

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL VII

GRAPH



Disusun Oleh :

NAMA : Muhammad Fachri Auravyano Saka
NIM : 103112430180

Dosen
FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Graf (Graph) merupakan struktur data non-linear yang merepresentasikan hubungan antar objek, terdiri dari himpunan simpul (node atau vertex) dan himpunan sisi (edge) yang menghubungkan sepasang simpul tersebut. Dalam implementasi pemrograman, graf sering direpresentasikan menggunakan Adjacency List (daftar ketetanggaan), di mana setiap node memiliki senarai berantai (linked list) dinamis yang menyimpan referensi ke node-node tetangganya; metode ini dianggap lebih efisien dalam penggunaan memori dibandingkan Adjacency Matrix, terutama untuk graf yang sparse (jarang memiliki sisi). Untuk menjelajahi seluruh elemen dalam graf, terdapat dua algoritma penelusuran fundamental, yaitu Depth First Search (DFS) dan Breadth First Search (BFS). DFS melakukan penelusuran secara mendalam menggunakan prinsip Last In First Out (LIFO) yang umumnya diimplementasikan melalui rekursi atau struktur data stack, di mana algoritma akan mengunjungi satu cabang sejauh mungkin sebelum melakukan backtracking. Sebaliknya, BFS melakukan penelusuran secara melebar (per level) menggunakan prinsip First In First Out (FIFO) dengan bantuan struktur data queue, yang memastikan semua tetangga terdekat dikunjungi terlebih dahulu sebelum melanjutkan ke node pada lapisan yang lebih dalam.

B. Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

Graf.cpp

The screenshot shows a code editor with two windows. The left window displays the source code for 'graf.cpp'. The right window shows the output of the program, which includes a message box and a terminal window.

graf.cpp

```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
1 #include "graf.h"
2 #include <queue>
3 #include <stack>
4
5 void createGraph(Graph &G) {
6     G.first = NULL;
7 }
8
9 addrNode AllocateNode(infograf X) {
10    addrNode P = new ElmNode;
11    P->info = X;
12    P->visited = 0;
13    P->firstEdge = NULL;
14    P->next = NULL;
15    return P;
16 }
17
18 addrEdge AllocateEdge(addrNode N) {
19    addrEdge P = new ElmEdge;
20    P->node = N;
21    P->next = NULL;
22    return P;
23 }
24
25 void InsertNode(Graph &G, infograf X) {
26    addrNode P = AllocateNode(X);
27    P->next = G.first;
28    G.first = P;
29 }
30
31 addrNode FindNode(Graph G, infograf X) {
32    addrNode P = G.first;
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8

In 110, Col 17 | Spaces: 4 | UTE-B | CRLF | C++ | F9 | Win32 | Pretier |

The image shows two side-by-side code editors in a dark-themed IDE, likely Dev-C++. Both editors are displaying the same file, `graf.cpp`.

Top Editor Content:

```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
30
31     addrNode FindNode(Graph G, infograf X) {
32         addrNode P = G.first;
33         while (P != NULL) {
34             if (P->info == X)
35                 return P;
36             P = P->next;
37         }
38         return NULL;
39     }
40
41     void ConnectNode(Graph &G, infograf A, infograf B) {
42         addrNode N1 = FindNode(G, A);
43         addrNode N2 = FindNode(G, B);
44
45         if (N1 == NULL || N2 == NULL) {
46             cout << "Node tidak ditemukan\n";
47             return;
48         }
49
50         // Buat Edge dari N1 ke N2
51         addrEdge E1 = AllocateEdge(N2);
52         E1->next = N1->firstEdge;
53         N1->firstEdge = E1;
54
55         // Karena undirected -> buat Edge balik
56         addrEdge E2 = AllocateEdge(N1);
57         E2->next = N2->firstEdge;
58         N2->firstEdge = E2;
59     }
60 }
```

Bottom Editor Content:

```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
61
62     void PrintInfoGraph(Graph G) {
63         addrNode P = G.first;
64         while (P != NULL) {
65             cout << P->info << " -> ";
66             addrEdge E = P->firstEdge;
67             while (E != NULL) {
68                 cout << E->node->info << " ";
69                 E = E->next;
70             }
71             cout << endl;
72             P = P->next;
73         }
74
75     void ResetVisited(Graph &G) {
76         addrNode P = G.first;
77         while (P != NULL) {
78             P->visited = 0;
79             P = P->next;
80         }
81
82     void PrintDFS(Graph G, addrNode N) {
83         if (N == NULL)
84             return;
85         N->visited = 1;
86         cout << N->info << " ";
87
88         addrEdge E = N->firstEdge;
89         while (E != NULL)
90         {
91             if (E->node->visited == 0)
92                 PrintDFS(G, E->node);
93         }
94     }
95 }
```

Output Panes:

Both output panes show the same results:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8

Ln 110, Col 17 Spaces:4 UITE8 CRLE [] C++

```
Go Run ... ← → Q. Code
```

graf.cpp

```
Modul 14 > Guided > graf.cpp > PrintBFS(Graph, addrNode)
```

```
83 void PrintDFS(Graph G, addrNode N) {
84
85     addrEdge E = N->firstEdge;
86     while (E != NULL) {
87
88         if (E->node->visited == 0) {
89             PrintDFS(G, E->node);
90         }
91         E = E->next;
92     }
93
94     void PrintBFS(Graph G, addrNode N) {
95
96         if (N == NULL)
97             return;
98
99         queue<addrNode> Q;
100        Q.push(N);
101
102        while (!Q.empty()) {
103            addrNode curr = Q.front();
104            Q.pop();
105            if (curr->visited == 0) {
106                curr->visited = 1;
107                cout << curr->info << " ";
108
109                addrEdge E = curr->firstEdge;
110                while (E != NULL) {
111                    if (E->node->visited == 0) {
112                        Q.push(E->node);
113                    }
114                    E = E->next;
115                }
116
117            }
118
119        }
120    }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 characters | Plain text | 100% | Wind | UTF-8

```
File Edit View Aa 🔍 ⚙️
```

Graf.h

```
Modul 14 > Guided > Graf.h
```

```
105
106    while (!Q.empty()) {
107        addrNode curr = Q.front();
108        Q.pop();
109        if (curr->visited == 0) {
110            curr->visited = 1;
111            cout << curr->info << " ";
112
113            addrEdge E = curr->firstEdge;
114            while (E != NULL) {
115                if (E->node->visited == 0) {
116                    Q.push(E->node);
117                }
118                E = E->next;
119            }
120        }
121    }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 characters | Plain text | 100% | Wind | UTF-8

```
File Edit View Aa 🔍 ⚙️
```

The screenshot shows a C++ development environment with two code files and their corresponding outputs.

graf.h

```
Modul 14 > Guided > C graf.h > FindNode(Graph, infograf)
1 #ifndef GRAF_H INCLUDED
2 #define GRAF_H INCLUDED
3
4 #include <iostream>
5 using namespace std;
6
7 typedef char infograf;
8
9 struct ElmNode;
10 struct ElmEdge;
11
12 typedef ElmNode* addrNode;
13 typedef ElmEdge* addrEdge;
14
15 struct ElmNode {
16     infograf info;
17     int visited;
18     addrEdge firstEdge;
19     addrNode next;
20 };
21
22 struct ElmEdge {
23     addrNode node;
24     addrEdge next;
25 };
26
27 struct Graph {
28     addrNode first;
29 };
30
31 // PRIMITIF GRAPH
32 void createGraph(Graph &G);
```

Main.cpp

```
33
34     addrNode AllocateNode(infograf X);
35     addrEdge AllocateEdge(addrNode N);
36
37     void InsertNode(Graph &G, infograf X);
38     addrNode FindNode(Graph G, infograf X);
39
40     void ConnectNode(Graph &G, infograf A, infograf B);
41
42     void PrintGraph(Graph G);
43
44     // Traversal
45     void ResetVisited(Graph &G);
46     void PrintDFS(Graph G, addrNode N);
47     void PrintBFS(Graph G, addrNode N);
48
49 #endif
```

Output Window:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

Bottom Status Bar:

Ln 2, Col 20 | 57 character | Plain t | 100% | Wind | UTF-8

Ln 37, Col 40 | Spaces:4 | UTF-8 | CR/LF | {} C++ |

Ln 2, Col 20 | 57 character | Plain t | 100% | Wind | UTF-8

Ln 37, Col 40 | Spaces:4 | UTF-8 | CR/LF | {} C++ |

Main.cpp

The image shows a code editor interface with a terminal window overlay. The code editor displays the file `main.cpp` containing C++ code for a graph traversal program. The terminal window shows the output of the program, which includes the names and NIM numbers of the students.

```
View Go ... ← → ⌂ Code
🕒 main.cpp ×

Modul 14 > Guided > ⌂ main.cpp > ⌂ main()

1 #include "graf.h"
2 #include "graf.cpp"
3 #include <iostream>
4 using namespace std;
5
6 int main() {
7     Graph G;
8     createGraph(G);
9
10    // Tambah Node
11    InsertNode(G, 'A');
12    InsertNode(G, 'B');
13    InsertNode(G, 'C');
14    InsertNode(G, 'D');
15    InsertNode(G, 'E');
16
17    // Hubungkan Node (graph tidak berarah)
18    ConnectNode(G, 'A', 'B');
19    ConnectNode(G, 'A', 'C');
20    ConnectNode(G, 'B', 'D');
21    ConnectNode(G, 'C', 'E');
22
23    cout << "___ Struktur Graph ___\n";
24    PrintInfoGraph(G);
25
26    cout << "\n___ DFS dari Node A ___\n";
27    ResetVisited(G);
28    PrintDFS(G, FindNode(G, 'A'));
29
30    cout << "\n\n___ BFS dari Node B ___\n";
31    ResetVisited(G);
32    PrintBFS(G, FindNode(G, 'A'));
33
34    cout << endl;
35    return 0;
36 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Screenshots Output

```
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided"; if ($?) { g++ main.cpp -o main };
if ($?) { .\main }
*** Struktur Graph ***
E -> C
D -> B
C -> E A
B -> D A
A -> C B

*** DFS dari Node A ***
A C E B D

*** BFS dari Node B ***
A C B E D
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided> 
```

Deskripsi:

Program ini adalah implementasi struktur data Graph tak berarah (Undirected Graph) menggunakan bahasa C++ dengan representasi daftar ketetanggaan (adjacency list). Program ini mendefinisikan tipe data dinamis untuk Node (simpul) dan Edge (sisi/jalur) menggunakan pointer, di mana setiap node memiliki daftar koneksi ke node lainnya.

Secara fungsional, program ini memungkinkan pengguna untuk membuat graph, menambahkan node (dalam contoh ini node A, B, C, D, dan E), serta menghubungkan node-node tersebut. Setelah struktur graph terbentuk, program menampilkan representasi visual konesinya ke layar. Selain itu, program ini juga mendemonstrasikan dua algoritma penelusuran (traversal) utama: DFS (Depth First Search) yang menelusuri graph secara mendalam menggunakan rekursi, dan BFS (Breadth First Search) yang menelusuri secara melebar menggunakan bantuan struktur data queue.

C. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

Main.cpp

The screenshot shows a code editor interface with two tabs: "main.cpp" and "graph.cpp". The "main.cpp" tab is active, displaying the following C++ code:

```
3
4 int main() {
5     Graph G;
6     CreateGraph(G);
7
8     // Tambah node sesuai Gambar pada soal
9     InsertNode(G, 'A');
10    InsertNode(G, 'B');
11    InsertNode(G, 'C');
12    InsertNode(G, 'D');
13    InsertNode(G, 'E');
14    InsertNode(G, 'F');
15    InsertNode(G, 'G');
16    InsertNode(G, 'H');
17
18    // Hubungkan node sesuai Gambar pada soal
19    adrNode A = FindNode(G, 'A');
20    adrNode B = FindNode(G, 'B');
21    adrNode C = FindNode(G, 'C');
22    adrNode D = FindNode(G, 'D');
23    adrNode E = FindNode(G, 'E');
24    adrNode F = FindNode(G, 'F');
25    adrNode GNode = FindNode(G, 'G');
26    adrNode H = FindNode(G, 'H');
27
28    // Koneksi sesuai gambar
29    ConnectNode(A, B);
30    ConnectNode(A, C);
31    ConnectNode(B, D);
32    ConnectNode(B, E);
33    ConnectNode(C, F);
34    ConnectNode(C, GNode);
```

The "graph.cpp" tab is partially visible at the top. To the right of the code editor, there is a terminal window showing the following output:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180
```

The terminal also displays status information at the bottom: Ln 2, Col 20 | 57 characters | Plain text | 100% | Wind | UTF-8.

main.cpp

```
27 // Koneksi sesuai gambar
28 ConnectNode(A, B);
29 ConnectNode(A, C);
30 ConnectNode(B, D);
31 ConnectNode(B, E);
32 ConnectNode(C, F);
33 ConnectNode(C, GNode);
34 ConnectNode(D, H);
35 ConnectNode(E, H);
36 ConnectNode(F, H);
37 ConnectNode(GNode, H);
38
39 // Tampilkan struktur graph
40 cout << "==== Struktur Graph ===" << endl;
41 PrintInfoGraph(G);
42
43 // Soal 2: DFS
44 cout << "\n==== DFS dari Node A ===" << endl;
45 ResetVisited(G);
46 PrintDFS(G, A);
47 cout << endl;
48
49 // Soal 3: BFS
50 cout << "\n==== BFS dari Node A ===" << endl;
51 ResetVisited(G);
52 PrintBFS(G, A);
53 cout << endl;
54
55
56 return 0;
57 }
```

graph.cpp

```
1 #include "graph.h"
2 #include <queue>
3
4 void CreateGraph(Graph &G) {
5     G.first = NULL;
6 }
7
8 adrNode AllocateNode(infoGraph X) {
9     adrNode P = new ElmNode;
10    P->info = X;
11    P->visited = 0;
12    P->firstEdge = NULL;
13    P->next = NULL;
14    return P;
15 }
16
17 adrEdge AllocateEdge(adrNode N) {
18     adrEdge P = new ElmEdge;
19     P->node = N;
20     P->next = NULL;
21     return P;
22 }
23
24 void InsertNode(Graph &G, infoGraph X) {
25     adrNode P = AllocateNode(X);
26     P->next = G.first;
27     G.first = P;
28 }
29
30 adrNode FindNode(Graph G, infoGraph X) {
31     adrNode P = G.first;
32     while (P != NULL) {
```

graph.cpp

```
1 #include "graph.h"
2 #include <queue>
3
4 void CreateGraph(Graph &G) {
5     G.first = NULL;
6 }
7
8 adrNode AllocateNode(infoGraph X) {
9     adrNode P = new ElmNode;
10    P->info = X;
11    P->visited = 0;
12    P->firstEdge = NULL;
13    P->next = NULL;
14    return P;
15 }
16
17 adrEdge AllocateEdge(adrNode N) {
18     adrEdge P = new ElmEdge;
19     P->node = N;
20     P->next = NULL;
21     return P;
22 }
23
24 void InsertNode(Graph &G, infoGraph X) {
25     adrNode P = AllocateNode(X);
26     P->next = G.first;
27     G.first = P;
28 }
29
30 adrNode FindNode(Graph G, infoGraph X) {
31     adrNode P = G.first;
32     while (P != NULL) {
```

Go Run ... ← → 🔍 Code ⌂ ⌂ ⌂

main.cpp graph.cpp

Modul 14 > Unguided > graph.cpp > ...

```
30 adrNode FindNode(Graph G, infoGraph X) {
31     adrNode P = G.first;
32     while (P != NULL) {
33         if (P->info == X)
34             return P;
35         P = P->next;
36     }
37     return NULL;
38 }

39 void ConnectNode(adrNode N1, adrNode N2) {
40     if (N1 == NULL || N2 == NULL) {
41         return;
42     }

43     // Edge dari N1 ke N2
44     adrEdge E1 = AllocateEdge(N2);
45     E1->next = N1->firstEdge;
46     N1->firstEdge = E1;

47     // Edge dari N2 ke N1 (undirected)
48     adrEdge E2 = AllocateEdge(N1);
49     E2->next = N2->firstEdge;
50     N2->firstEdge = E2;
51 }

52 void PrintInfoGraph(Graph G) {
53     adrNode P = G.first;
54     while (P != NULL) {
55         cout << P->info << " -> ";
56         adrEdge E = P->firstEdge;
57         while (E != NULL) {
58             cout << E->node->info << " ";
59             E = E->next;
60         }
61         cout << endl;
62         P = P->next;
63     }
64 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 character | Plain text | 100% | Wind | UTF-8

Go Run ... ← → 🔍 Code ⌂ ⌂ ⌂

main.cpp graph.cpp

Modul 14 > Unguided > graph.cpp > ...

```
56 void PrintInfoGraph(Graph G) {
57     while (P != NULL) {
58         cout << P->info << " -> ";
59         adrEdge E = P->firstEdge;
60         while (E != NULL) {
61             cout << E->node->info << " ";
62             E = E->next;
63         }
64         cout << endl;
65         P = P->next;
66     }
67 }

68 void ResetVisited(Graph &G) {
69     adrNode P = G.first;
70     while (P != NULL) {
71         P->visited = 0;
72         P = P->next;
73     }
74 }

75 // Soal 2: Prosedur PrintDFS
76 void PrintDFS(Graph G, adrNode N) {
77     if (N == NULL)
78         return;

79     N->visited = 1;
80     cout << N->info << " ";

81     adrEdge E = N->firstEdge;
82     while (E != NULL) {
83         if (E->node->visited == 0) {
84             cout << E->node->info << " ";
85             E->node->visited = 1;
86             PrintDFS(G, E->node);
87         }
88     }
89 }
```

NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM : 103112430180

Ln 2, Col 20 | 57 character | Plain text | 100% | Wind | UTF-8

The screenshot shows a C++ development environment with two code editors and a terminal window.

Code Editors:

- Top Editor (graph.cpp):**

```
Modul 14 > Unguided > graph.cpp ...
79 void PrintDFS(Graph G, adrNode N) {
80     adrEdge E = N->firstEdge;
81     while (E != NULL) {
82         if (E->node->visited == 0) {
83             PrintDFS(G, E->node);
84         }
85         E = E->next;
86     }
87
88 // Soal 3: Prosedur PrintBFS
89 void PrintBFS(Graph G, adrNode N) {
90     if (N == NULL)
91         return;
92
93     queue<adrNode> Q;
94     Q.push(N);
95
96     while (!Q.empty()) {
97         adrNode curr = Q.front();
98         Q.pop();
99
100        if (curr->visited == 0) {
101            curr->visited = 1;
102            cout << curr->info << " ";
103
104            adrEdge E = curr->firstEdge;
105            while (E != NULL) {
106                if (E->node->visited == 0) {
107                    Q.push(E->node);
108                }
109                E = E->next;
110            }
111        }
112    }
113
114
115
116 }
```
- Bottom Editor (graph.h):**

```
102
103     while (!Q.empty()) {
104         adrNode curr = Q.front();
105         Q.pop();
106
107         if (curr->visited == 0) {
108             curr->visited = 1;
109             cout << curr->info << " ";
110
111             adrEdge E = curr->firstEdge;
112             while (E != NULL) {
113                 if (E->node->visited == 0) {
114                     Q.push(E->node);
115                 }
116                 E = E->next;
117             }
118         }
119     }
120 }
```

Terminal Window:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180
```

graph.h

The screenshot shows a code editor interface with two windows. The left window contains the contents of `graph.h` and `main.cpp`. The right window shows a terminal or output pane.

graph.h:

```
Modul 14 > Unguided > C graph.h > ...
1 #ifndef GRAPH_H_INCLUDED
2 #define GRAPH_H_INCLUDED
3
4 #include <iostream>
5 using namespace std;
6
7 typedef char infoGraph;
8
9 struct ElmNode;
10 struct ElmEdge;
11
12 typedef ElmNode* adrNode;
13 typedef ElmEdge* adrEdge;
14
15 struct ElmNode {
16     infoGraph info;
17     int visited;
18     adrEdge firstEdge;
19     adrNode next;
20 };
21
22 struct ElmEdge {
23     adrNode node;
24     adrEdge next;
25 };
26
27 struct Graph {
28     adrNode first;
29 };
30
31 // Primitif sesuai soal latihan 1
32 void CreateGraph(Graph &G);
```

main.cpp:

```
Modul 14 > Unguided > C graph.h > ...
1 #ifndef GRAPH_H_INCLUDED
2
3 struct Graph {
4     adrNode first;
5 };
6
7 // Primitif sesuai soal latihan 1
8 void CreateGraph(Graph &G);
9 void InsertNode(Graph &G, infoGraph X);
10 void ConnectNode(adrNode N1, adrNode N2);
11 void PrintInfoGraph(Graph G);
12
13 // Primitif tambahan untuk mendukung
14 adrNode AllocateNode(infoGraph X);
15 adrEdge AllocateEdge(adrNode N);
16 adrNode FindNode(Graph G, infoGraph X);
17 void ResetVisited(Graph &G);
18
19 // Soal 2 dan 3
20 void PrintDFS(Graph G, adrNode N);
21 void PrintBFS(Graph G, adrNode N);
22
23#endif
```

Output Terminal:

```
NAMA : MUHAMMAD FACHRI AURAVYANO SAKA
NIM  : 103112430180

Ln 2, Col 20 | 57 character Plain text 100% Wind UTF-8
```

Screenshot Output

```

PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Guided> cd "e:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Unguided\" ; if ($?) { g++ main.cpp -o main }
; If ($?) { .\main }
*** Struktur Graph ***
H -> G F E D
G -> H C
F -> H C
E -> H B
D -> H B
C -> G F A
B -> E D A
A -> C B

*** DFS dari Node A ===
A C G H F E B D

*** BFS dari Node A ===
A C B G F E D H
PS E:\KULIAH\SEMESTER 3\Struktur Data\Code\Modul 14\Unguided>

```

Deskripsi:

Kode ini merupakan implementasi lengkap struktur data Graph Tak Berarah (Undirected Graph) menggunakan bahasa C++ dengan metode representasi daftar ketetanggaan (adjacency list), di mana program utama (main.cpp) membangun sebuah topologi jaringan spesifik yang terdiri dari 8 node (A hingga H). Dalam skenario ini, Node A berfungsi sebagai titik awal yang bercabang ke B dan C, lalu cabang-cabang tersebut terus meluas hingga akhirnya menyatu kembali (konvergen) di satu titik temu yaitu Node H. Selain memvisualisasikan struktur koneksi antar node tersebut ke layar, program ini bertujuan mendemonstrasikan perbandingan logika penelusuran graph dengan menjalankan algoritma Depth First Search (DFS) yang menelusuri kedalaman cabang hingga tuntas terlebih dahulu, dan Breadth First Search (BFS) yang menelusuri node secara melebar per level, yang keduanya dieksekusi berturut-turut dimulai dari sumber yang sama yaitu Node A.

D. Kesimpulan

Berdasarkan implementasi program yang telah dilakukan, dapat disimpulkan bahwa representasi graf menggunakan adjacency list dengan pointer memberikan fleksibilitas tinggi dalam memanipulasi topologi jaringan yang kompleks, seperti penambahan node dan koneksi antar node secara dinamis tanpa batasan ukuran array yang statis. Uji coba pada topologi graf yang memiliki titik percabangan dan titik temu (konvergensi) berhasil memperlihatkan perbedaan karakteristik yang signifikan antara metode DFS dan BFS; DFS terbukti memprioritaskan kedalaman dengan menelusuri satu jalur hingga node tujuan akhir sebelum beralih ke jalur lain, sedangkan BFS memprioritaskan keluasan dengan menyapu bersih semua node pada level yang sama sebelum turun ke level berikutnya. Hal ini menegaskan bahwa pemilihan algoritma traversal sangat bergantung pada tujuan spesifik aplikasi: DFS lebih sesuai untuk simulasi pencarian jalur solusi dalam labirin atau teka-teki, sementara BFS lebih efektif untuk mencari tetangga terdekat atau jalur terpendek dalam graf tak berbobot.

E. Referensi

- Muñoz, D. F. (2024, June). A C++ library for fast simulation of queues and some experimental results. In *AIP Conference Proceedings* (Vol. 3094, No. 1, p. 110002). AIP Publishing LLC.

Goponenko, A., & Carroll, S. (2019). A C++ implementation of a lock-free priority queue based on Multi-Dimensional Linked List. *Link: https://www.researchgate.net/publication/337020321_A_C_Implementation_of_a_Lock-Free_Priority_Queue_Based_on_Multi-Dimensional_Linked_List.*

Malik, D. S. (2010). *Data structures using C++*. USA.