# PRAKTIKUM 31
# GRAPH
# ALGORITMA TRAVERSAL GRAPH

## A. TUJUAN PEMBELAJARAN

1. Memahami konsep dari Algoritma Traversal Graph yaitu algoritma Breadth First Search (BFS) dan Depth First Search (DFS).
2. Memahami cara mengimplementasikan algoritma traversal Graph yaitu BFS dan DFS ke dalam bahasa pemrograman Java.

## B. DASAR TEORI

### AlGORITMA TRAVERSAL GRAPH

Terdapat beberapa perbedaan Tree dan Graph dijelaskan pada tabel 31.1.

Tabel 31.1 Perbedaan Tree dan Graph

| Tree | Graph |
|---|---|
| Mempunyai root | Tidak mempunyai root, proses traversal dapat dilakukan pada sembarang vertex. |
| Terdapat path unik dari root menuju ke vertex tertentu | Graph memungkinkan memiliki cycle yang menghasilkan multiple visit ke sebuah vertex |

Terdapat dua metode Algoritma Traversal Graph yaitu :

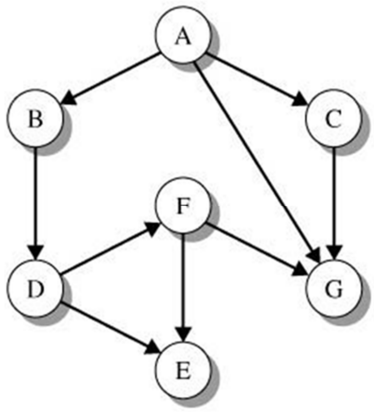
**1. Pencarian Melebar Pertama (Breadth-First Search)**

Pada metode Breadth-First Search, semua vertex pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi vertex-vertex pada level n+1. Pencarian dimulai dari vertex awal terus ke level ke-1 dari kiri ke kanan, kemudian berpindah ke level berikutnya demikian pula dari kiri ke kanan sampai ditemukannya solusi.

- Algoritma BFS
  1. Tandai semua Vertex yang terdapat pada Graph dengan warna WHITE

2. Tentukan Vertex Awal .
3. Buat sebuah Queue, masukkan vertex awal ke Queue, tandai dengan warna GRAY
4. Ambil vertex dari Queue (sebut Vertex P), tandai dengan BLACK, langsung tulis
5. jika Vertex P ≠ GOAL, diganti dengan NEIGHBORS/tetangganya (pilih Vertex yang masih berwarna WHITE), masukkan dalam QUEUE, tandai Vertex-vertex tersebut dengan warna GRAY. Lakukan pengulangan langkah 5.
6. Bila vertex P = GOAL, selesai

Sebagai contoh, kita lakukan algoritma traversal pada graph pada gambar 1.



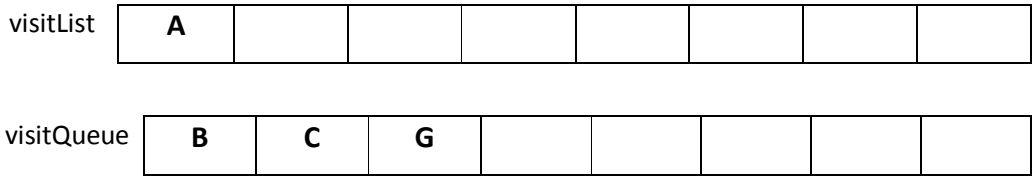

Gambar 31.1. Contoh Graph

Langkah-langkah algoritma traversal Graph menggunakan BFS.

- Terdapat dua QUEUE yaitu visitQueue untuk menyimpan Vertex-Vertex yang sedang dikunjungi dan visitList untuk Vertex-Vertex yang sudah dikunjungi.
- Warnai semua Vertex dengan WHITE dan masukan Vertex Awal yaitu A ke visitQueue, warna Vertex A dengan GRAY

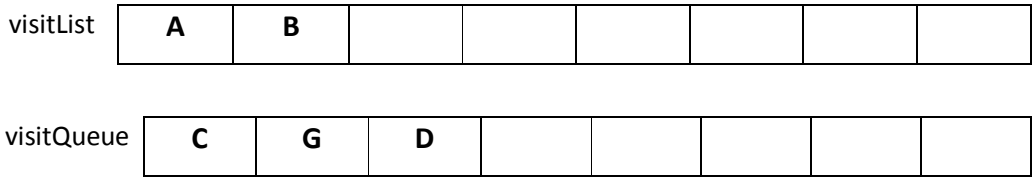

Gambar 31.2 Langkah 1 Algoritma Traversal Graph : BFS

- Ambil A dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex A yang masih berwarna WHITE yaitu B, C dan G, masukkan dalam visitQueue. Warnai Vertex B,C,G dengan GRAY.



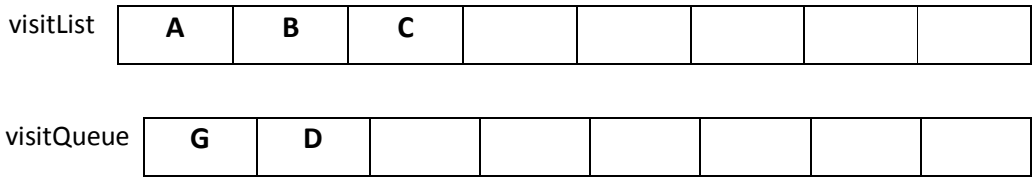

Gambar 31.3 Langkah 2 Algoritma Traversal Graph : BFS

- Ambil B dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex B yang masih berwarna WHITE yaitu D, masukkan dalam visitQueue. Warnai Vertex D dengan GRAY.



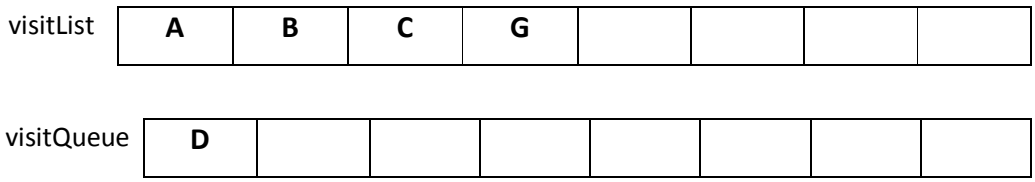

Gambar 31.4 Langkah 3 Algoritma Traversal Graph : BFS

- Ambil C dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex C yang masih berwarna WHITE. Tetangga dari Vertex C adalah G tapi berwarna GRAY, sehingga tidak ada Vertex yang dapat dimasukkan dalam visitQueue.



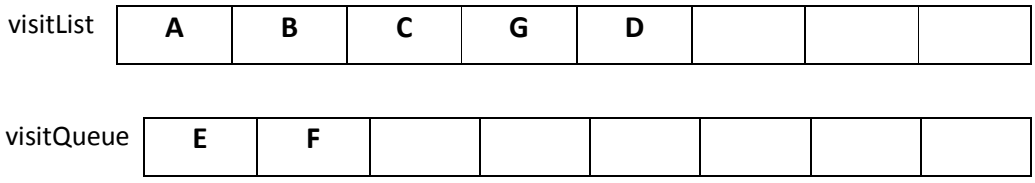

Gambar 31.5 Langkah 4 Algoritma Traversal Graph : BFS

- Ambil G dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Vertex G tidak memiliki tetangga sehingga tidak ada Vertex yang dapat dimasukkan dalam visitQueue.



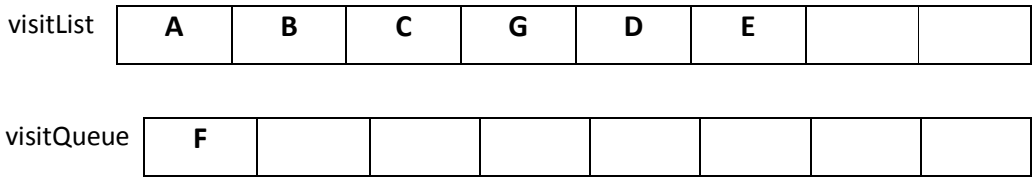

Gambar 31.6 Langkah 5 Algoritma Traversal Graph : BFS

- Ambil D dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex D yang masih berwarna WHITE yaitu E dan F, masukkan dalam visitQueue. Warnai Vertex E dan F dengan GRAY.



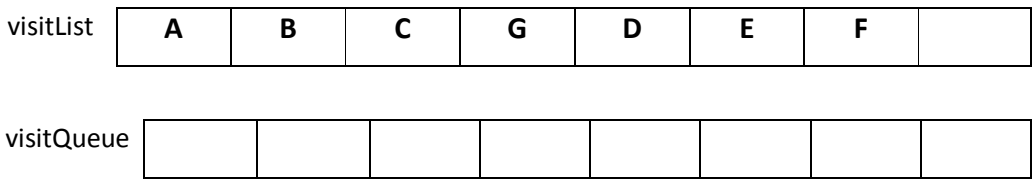

Gambar 31.7 Langkah 6 Algoritma Traversal Graph : BFS

- Ambil E dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Vertex E tidak memiliki tetangga sehingga tidak ada Vertex yang dapat dimasukkan dalam visitQueue.




Gambar 31.8 Langkah 7 Algoritma Traversal Graph : BFS

- Ambil F dari visitQueue, warnai dengan BLACK dan masukkan dalam visitList. Vertex F tidak memiliki tetangga sehingga tidak ada Vertex yang dapat dimasukkan dalam visitQueue.




Gambar 31.9 Langkah 8 Algoritma Traversal Graph : BFS

- Karena Queue kosong maka proses traversal selesai

2. Pencarian Mendalam Pertama (Depth-First Search)

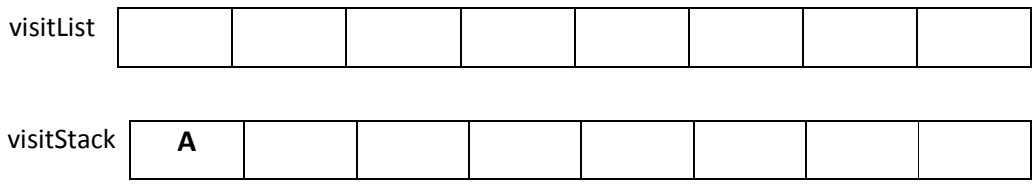
Pada Depth First Search, proses pencarian akan dilaksanakan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukaannya solusi.

Algoritma DFS

1. Tandai semua Vertex yang terdapat pada Graph dengan warna WHITE
2. Tentukan Vertex Awal .
3. Buat sebuah Stack, masukkan vertex awal ke Stack, tandai dengan warna GRAY
4. Ambil vertex dari Stack (sebut Vertex P), tandai dengan BLACK, langsung tulis
5. jika Vertex P ≠ GOAL, diganti dengan NEIGHBORS/tetangganya (pilih Vertex yang masih berwarna WHITE), masukkan dalam Stack, tandai Vertex-vertex tersebut dengan warna GRAY. Lakukan pengulangan langkah 5.
6. Bila vertex P = GOAL, selesai

Sebagai contoh, kita lakukan algoritma traversal pada graph pada gambar 31.1.
Langkah-langkah algoritma traversal Graph menggunakan DFS.

- Terdapat Stack yaitu visitStack untuk menyimpan Vertex-Vertex yang sedang dikunjungi dan Queue untuk visitList untuk Vertex-Vertex yang sudah dikunjungi.
- Warnai semua Vertex dengan WHITE dan masukan Vertex Awal yaitu A ke visitStack, ubah warna Vertex A menjadi GRAY




Gambar 31.10 Langkah 1 Algoritma Traversal Graph : DFS

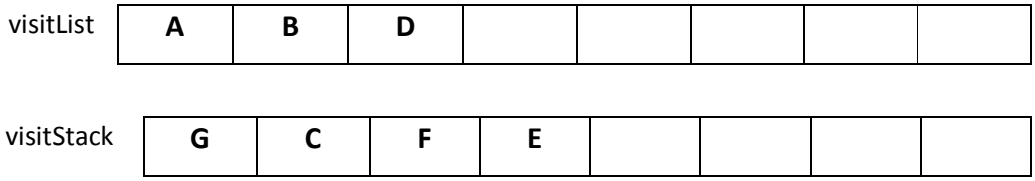
- Ambil A dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex A yang masih berwarna WHITE yaitu B, C dan G, masukkan dalam visitStack. Warnai Vertex B,C,G dengan GRAY.


Gambar 31.11 Langkah 2 Algoritma Traversal Graph : DFS

- Ambil B dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex B yang masih berwarna WHITE yaitu D, masukkan dalam visitStack. Warnai Vertex D dengan GRAY.

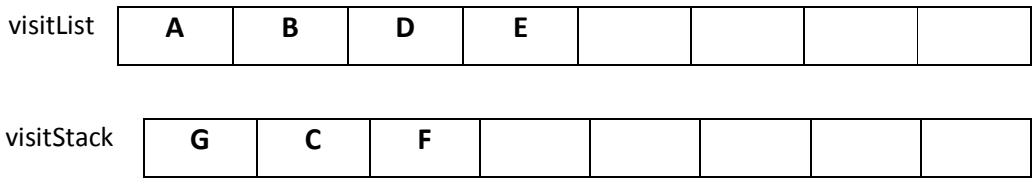

Gambar 31.12 Langkah 3 Algoritma Traversal Graph : DFS

- Ambil D dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex D yang masih berwarna WHITE yaitu E dan F.



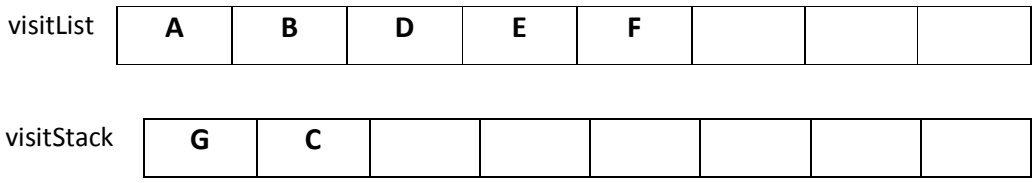

Gambar 31.13 Langkah 4 Algoritma Traversal Graph : DFS

- Ambil E dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Vertex E tidak memiliki tetangga sehingga tidak ada Vertex yang dapat dimasukkan dalam visitStack.
-



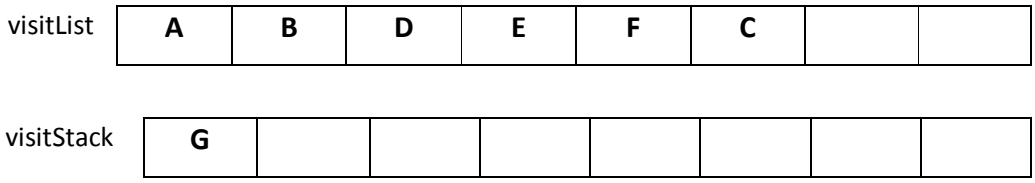

Gambar 31.14 Langkah 5 Algoritma Traversal Graph : DFS

- Ambil F dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex F yang masih berwarna WHITE. Karena tetangga dari E dan G tidak berwarna WHITE maka tidak ada yang dimasukkan ke visitStack.




Gambar 31.15 Langkah 6 Algoritma Traversal Graph : DFS

- Ambil C dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Cari tetangga dari Vertex C yang masih berwarna WHITE. Karena tetangga dari C adalah G tidak berwarna WHITE maka tidak ada yang dimasukkan ke visitStack.




Gambar 31.16 Langkah 7 Algoritma Traversal Graph : DFS

- Ambil G dari visitStack, warnai dengan BLACK dan masukkan dalam visitList. Vertex G tidak memiliki tetangga sehingga tidak ada Vertex yang dapat dimasukkan dalam visitStack.

- visitList

| A | B | D | E | F | C | G | |
|---|---|---|---|---|---|---|---|

visitStack

| | | | | | | | |
|---|---|---|---|---|---|---|---|

Gambar 31.17 Langkah 8 Algoritma Traversal Graph : DFS

- Karena Stack kosong maka proses traversal selesai

## C. TUGAS PENDAHULUAN

Buatlah review mengenai :

- Konsep dari Algoritma BFS dan DFS
- Berikan 1 contoh Graph selesaikan dengan algoritma traversal BFS dan DFS.

## D. PERCOBAAN

Pada praktikum algoritma traversal Graph menggunakan algoritma BFS dan DFS ini tambahkan pada class DiGraph.

Tabel 31.1 Method-method yang ditambahkan pada class DiGraph untuk mengimplementasikan algoritma BFS dan DFS

| Method | Kegunaan |
|---|---|
| public void colorWhite() | Memberikan warna pada setiap Vertex dengan warna WHITE |
| public VertexColor getColor(T v) | Mengembalikan warna dari Vertex |
| public VertexColor setColor(T v, VertexColor c) | Memberikan warna dari Vertex v dan mengembalikan warna sebelumnya dari Vertex tersebut. |
| public static <T> LinkedList<T> bfs( DiGraph<T> g, T sVertex) | Algoritma BFS, mengembalikan list vertex yang dikunjungi dengan algoritma BFS |
| public static <T> LinkedList<T> bfs( DiGraph<T> g, T sVertex) | Algoritma DFS, mengembalikan list vertex yang dikunjungi dengan algoritma DFS |

**Percobaan 1 : Method colorWhite() untuk memberikan warna pada setiap Vertex dengan warna WHITE**

```
public void colorWhite() {
        Iterator it = vInfo.iterator();
        while (it.hasNext()) {
            VertexInfo<T> v = (VertexInfo<T>) it.next();
            v.color = VertexColor.WHITE;
        }
    }
```

**Percobaan 2 : Method getColor() untuk mengembalikan warna dari Vertex**

```
  public VertexColor getColor(T v) {
        int index = getVInfoIndex(v);
        return vInfo.get(index).color;
    }
```

**Percobaan 3 : Method setColor() untuk memberikan warna dari Vertex v dan mengembalikan warna sebelumnya dari Vertex tersebut.**

```
    public VertexColor setColor(T v, VertexColor c) {
        int index = getVInfoIndex(v);
        VertexColor temp = vInfo.get(index).color;
        vInfo.get(index).color = c;
        return temp;
    }
```

## E. LATIHAN

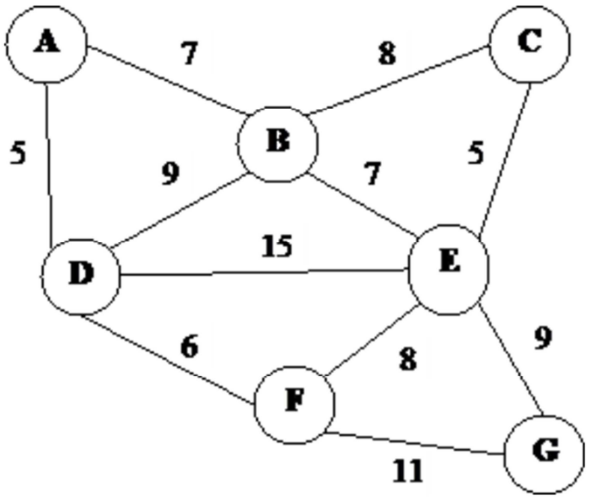
1. Implementasikan algoritma traversal Graph yaitu **BFS.**

```
public static <T> LinkedList<T> bfs( DiGraph<T> g, T sVertex)
```

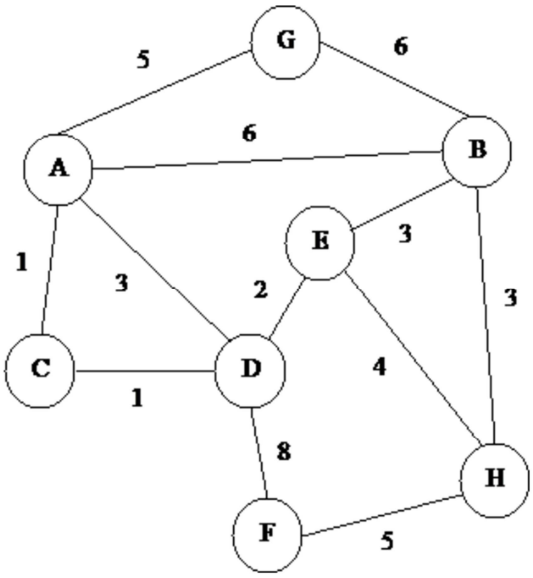
2. Implementasikan algoritma traversal Graph yaitu **DFS.**

```
public static <T> LinkedList<T> dfs( DiGraph<T> g, T sVertex)
```

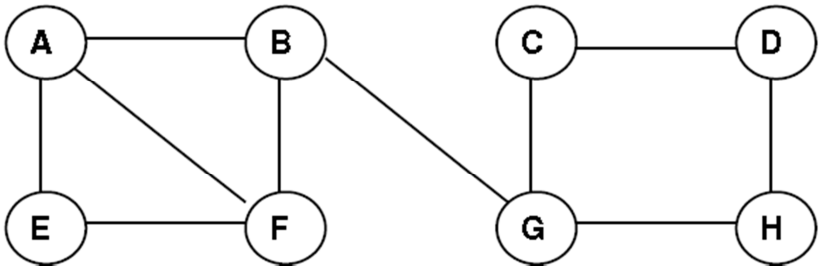
3. Selesaikan Graph yang terdapat pada gambar 31.18, gambar 31.19 dan gambar 31.20 menggunakan metode BFS dan DFS.




Gambar 31.18 Graph 1




Gambar 31.19 Graph 2




Gambar 31.20 Graph 3

## F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.