```
class Graph:
    def __init__(self):
        self.graph = {}
    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        if v not in self.graph:
            self.graph[v] = []
        self.graph[u].append(v)
        self.graph[v].append(u)
    def dfs(self, start_node):
        visited = set()
        self._dfs_util(start_node, visited)
        return visited
    def _dfs_util(self, node, visited):
        visited.add(node)
        print(node, end=" ")
        for neighbor in self.graph[node]:
            if neighbor not in visited:
                self._dfs_util(neighbor, visited)
def main():
    g = Graph()
    num_edges = int(input("Enter the number of edges: "))
    for _ in range(num_edges):
        u, v = map(int, input("Enter an edge (u,v): ").split())
        g.add_edge(u, v)

    start_node = int(input("Enter the starting node for DFS Traversal: "))
    print("DFS Traversal starting from node", start_node, ":")
    g.dfs(start_node)
if __name__ == "__main__":
    main()
```

Output:

Enter the number of edges: 7

Enter an edge (u,v): 1 4

Enter an edge (u,v): 2 5

Enter an edge (u,v): 6 9

Enter an edge (u,v): 4 3

Enter an edge (u,v): 3 1

Enter an edge (u,v): 1 2

Enter an edge (u,v): 7 5

Enter the starting node for DFS Traversal: 5

DFS Traversal starting from node 5 :

5 2 1 4 3 7