1. **BFS:-**

**from collections import deque**

**# define the graph as a dictionary of adjacency lists**

**graph = {}**

**# get the number of vertices and edges from the user**

**n = int(input("Enter the number of vertices: "))**

**m = int(input("Enter the number of edges: "))**

**# get the edges from the user and build the adjacency lists**

**for i in range(m):**

**u, v = map(int, input("Enter an edge (u v): ").split())**

**if u not in graph:**

**graph[u] = []**

**if v not in graph:**

**graph[v] = []**

**graph[u].append(v)**

**graph[v].append(u)**

**# get the source vertex from the user**

**s = int(input("Enter the source vertex: "))**

**# initialize the visited array and the queue**

**visited = [False] \* (n+1)**

**q = deque()**

**# mark the source vertex as visited and enqueue it**

**visited[s] = True**

**q.append(s)**

**# BFS algorithm**

**while q:**

**# dequeue a vertex from the queue**

**u = q.popleft()**

**print(u, end=" ")**

**# visit all the neighbors of u**

**for v in graph[u]:**

**if not visited[v]:**

**visited[v] = True**

**q.append(v)**

**2) DFS:-**

**from collections import defaultdict**

**class Graph:**

**# Constructor**

**def \_\_init\_\_(self):**

**# default dictionary to store graph**

**self.graph = defaultdict(list)**

**# function to add an edge to graph**

**def addEdge(self, u, v):**

**self.graph[u].append(v)**

**# A function used by DFS**

**def DFSUtil(self, v, visited):**

**# Mark the current node as visited**

**# and print it**

**visited.add(v)**

**print(v, end=' ')**

**# Recur for all the vertices**

**# adjacent to this vertex**

**for neighbour in self.graph[v]:**

**if neighbour not in visited:**

**self.DFSUtil(neighbour, visited)**

**# The function to do DFS traversal. It uses**

**# recursive DFSUtil()**

**def DFS(self, v):**

**# Create a set to store visited vertices**

**visited = set()**

**# Call the recursive helper function**

**# to print DFS traversal**

**self.DFSUtil(v, visited)**

**# Driver code**

**g = Graph()**

**n = int(input("Enter total no of edges"))**

**for i in range(0, n):**

**u = int(input("Enter start vertex: "))**

**v = int(input("Enter end vertex: "))**

**g.addEdge(u, v)**

**g.addEdge(v, u)**

**g.DFS(int(input("Enter the starting vertex: ")))**