Dfs bfs

def dfs(visited,graph,node):

if node not in visited:

print(node,end = " ")

visited.add(node)

for neighbour in graph[node]:

dfs(visited, graph, neighbour)

def bfs(visited,graph,node,queue):

visited.add(node)

queue.append(node)

while queue:

s = queue.pop(0)

print(s,end = " ")

for neighbour in graph[s]:

if neighbour not in visited:

visited.add(neighbour)

queue.append(neighbour)

def main():

visited1 = set() # TO keep track of DFS visited nodes

visited2 = set() # TO keep track of BFS visited nodes

queue = [] # For BFS

n = int(input("Enter number of nodes : "))

graph = dict()

for i in range(1,n+1):

edges = int(input("Enter number of edges for node {} : ".format(i)))

graph[i] = list()

for j in range(1,edges+1):

node = int(input("Enter edge {} for node {} : ".format(j,i)))

graph[i].append(node)

print("The following is DFS")

dfs(visited1, graph, 1)

print()

print("The following is BFS")

bfs(visited2, graph, 1, queue)

if \_\_name\_\_=="\_\_main\_\_":

main()

**inputs**

Enter number of nodes : 4

Enter number of edges for node 1 : 2

Enter edge 1 for node 1 : 2

Enter edge 2 for node 1 : 3

Enter number of edges for node 2 : 1

Enter edge 1 for node 2 : 4

Enter number of edges for node 3 : 1

Enter edge 1 for node 3 : 4

Enter number of edges for node 4 : 0

Theory

Depth First Search (DFS):

DFS is a graph traversal algorithm that starts from a source node and explores as far as possible along each branch before backtracking.

It uses recursion and a set to keep track of visited nodes.

In the code:

A function dfs() is defined.

It checks if a node is visited; if not, it processes the node and recursively visits all its unvisited neighbors.

Characteristics:

Works like a stack (LIFO behavior).

Good for pathfinding and topological sorting.

Breadth First Search (BFS):

BFS is a graph traversal algorithm that starts from a source node and explores all its neighbors before going to the next level neighbors.

It uses a queue and a set to keep track of visited nodes.

In the code:

A function bfs() is defined.

It starts from the node, enqueues it, and then continues to process all neighbors level by level.

Characteristics:

Works like a queue (FIFO behavior).

Good for shortest path in unweighted graphs.