

Nama : Muhammad Galuh Gumelar

Nim : J0403221017

Kelas : BP1

1. Jelaskan perbedaan BFS dan DFS.

Jawab :

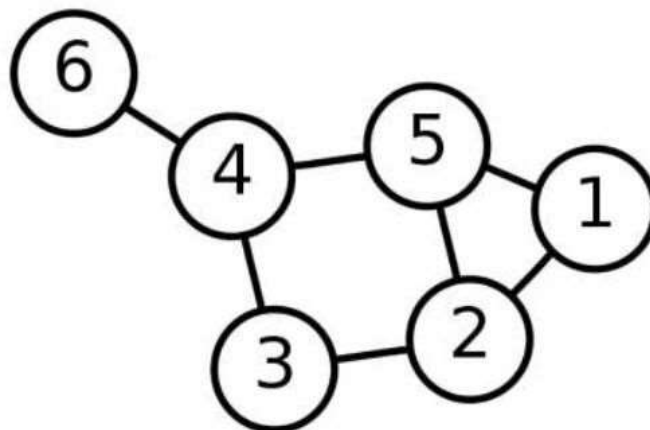
BFS (Breadth-First Search):

- BFS mengunjungi node-node secara melebar atau level per level.
- Dimulai dari node awal, BFS mengunjungi semua tetangga dari node tersebut terlebih dahulu sebelum melanjutkan ke tetangga-tetangga selanjutnya.
- Menggunakan struktur data antrian (queue) untuk melacak node yang akan dikunjungi selanjutnya.
- Algoritma BFS berguna dalam mencari jalur terpendek atau menemukan solusi terdekat dalam graf yang memiliki bobot seragam pada setiap edge.

DFS (Depth-First Search):

- DFS mengunjungi node-node secara dalam atau mencapai kedalaman sejauh mungkin sebelum kembali.
- Dimulai dari node awal, DFS mengunjungi salah satu tetangga dari node tersebut dan melanjutkan ke tetangga-tetangga lainnya secara rekursif sampai tidak ada tetangga lagi yang belum dikunjungi.
- Menggunakan struktur data tumpukan (stack) atau rekursi untuk melacak node yang akan dikunjungi selanjutnya.
- Algoritma DFS berguna dalam mencari jalur tertentu dalam graf, mengeksplorasi seluruh cabang dari suatu node sebelum beralih ke cabang lainnya.

2. Perhatikan graph yang disediakan, carilah BFS dan DFS nya.



Jawab :

- BFS

1. Tentukan node awal yang akan digunakan sebagai titik awal pencarian. Dalam kasus ini, kita menggunakan node awal 1.
2. Buat antrian kosong dan masukkan node awal (1) ke dalam antrian.
3. Buat himpunan kosong untuk melacak node yang telah dikunjungi.
4. Selama antrian tidak kosong, lakukan langkah-langkah berikut:
  - a. Ambil node pertama dari antrian. node pertama adalah 1.
  - b. Periksa apakah node tersebut telah dikunjungi sebelumnya. Jika ya, lewati langkah berikutnya.
  - c. Tandai node tersebut sebagai telah dikunjungi.
  - d. Proses node tersebut dan mencetak nilainya.
  - e. Masukkan semua tetangga yang belum dikunjungi dari node tersebut ke dalam antrian. Tetangga dari 1 adalah 2 dan 5.
5. Ulangi langkah 4 sampai antrian kosong.
6. Setelah selesai, jalur BFS akan mencakup semua node yang dapat dicapai dari node awal (1) dengan jumlah langkah terkecil.
7. Output yang dihasilkan adalah urutan kunjungan node BFS adalah :  
1, 2, 5, 3, 4, 6.

- DFS

1. Tentukan node awal yang akan digunakan sebagai titik awal pencarian. kita menggunakan node awal 1.
2. Buat himpunan kosong untuk melacak node yang telah dikunjungi.
3. Panggil fungsi DFS dengan node awal 1 sebagai argumen.
4. Dalam fungsi DFS, lakukan langkah-langkah berikut:
  - a. Tandai node saat ini (start) sebagai telah dikunjungi.
  - b. Proses node saat ini dan kita mencetak nilainya.
  - c. Untuk setiap tetangga yang belum dikunjungi dari node saat ini, panggil rekursif fungsi DFS dengan tetangga tersebut sebagai node awal.
5. Ulangi langkah 4 sampai semua node yang dapat dicapai telah dikunjungi.
6. Output yang dihasilkan adalah urutan kunjungan node DFS adalah :  
1, 2, 3, 4, 5, 6.

3. buatlah code python menggunakan algoritma BFS dan DFS dari graph yang disediakan atau boleh cari di internet, (cantumkan sumbernya) dan jelaskan dengan komentar dalam code tersebut jawab :

- DFS

```
File Edit Selection View Go Run Terminal Help
dfhy - port 11 - Visual Studio Code

dfhy.py
graph = {
    1: [2, 5],          # 1 terhubung ke 2 dan 5
    2: [1, 3, 5],       # 2 terhubung ke 1, 3, dan 5
    3: [2, 4],          # 3 terhubung ke 2 dan 4
    4: [3, 5, 6],       # 4 terhubung ke 3, 5, dan 6
    5: [1, 2, 4],       # 5 terhubung ke 1, 2, dan 4
    6: [4]              # 6 terhubung ke 4
}

def dfs(graph, start):
    """Melakukan fungsi dfs"""
    visited = set() # set untuk menyimpan node yang sudah dikunjungi

    def dfs_helper(node):
        """Membantu fungsi dfs_helper"""
        visited.add(node) # Tandai node saat ini sebagai dikunjungi
        print(node, end=" ") # Cetak node saat ini

        for neighbor in graph[node]:
            # Iterasi setiap tetangga node saat ini
            if neighbor not in visited: # Jika tetangga belum dikunjungi
                dfs_helper(neighbor) # Rekursi: lanjutkan ke tetangga yang belum dikunjungi

    dfs_helper(start) # Memulai dfs dari node start

dfs(graph, 1) # Panggil fungsi dfs dengan node start 1
```

- BFS

```
File Edit Selection View Go Run Terminal Help
dfhy - port 11 - Visual Studio Code

dfhy.py
graph = {
    1: [2, 5],          # 1 terhubung ke 2 dan 5
    2: [1, 3, 5],       # 2 terhubung ke 1, 3, dan 5
    3: [2, 4],          # 3 terhubung ke 2 dan 4
    4: [3, 5, 6],       # 4 terhubung ke 3, 5, dan 6
    5: [1, 2, 4],       # 5 terhubung ke 1, 2, dan 4
    6: [4]              # 6 terhubung ke 4
}

def bfs_terdekat(graph, mulai, tujuan):
    """Melakukan fungsi bfs_terdekat"""
    explored = [] # array untuk menyimpan node yang sudah dikunjungi
    queue = [] # array untuk menyimpan node yang akan dikunjungi

    # Parameter: mulai dan tujuan
    if mulai == tujuan:
        return "Awal adalah tujuan" # Jika awal dan tujuan sama, langsung kembalikan pesan ini

    queue.append(mulai) # Tambahkan node awal ke queue

    while queue:
        jalur = queue.pop(0) # Ambil jalur pertama dari queue
        node = jalur[1] # Ambil node terbaru dalam jalur

        if node not in explored: # Jika node tidak explored
            neighbours = graph[node] # Dapatkan semua node tetangga
            for neighbour in neighbours:
                jalur_baru = list(jalur) # Jalur baru = list jalur saat ini
                jalur_baru.append(neighbour) # Tambahkan tetangga ke jalur baru
                queue.append(jalur_baru) # Tambahkan jalur baru ke queue

            explored.append(node) # Tandai node ini sebagai explored
```