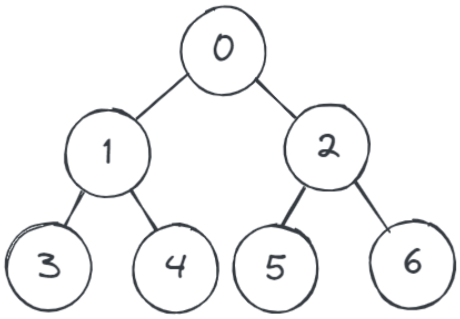
Practical – 9

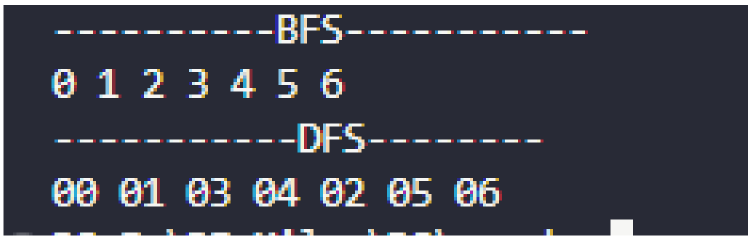
AIM: Implementation of Graph and Searching (DFS and BFS).

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
class graph
{
public:
    int vertices;
    vector<bool> visitedListdfs;
    vector<vector<int>> adj;
    graph(int size)
    {
        // must assign size ..
        adj.resize(size);
        visitedListdfs.resize(size);
        vertices = size;
    }

    int BFS(int x);
    int DFS(int x);
    void addedge(int a, int b);
};


void graph::addedge(int a, int b)
{
    adj[a].push_back(b);
    adj[b].push_back(a);
}

int graph::BFS(int x)
{
    vector<bool> visitedList;
    visitedList.resize(vertices, false);
    queue<int> queue;
    // get queue front and then push thier sub child in queue untill queue is not
empty..
    queue.push(x);
    visitedList[x] = true;

    while (queue.size() > 0)
    {
        int x = queue.front();
        cout << x << " ";
        queue.pop();

        for (auto node : adj[x])
        {
            if (visitedList[node] == false)
            {
                visitedList[node] = true;
                queue.push(node);
            }
        }
    }
    cout << endl;
}

int graph::DFS(int x)
{
    cout << visitedListdfs[x];
    visitedListdfs[x] = true;
    cout << x << " ";

    for (auto i : adj[x])
    {

        if (!visitedListdfs[i])
        {
            DFS(i);
        }
    }
}


int main()
{
    graph obj(8);
    cout << "Graph" << endl;
    cout << "Graph 1" << endl;
    obj.addedge(0, 1);
    obj.addedge(0, 2);
    obj.addedge(1, 3);
    obj.addedge(1, 4);
    obj.addedge(2, 5);
    obj.addedge(2, 6);

    cout << "----------BFS-----------" << endl;
    obj.BFS(0);
    cout << "-----------DFS--------" << endl;
    obj.DFS(0);
}
```

OUTPUT

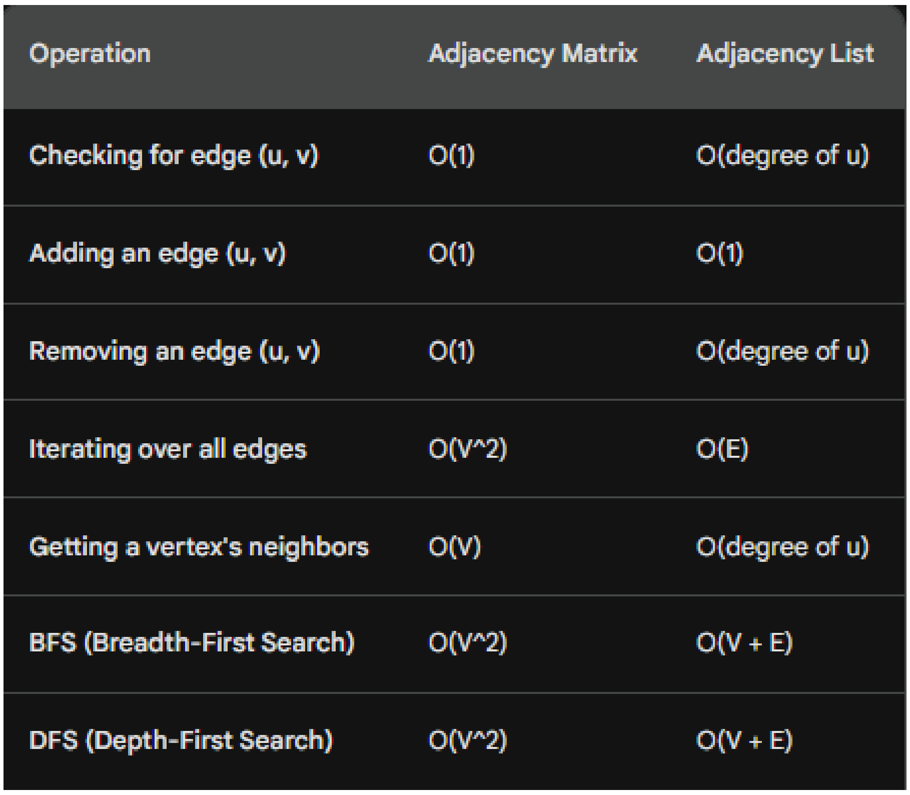
visualize





```
----------BFS-----------
0 1 2 3 4 5 6
-----------DFS--------
00 01 03 04 02 05 06
```

Time analysis



Operation	Adjacency Matrix	Adjacency List
Checking for edge (u, v)	O(1)	O(degree of u)
Adding an edge (u, v)	O(1)	O(1)
Removing an edge (u, v)	O(1)	O(degree of u)
Iterating over all edges	O(V^2)	O(E)
Getting a vertex's neighbors	O(V)	O(degree of u)
BFS (Breadth-First Search)	O(V^2)	O(V + E)
DFS (Depth-First Search)	O(V^2)	O(V + E)

- V represents the number of vertices (nodes) in the graph.
- E represents the number of edges in the graph.

➢ **Use adjacency matrix:**
- When the graph is dense (many edges).
- When you need to quickly check for edge existence or add/remove edges frequently.

➢ **Use adjacency list:**
- When the graph is sparse (few edges).
- When you need to iterate over edges or get a vertex's neighbors frequently.
- When you need to perform BFS or DFS.

Application

➢ Google maps uses graphs for building transportation systems, where intersection of two(or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge, thus their navigation system is based on the algorithm to calculate the shortest path between two vertices.

➢ In Facebook, users are considered to be the vertices and if they are friends then there is an edge running between them. Facebook's Friend suggestion algorithm uses graph theory. Facebook is an example of undirected graph.

- In World Wide Web, web pages are considered to be the vertices. There is an edge from a page u to other page v if there is a link of page v on page u. This is an example of Directed graph. It was the basic idea behind Google Page Ranking Algorithm.
- In Operating System, we come across the Resource Allocation Graph where each process and resources are considered to be vertices. Edges are drawn from resources to the allocated process, or from requesting process to the requested resource. If this leads to any formation of a cycle then a deadlock will occur.
- In mapping system we use graph. It is useful to find out which is an excellent place from the location as well as your nearby location. In GPS we also use graphs.
- Facebook uses graphs. Using graphs suggests mutual friends. it shows a list of the f following pages, friends, and contact list.
- Microsoft Excel uses DAG means Directed Acyclic Graphs.
- In the Dijkstra algorithm, we use a graph. we find the smallest path between two or many nodes