```
graph = {}
edge_set =set()

def add_node(node):
    if node in graph:
        print(f" '(node)' already exists. please enter a different node.")
        return False
    graph[node] =[]
    return True

def add_edge(u,v):
    edge = tuple(sorted((u,v)))
    if edge in edge_set:
        print (f" Edge {u} -{v} already exists. Please enter a differnt
node.")
        return False
    if u not in graph or v not in graph:
        print("Both nodes must be added before connecting them
with an edge.")
        return False
    graph[u].append(v)
    graph[v].append(u)
    edge_set.add(edge)
    return True
#bfs
def bfs(start):
    visited =[]
    queue =[start]
    print("BFS:", end =" ")
```

```
    while queue:
        node = queue.pop(0)
        if node not in visited:
            print(node, end=" ")
            visited.append(node)
            for neighbor in graph[node]:
                if neighbor not in visited:
                    queue.append(neighbor)
    print()

#DFS
def dfs(node , visited=None):
        if visited is None:
            visited=[]
            print("DFS:", end=" ")
        if node not in visited:
            print(node, end=" ")
            visited.append(node)
            for neighbor in graph[node]:
                dfs(neighbor, visited)

#input
n = int(input("enter number of nodes: "))
i=0
while i<n:
    node = input(f"enter node {i+1}: ").strip()
    if add_node(node):
        i += 1
```

```
#add edges
e= int(input("enter the no of edges: "))
for i in range(e):
   while True:
      u,v = input(f"enter edges {i+1} (two nodes) : ").split()

      if add_edge(u,v):
         break

# start
start = input("Enter starting node: ").strip()
if start in graph:
   bfs(start)
   dfs(start)
   print()
else:
   print("Starting node is not found in the graph.")
```