Praktikum 2

1. Jelaskan detail dari Langkah ketiga pada percobaan 1

**def** bfs(**visited**,**graph**,**node**):

    visited.append(node)

    queue.append(node)

**while** queue:

        s **=** queue.pop(0)

        print(s,**end** **=** " ")

**for** neighbour **in** graph[s]:

**if** neighbour **not** **in** visited:

                visited.append(neighbour)

                queue.append(neighbour)

detail:

1. Membuat def/fungsi yang memilki parameter berupa visited(untuk yang sudah pernah dikunjungi), graph(yang berisi data), dan node(yang merupakan root dari node/ kali ini graph)
2. Setelah itu pada awalnya kita memasukkan node yang sudah dikunjungi ke dalam queue dan visited, visited dan queue akan terus bertambah karena kita ingin melacak yang sudah dikunjungi
3. Queue akan diisi dengan edges
4. Node yang sudah dikunjungi dan ada di queue akan di pop dan di print
5. ketika tidak ada node yang bersebelahan lagi, edge akan berpindah
6. proses berulang

2. Bagaimana hasil penelusuran graf diatas bila node E terhubung dengan node F dan G (neighbour dari E adalah F dan G)? Mengapa?

Dalam prosesnya ketika graph mencapai index ke E, G akan dimasukkan dalam queue, setelah F, namun tidak sampai merubah urutan dari output karena, akan tetap mengikuti arah BFS yang berurut sesuai level

praktikum 2

1. Jelaskan secara rinci Langkah ketiga dari percobaan 2!

**def** dfs(**visited**, **graph**, **node**):

**if** node **not** **in** visited:

        print(node)

        visited.add(node)

**for** neighbour **in** graph[node]:

            dfs(visited,graph, neighbour)

1. fungsi dibuat dengan parameter visited, graph, dan node
2. jika node(root) tidak ada dalam visited node akan di print dan ditambahkan pada visited
3. setelah itu akan terjadi rekursif, dimana yang menjadi rootnya adalah yang tadi merupakan neighbour

2. Apakah terdapat perbedaan hasil penelusuran bila node B terhubung dengan node C? Mengapa?

Hasilnya akan berbeda, karena C tidak lagi ditampilkan terakhir, namun ditampilkan setelah b karena yang di kunjungi dulu adalah c, dan d akan dikunjungi terakhir, karena