Assignment No 1

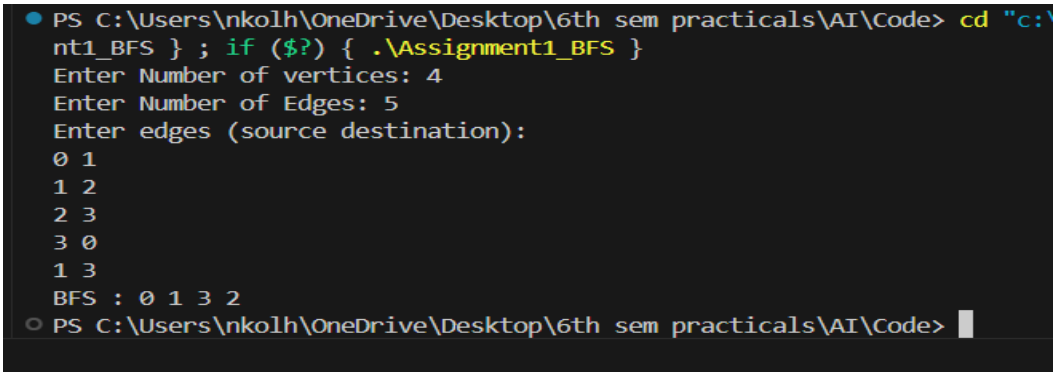
Code

BFS :

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
int main(){

   int v=0, e=0;
   cout << "Enter Number of vertices: ";
   cin >> v;
   cout << "Enter Number of Edges: ";
   cin >> e;

   vector<vector<int>> adj(v);

   cout << "Enter edges (source destination):\n";
   for (int i = 0; i < e; i++) {
      int s, d;
      cin >> s >> d;
      adj[s].push_back(d);
      adj[d].push_back(s);
   }

   vector<bool> visited(v,false);

   queue<int> q;
   q.push(0);
   visited[0]=true;

   cout<<"BFS : ";

   while(!q.empty()){
      int curr=q.front();
      q.pop();
      cout<<curr<<" ";

      for(auto i:adj[curr]){
         if(!visited[i]){
```

```
            visited[i]=true;
            q.push(i);
        }
    }

  }

  return 0;
}
```

Output :-



DFS :

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
void DFS(int node, vector<vector<int>> &adj, vector<bool> &visited) {
  cout << node << " ";
  visited[node] = true;

  for (int neighbor : adj[node]) {
    if (!visited[neighbor]) {
      DFS(neighbor, adj, visited);
    }
  }
}
int main(){

  int v=0, e=0;
```

```
    cout << "Enter Number of vertices: ";
    cin >> v;
    cout << "Enter Number of Edges: ";
    cin >> e;

    vector<vector<int>> adj(v);

    cout << "Enter edges (source destination):\n";
    for (int i = 0; i < e; i++) {
        int s, d;
        cin >> s >> d;
        adj[s].push_back(d);
        adj[d].push_back(s);
    }

    vector<bool> visited(v,false);

    cout << "\nDFS Traversal: ";
    DFS(0, adj, visited);

    return 0;
}
```

Output :-