Experiment No 1

Aim: WAP to implement BFS Search technique.

Theory:

Uninformed Search: The search algorithms in this section have no additional information on the goal node other than the one provided in the problem definition. The plans to reach the goal state from the start state differ only by the order and/or length of actions. Uninformed search is also called **Blind search**.

Breadth First Search:

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a ‘search key’), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

**Time complexity:**Equivalent to the number of nodes traversed in BFS until the shallowest solution.   
**Space complexity:**Equivalent to how large can the fringe get.   
**Completeness:**BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.  
**Optimality:**BFS is optimal as long as the costs of all edges are equal.

**Code:**

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)

while queue:

s = queue.pop(0)

print (s, end = " ")

for neighbour in graph[s]:

if neighbour not in visited:

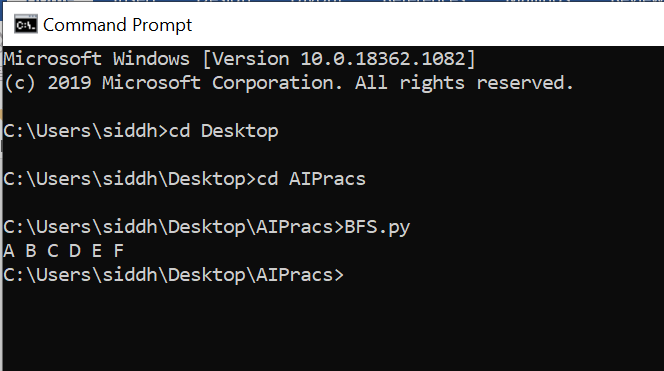
visited.append(neighbour)

queue.append(neighbour)

# Driver Code

bfs(visited, graph, 'A')

Output:



Conclusion: