Bansilal Ramnath Agarwal Charitable Trust's
VISHWAKARMA INSTITUTE OF TECHNOLOGY – PUNE

ARTIFICIAL INTELLIGENCE

Division	A
Batch	2
GR-no	12311493
Roll no	54
Name	Atharva Kangralkar

Assignment 2:

1.A Implement BFS and DFS algorithm on Graph (UsingArray)

1.B Implement BFS and DFS algorithm on Tree (Using linked list)

1. A Implement BFS and DFS algorithm on Graph(Using Array)

Description:-

BFS (Breadth-First Search): This algorithm explores a graph level by level, starting from the source node. It uses a queue data structure to keep track of nodes to visit. BFS guarantees the shortest path in an unweighted graph. It visits nodes in layers, from the source node outward.
DFS (Depth-First Search): DFS explores a graph by traversing as far as possible along each branch before backtracking. It uses a stack data structure (or recursion) to manage the nodes. DFS can be useful for pathfinding or topological sorting. It does not guarantee the shortest path.

- **Code:**

```
#include<stdio.h>

int g[100][100],v[100],q[100],f=0,r=-1,n;

void bfs(int s){
   int i;
   q[++r]=s,v[s]=1;
   printf("BFS: ");
   while(f<=r){
      s=q[f++];
      printf("%d ",s);
      for(i=0;i<n;i++)
         if(g[s][i]&&!v[i])
            q[++r]=i,v[i]=1;
   }
}

void dfs(int s){
   int i;
   printf("%d ",s),v[s]=1;
   for(i=0;i<n;i++)
      if(g[s][i]&&!v[i])
```

```
            dfs(i);
}

int main(){
    int e,i,a,b,s;
    printf("Enter number of nodes: ");
    scanf("%d",&n);
    printf("Enter number of edges: ");
    scanf("%d",&e);
    for(i=0;i<e;i++){
        printf("Enter edge %d: ",i+1);
        scanf("%d%d",&a,&b);
        g[a][b]=g[b][a]=1;
    }
    printf("\nAdjacency Matrix:\n");
    for(i=0;i<n;i++){
        for(int j=0;j<n;j++)
            printf("%d ",g[i][j]);
        printf("\n");
    }
    printf("\nEnter start node for traversal: ");
    scanf("%d",&s);
    bfs(s);
    for(i=0;i<n;i++)v[i]=0;
    printf("\nDFS: ");
    dfs(s);
}
```

Screenshots/Output:

```
Enter number of nodes: 5
Enter number of edges: 3
Enter edge 1: 0 1
Enter edge 2: 0 3
Enter edge 3: 1 3

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

Enter start node for traversal: 0
BFS: 0 1 3
DFS: 0 1 3
```