

**LAPORAN PRAKTIKUM  
MODUL 14  
“GRAPH”**



**Disusun Oleh:  
GANES GEMI PUTRA  
2311104075  
SE-07-02**

**Dosen :  
WAHYU ANDI SAPUTRA**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY  
PURWOKERTO  
2024**

**TUJUAN PRAKTIKUM :**

1. Memahami konsep graph
2. Mengimplementasikan graph dengan menggunakan pointer.

**LANDASAN TEORI :****Graf:**

- Graf adalah struktur data yang terdiri dari kumpulan simpul (vertices) dan sisi (edges) yang menghubungkan pasangan simpul.
- Graf dapat dibagi menjadi:
  - **Graf berarah (Directed Graph):** Sisi memiliki arah tertentu.
  - **Graf tak berarah (Undirected Graph):** Sisi tidak memiliki arah.
- Representasi graf:
  - **List Adjacency:** Representasi graf sebagai daftar node dan sisi yang terhubung.
  - **Matrix Adjacency:** Matriks yang menunjukkan hubungan antar simpul.

**Traversal Graf:**

- Proses mengunjungi semua simpul pada graf.
- Dua teknik traversal yang umum adalah DFS dan BFS.

**Algoritma DFS (Depth-First Search):**

- Traversal dengan menjelajahi cabang sedalam mungkin sebelum kembali ke simpul sebelumnya.
- Cocok untuk menemukan jalur, komponen terhubung, dan deteksi siklus.
- Implementasi:
  - Menggunakan rekursi atau tumpukan (*stack*).
- Kompleksitas waktu:  $O(V + E)$ , di mana  $V$  adalah jumlah simpul, dan  $E$  adalah jumlah sisi.

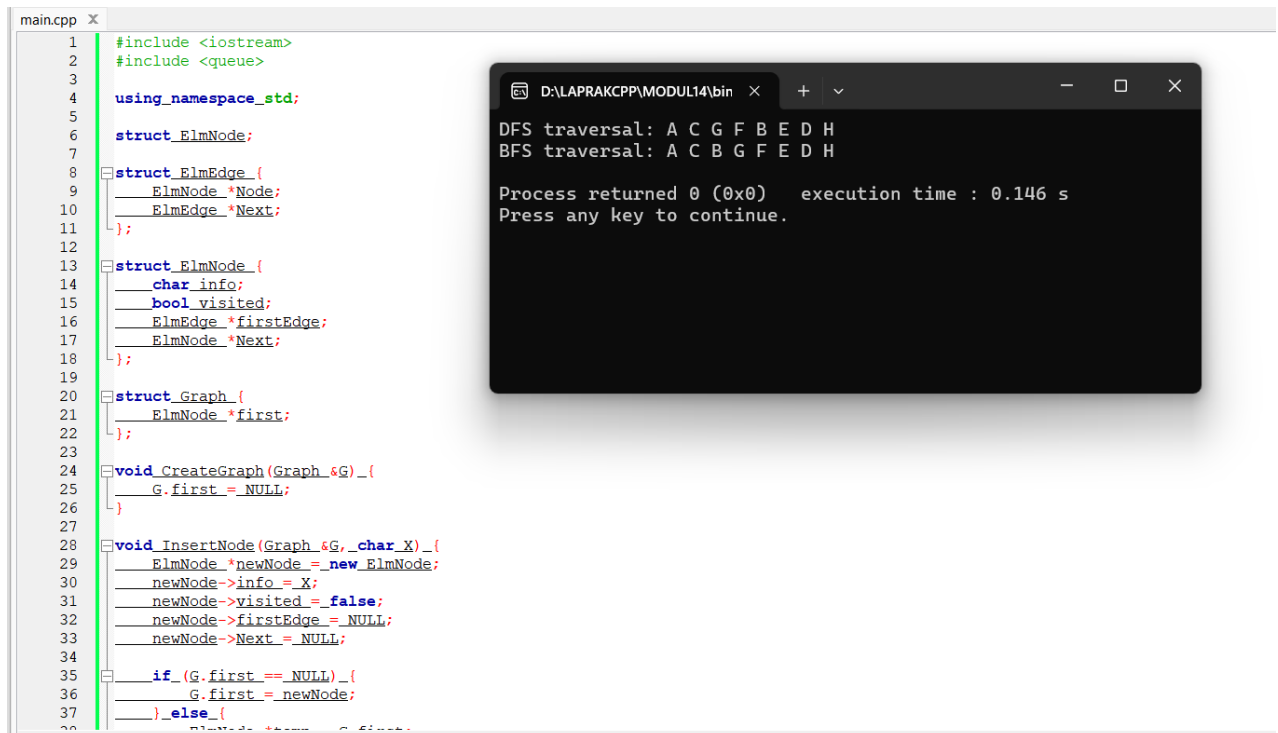
**Algoritma BFS (Breadth-First Search):**

- Traversal dengan menjelajahi simpul pada level yang sama sebelum melanjutkan ke level berikutnya.
- Cocok untuk menemukan jalur terpendek pada graf tak berbobot.
- Implementasi:
  - Menggunakan antrean (*queue*).
- Kompleksitas waktu:  $O(V + E)$ .

**Aplikasi Traversal Graf:**

- Sistem navigasi (pencarian rute).
- Pemrosesan jaringan sosial.
- Penyelesaian masalah *topological sort*.
- Deteksi konektivitas pada graf.

## GUIDED :



The screenshot shows a C++ program in a text editor and its execution output in a terminal window.

**main.cpp**

```

1  #include <iostream>
2  #include <queue>
3
4  using namespace std;
5
6  struct ElmEdge;
7
8  struct ElmEdge {
9      ElmNode *Node;
10     ElmEdge *Next;
11 };
12
13 struct ElmNode {
14     char info;
15     bool visited;
16     ElmEdge *firstEdge;
17     ElmNode *Next;
18 };
19
20 struct Graph {
21     ElmNode *first;
22 };
23
24 void CreateGraph(Graph &G) {
25     G.first = NULL;
26 }
27
28 void InsertNode(Graph &G, char X) {
29     ElmNode *newNode = new ElmNode;
30     newNode->info = X;
31     newNode->visited = false;
32     newNode->firstEdge = NULL;
33     newNode->Next = NULL;
34
35     if (G.first == NULL) {
36         G.first = newNode;
37     } else {
38         newNode->Next = G.first;
39     }
40 }

```

**Terminal Output:**

```

D:\LAPRAKCPP\MODUL14\bin x + v
DFS traversal: A C G F B E D H
BFS traversal: A C B G F E D H

Process returned 0 (0x0)   execution time : 0.146 s
Press any key to continue.

```

## PENJELASAN :

### Struktur Data

#### struct ElmNode:

- Merepresentasikan node pada graf.
- Memiliki informasi seperti:
  - char\_info: Nama atau label node (contoh: A, B, C).
  - bool\_visited: Penanda apakah node sudah dikunjungi.
  - firstEdge: Pointer ke daftar edge (sisi) yang berhubungan dengan node ini.
  - Next: Pointer ke node berikutnya dalam daftar graf.

#### struct ElmEdge:

- Merepresentasikan sisi (edge) antar node.
- Berisi:
  - Node: Node yang menjadi tujuan sisi.
  - Next: Pointer ke sisi berikutnya.

#### struct Graph:

- Struktur utama yang merepresentasikan graf.
- Berisi pointer ke node pertama dalam graf.

### Algoritma DFS dan BFS

#### DFS (Depth-First Search):

- Algoritma ini menjelajahi graf secara mendalam dengan mengunjungi seluruh cabang dari sebuah node sebelum berpindah ke node berikutnya.
- Traversal pada output: A C G F B E D H.

#### BFS (Breadth-First Search):

- Algoritma ini menjelajahi graf secara meluas dengan mengunjungi semua tetangga

dari sebuah node sebelum melanjutkan ke tingkat berikutnya.

- Traversal pada output: A C B G F E D H.

### Hasil Output

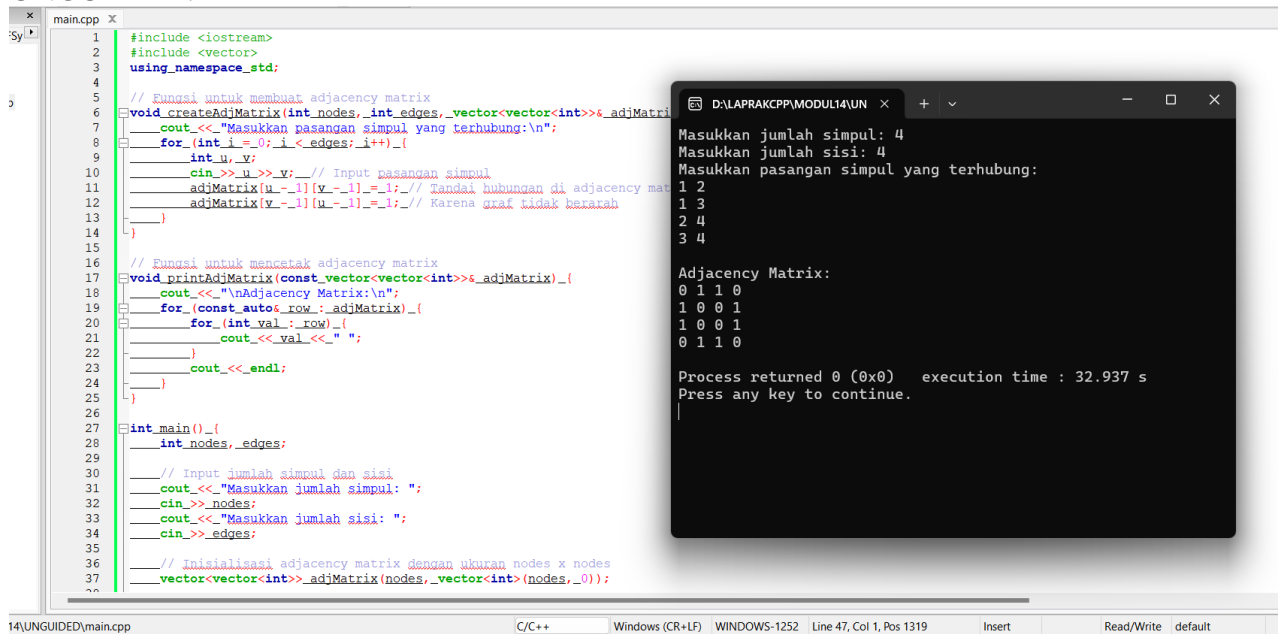
Program menampilkan urutan traversal node untuk DFS dan BFS dari graf yang dibuat.

- **DFS traversal:** Menghasilkan urutan yang lebih dalam dulu.
- **BFS traversal:** Menghasilkan urutan yang meluas dulu.

### Eksekusi

- **Execution time** menunjukkan waktu yang dibutuhkan untuk menjalankan program, yaitu 0.146 s.
- Proses berhasil diselesaikan tanpa error (Process returned 0 (0x0)).

### UNGUIDED :



```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  // Fungsi untuk membuat adjacency matrix
6  void createAdjMatrix(int nodes, int edges, vector<vector<int>>& adjMatrix) {
7      cout << "Masukkan pasangan simpul yang terhubung:\n";
8      for (int i = 0; i < edges; i++) {
9          int u, v;
10         cin >> u >> v; // Input pasangan simpul
11         adjMatrix[u-1][v-1] = 1; // Tandai hubungan di adjacency matrix
12         adjMatrix[v-1][u-1] = 1; // Karena graf tidak berarah
13     }
14 }
15
16 // Fungsi untuk mencetak adjacency matrix
17 void printAdjMatrix(const vector<vector<int>>& adjMatrix) {
18     cout << "\nAdjacency Matrix:\n";
19     for (const auto& row : adjMatrix) {
20         for (int val : row) {
21             cout << val << " ";
22         }
23         cout << endl;
24     }
25 }
26
27 int main() {
28     int nodes, edges;
29
30     // Input jumlah simpul dan sisi
31     cout << "Masukkan jumlah simpul: ";
32     cin >> nodes;
33     cout << "Masukkan jumlah sisi: ";
34     cin >> edges;
35
36     // Inisialisasi adjacency matrix dengan ukuran nodes x nodes
37     vector<vector<int>> adjMatrix(nodes, vector<int>(nodes, 0));
38 }
  
```

Output Terminal:

```

Masukkan jumlah simpul: 4
Masukkan jumlah sisi: 4
Masukkan pasangan simpul yang terhubung:
1 2
1 3
2 4
3 4

Adjacency Matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0

Process returned 0 (0x0)   execution time : 32.937 s
Press any key to continue.
  
```

### Penjelasan Kode

#### 1. Input Jumlah Simpul dan Sisi:

- Program meminta pengguna untuk memasukkan jumlah simpul dan jumlah sisi dari graf.

#### 2. Membentuk Adjacency Matrix:

- Matriks berukuran  $\text{nodes} \times \text{nodes}$  diinisialisasi dengan nilai 0.
- Input pasangan simpul digunakan untuk menandai hubungan antar simpul dengan nilai 1 di matriks.

#### 3. Output Adjacency Matrix:

- Matriks ditampilkan untuk menunjukkan hubungan antar simpul.

## KESIMPULAN

Graph merupakan struktur data yang terdiri dari simpul (vertices) dan sisi (edges) dengan dua jenis utama, yaitu graf berarah dan graf tak berarah.

Dua teknik traversal yang digunakan adalah DFS (Depth-First Search) dan BFS (Breadth-First Search), masing-masing dengan keunggulan untuk menjelajahi jalur secara mendalam dan meluas.

Implementasi algoritma DFS menggunakan tumpukan atau rekursi, sementara BFS menggunakan antrean, dengan kompleksitas waktu keduanya adalah  $O(V + E)$ .

Aplikasi traversal graf meliputi navigasi rute, pemrosesan jaringan sosial, deteksi konektivitas, dan lainnya.

Program berhasil menampilkan hasil traversal sesuai ekspektasi tanpa error, dengan waktu eksekusi yang efisien.