**Assignment 1**

**TITLE:** Implement DFS and BFS Algorithm. Use and Undirected Graph and develop a Recursive Algorithm for searching all the vertices of the graph or tree data structure.

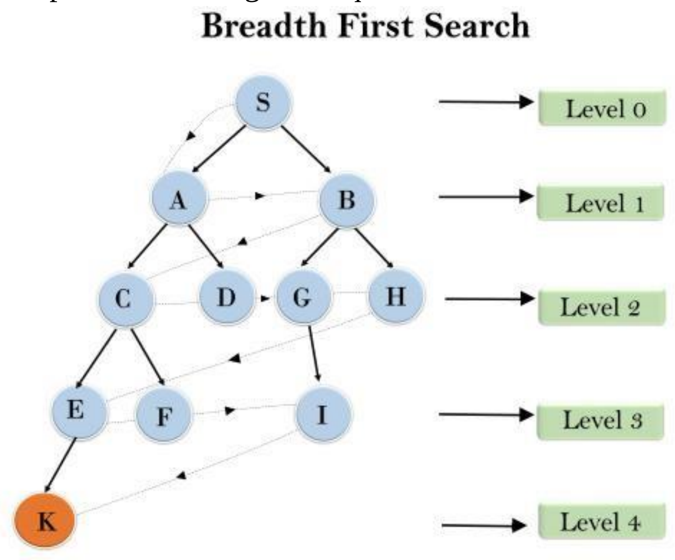
**AIM:** Aim of this practical is to develop DFS and BFS algorithm programs in programming language.

**OBJECTIVES:** Based on above main aim following are the objectives

1. To study BFS
2. To study DFS
3. To apply algorithmic logic in implementation of program.

## 1. Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.



In the above tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

S---> A--->B---->C--->D---->G--->H--->E---->F---->I >K

- **Time Complexity:** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.
- **$T(b) = 1+b^2+b^3+. \quad + b^d = O(b^d)$**
- **Space Complexity:** Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.
- **Completeness:** BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.
- **Optimality:** BFS is optimal if path cost is a non-decreasing function of the depth of the node.

**Algorithm:**

1. Pick any node, visit the adjacent unvisited vertex, mark it as visited, display it, and insert it in a queue.
2. If there are no remaining adjacent vertices left, remove the first vertex from the queue.
3. Repeat step 1 and step 2 until the queue is empty or the desired node is found.

**Program:**

```
graph = {

  'A' : ['B','C'],

  'B' : ['D', 'E'],

  'C' : ['F'],

  'D' : [],

  'E' : ['F'],
  'F' : []

}

visited = [] # List to keep track of visited nodes.
queue = []     #Initialize a queue


def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)
```

```
        visited.append(neighbour)
        queue.append(neighbour)



# Driver Code

print("Following is the Path using Breadth-First Search")
bfs(visited, graph, 'A')
```

**Output:**

```
Following is the Path using Breadth-First Search
A B C D E F
```

**Conclusion**: Thus we have studied BFS & DFS algorithm in detail and implemented using recursive function.