# DFS algorithm

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

if visited is None:

visited = set()

visited.add(start)

print(start)

for next in graph[start] - visited:

dfs(graph, next, visited)

return visited

graph = {'0': set(['1', '2']),

'1': set(['0', '3', '4']),

'2': set(['0']),

'3': set(['1']),

'4': set(['2', '3'])}

dfs(graph, '0')

OUTPUT:

0

1

3

4

2

2

{'0', '1', '2', '3', '4'}

# BFS algorithm in Python

import collections

# BFS algorithm

def bfs(graph, root):

visited, queue = set(), collections.deque([root])

visited.add(root)

while queue:

# Dequeue a vertex from queue

vertex = queue.popleft()

print(str(vertex) + " ", end="")

# If not visited, mark it as visited, and

# enqueue it

for neighbour in graph[vertex]:

if neighbour not in visited:

visited.add(neighbour)

queue.append(neighbour)

if \_\_name\_\_ == '\_\_main\_\_':

graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}

print("Following is Breadth First Traversal: ")

bfs(graph, 0)

OUTPUT:

Following is Breadth First Traversal:

0 1 2 3