DAY-8,9:

SEARCHING AND SORTING:  
PROGRAM:

class GraphList:

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 bfs(self, start):

visited = set()

queue = [start]

while queue:

vertex = queue.pop(0)

if vertex not in visited:

visited.add(vertex)

print(vertex, end=' ')

for i in self.graph[vertex]:

if i not in visited:

queue.append(i)

def display(self):

for node in self.graph:

print(f"{node} --> {self.graph[node]}")

g = GraphList()

g.add\_edge(1, 2)

g.add\_edge(1, 3)

g.add\_edge(2, 3)

g.add\_edge(2, 4)

g.add\_edge(3, 5)

g.add\_edge(5, 7)

print("Graph representation:")

g.display()

print("\nBFS traversal starting from node 1:")

g.bfs(1)

OUTPUT:

Graph representation:

1 --> [2, 3]

2 --> [1, 3, 4]

3 --> [1, 2, 5]

4 --> [2]

5 --> [3, 7]

7 --> [5]

BFS and DFS:

PROGRAM:

class GraphList:

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 bfs(self, start):

visited=set()

queue=[start]

while queue:

vertex=queue.pop(0)

if vertex not in visited:

visited.add(vertex)

print(vertex, end=' ')

for i in self.graph[vertex]:

if i not in visited:

queue.append(i)

def dfs(self,start, visited=None):

if visited is None:

visited=set()

visited.add(start)

print(start, end=' ')

for i in self.graph[start]:

if i not in visited:

self.dfs(i, visited)

def display(self):

for node in self.graph:

print(f"{node} --> {self.graph[node]}")

# Inserting nodes

g = GraphList()

g.add\_edge(1, 2)

g.add\_edge(1, 3)

g.add\_edge(2, 3)

g.add\_edge(2, 4)

g.add\_edge(3, 5)

g.add\_edge(5, 7)

print("Graph representation:")

g.display()

print("\nBFS traversal starting from node 1:")

g.bfs(1)

print("\nDFS traversal starting from node 1:")

g.dfs(1)

OUTPUT:

Graph representation:

1 --> [2, 3]

2 --> [1, 3, 4]

3 --> [1, 2, 5]

4 --> [2]

5 --> [3, 7]

7 --> [5]