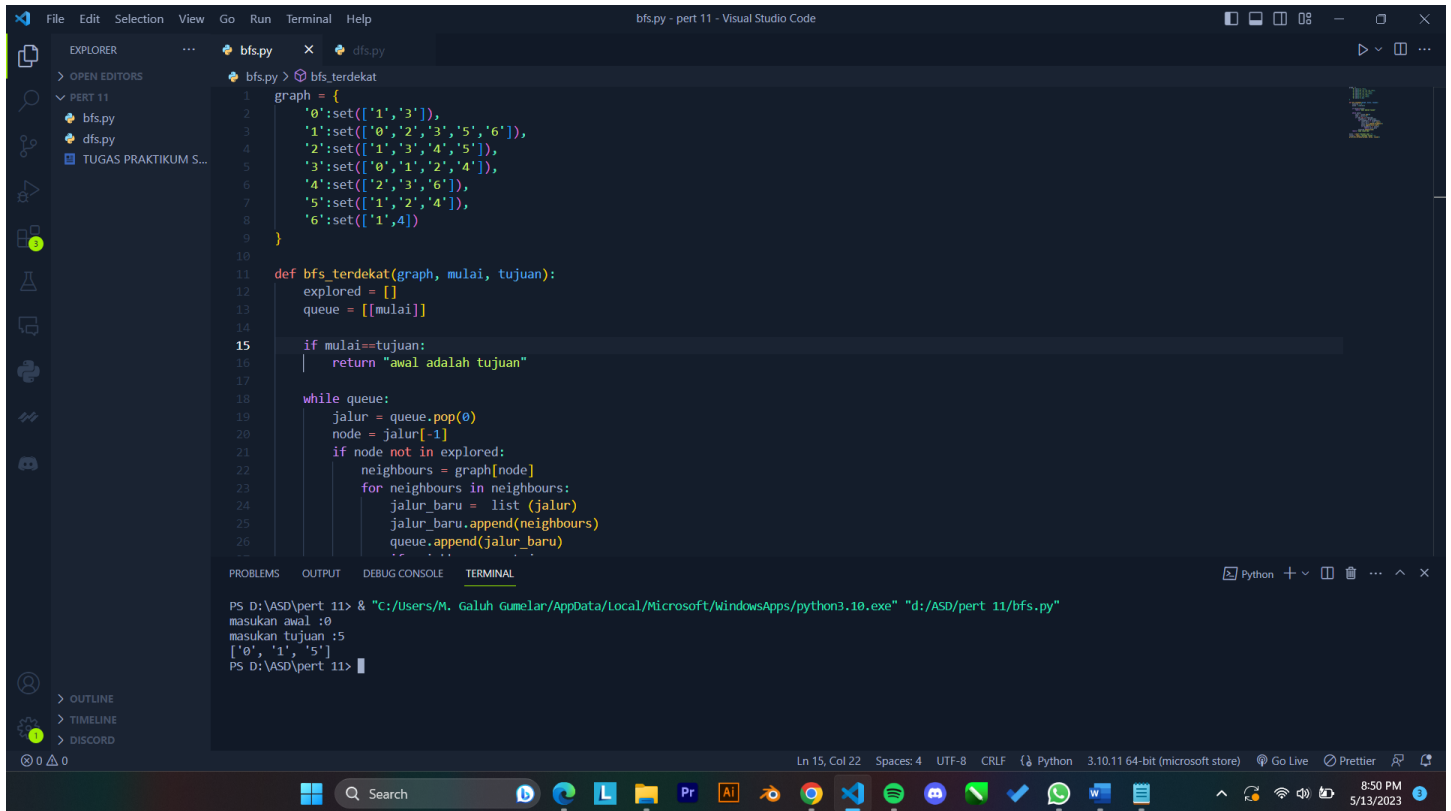
1. Tentukan starting vertex dan target vertex (jika ingin mencari jalur dari satu vertex ke vertex lain).
2. Tandai starting vertex sebagai dikunjungi.
3. Masukkan starting vertex ke dalam queue.
4. Loop sampai queue kosong:
 - a. Ambil node dari depan queue.
 - b. Cek apakah node sama dengan target vertex (jika ingin mencari jalur ke target). i. Jika ya, selesai.
 - c. Tandai node sebagai dikunjungi.
 - d. Tambahkan tetangga node yang belum dikunjungi ke dalam queue.
5. Jika queue kosong dan target vertex belum ditemukan, berarti target tidak dapat dicapai dari starting vertex.

Code :



```
1 graph = {
2     '0':set(['1','3']),
3     '1':set(['0','2','3','5','6']),
4     '2':set(['1','3','4','5']),
5     '3':set(['0','1','2','4']),
6     '4':set(['2','3','6']),
7     '5':set(['1','2','4']),
8     '6':set(['1','4'])
9 }
10
11 def bfs_terdekat(graph, mulai, tujuan):
12     explored = []
13     queue = [[mulai]]
14
15     if mulai==tujuan:
16         return "awal adalah tujuan"
17
18     while queue:
19         jalur = queue.pop(0)
20         node = jalur[-1]
21         if node not in explored:
22             neighbours = graph[node]
23             for neighbours in neighbours:
24                 jalur_baru = list(jalur)
25                 jalur_baru.append(neighbours)
26                 queue.append(jalur_baru)
27                 if neighbours == tujuan:
28                     return jalur_baru
29             explored.append(node)
30     return "node tidak ada"
31
32 mulai = input("masukan awal :")
33 tujuan = input("masukan tujuan :")
34 print(bfs_terdekat(graph, mulai, tujuan))
```

Output :



The image shows a Visual Studio Code editor window with a Python file named `bfs.py` open. The file contains a graph structure and a `bfs_terdekat` function. The graph is defined as follows:

```
graph = {
    '0':set(['1','3']),
    '1':set(['0','2','3','5','6']),
    '2':set(['1','3','4','5']),
    '3':set(['0','1','2','4']),
    '4':set(['2','3','6']),
    '5':set(['1','2','4']),
    '6':set(['1','4'])
}
```

The `bfs_terdekat` function is defined as:

```
def bfs_terdekat(graph, mulai, tujuan):
    explored = []
    queue = [[mulai]]
    if mulai==tujuan:
        return "awal adalah tujuan"
    while queue:
        jalur = queue.pop(0)
        node = jalur[-1]
        if node not in explored:
            neighbours = graph[node]
            for neighbours in neighbours:
                jalur_baru = list(jalur)
                jalur_baru.append(neighbours)
                queue.append(jalur_baru)
                ...
```

The terminal output shows the execution of the script:

```
PS D:\ASD\pert 11> & "C:/Users/M. Galuh Gumelar/AppData/Local/Microsoft/windowsApps/python3.10.exe" "d:/ASD/pert 11/bfs.py"
masukan awal :0
masukan tujuan :5
['0', '1', '5']
PS D:\ASD\pert 11>
```

The status bar at the bottom indicates the file is at line 15, column 22, using UTF-8 encoding and CRLF line endings. The Python version is 3.10.11 64-bit (microsoft store).