PROGRAM-1

Aim: Write a python program to implement Breadth First Search (BFS) Traversal

Logic: Breadth-first search (BFS) is an algorithm for searching a tree data structure for a node that satisfies a given property. It starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level. Extra memory, usually a queue, is needed to keep track of the child nodes that were encountered but not yet explored.

Example:

Algorithm:

Step 1: Define a Queue of size total number of vertices in the graph.
Step 2: Select any vertex as starting point for traversal. Visit that vertex and insert it into the Queue.
Step 3: Visit all the non-visited adjacent vertices of the vertex which is at front of the Queue and insert them into the Queue.
Step 4: When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.
Step 5: Repeat steps 3 and 4 until queue becomes empty.
Step 6: When queue becomes empty, then produce final spanning tree by removing unused edges from the graph

Implementation:

```
from queue import Queue

def initializeValues(visited,level,parent,queue):
    for node in adjList.keys():
        visited[node] = False
        parent[node] = None
        level[node] = -1
    s="A"
    visited[s] = True
    level[s] = 0
    queue.put(s)
```

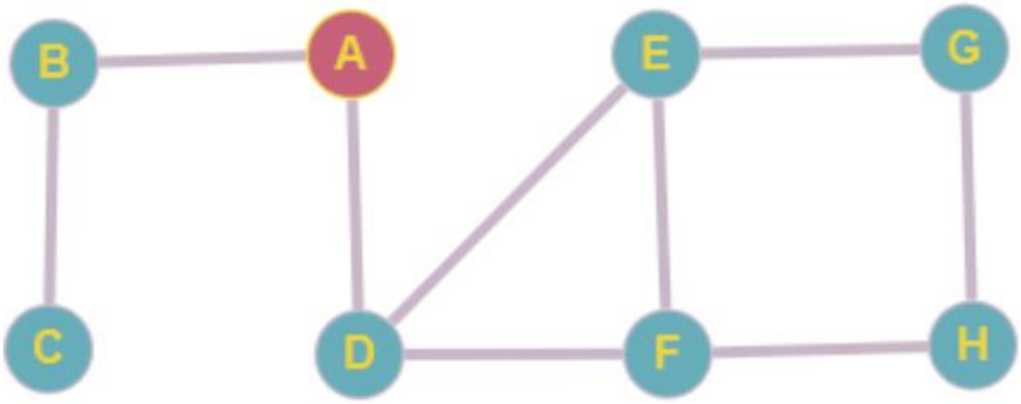
```
def bfsTraversal(adjList,parent,level,queue,bfsTraversalOutput):
    while not queue.empty():
        u = queue.get()
        bfsTraversalOutput.append(u)
        for v in adjList[u]:
            if not visited[v]:
                visited[v] = True
                parent[v] = u
                level[v] = level[u] + 1
                queue.put(v)

def printValues(bfsTraversalOutput):
    print("Output : ",end="")
    for node in bfsTraversalOutput:
        print(node,end=" ")
    print()

if __name__ == '__main__':
    adjList = {
        "A" : ["B","D"],
        "B" : ["C","A"],
        "C" : ["B"],
        "D" : ["A","E","F"],
        "E" : ["D","F","G"],
        "F" : ["D","E","H"],
        "G" : ["E","H"],
        "H" : ["G","F"]
    }
    visited = {}
    level = {}
    parent = {}
    bfsTraversalOutput=[]

    queue = Queue()
    initializeValues(visited,level,parent,queue)
    bfsTraversal(adjList,parent,level,queue,bfsTraversalOutput)
    printValues(bfsTraversalOutput)
```

Input:

Code:

```
adjList = {
        "A" : ["B","D"],
        "B" : ["C","A"],
        "C" : ["B"],
        "D" : ["A","E","F"],
        "E" : ["D","F","G"],
        "F" : ["D","E","H"],
        "G" : ["E","H"],
        "H" : ["G","F"]
    }
```

Graphically:



Output: