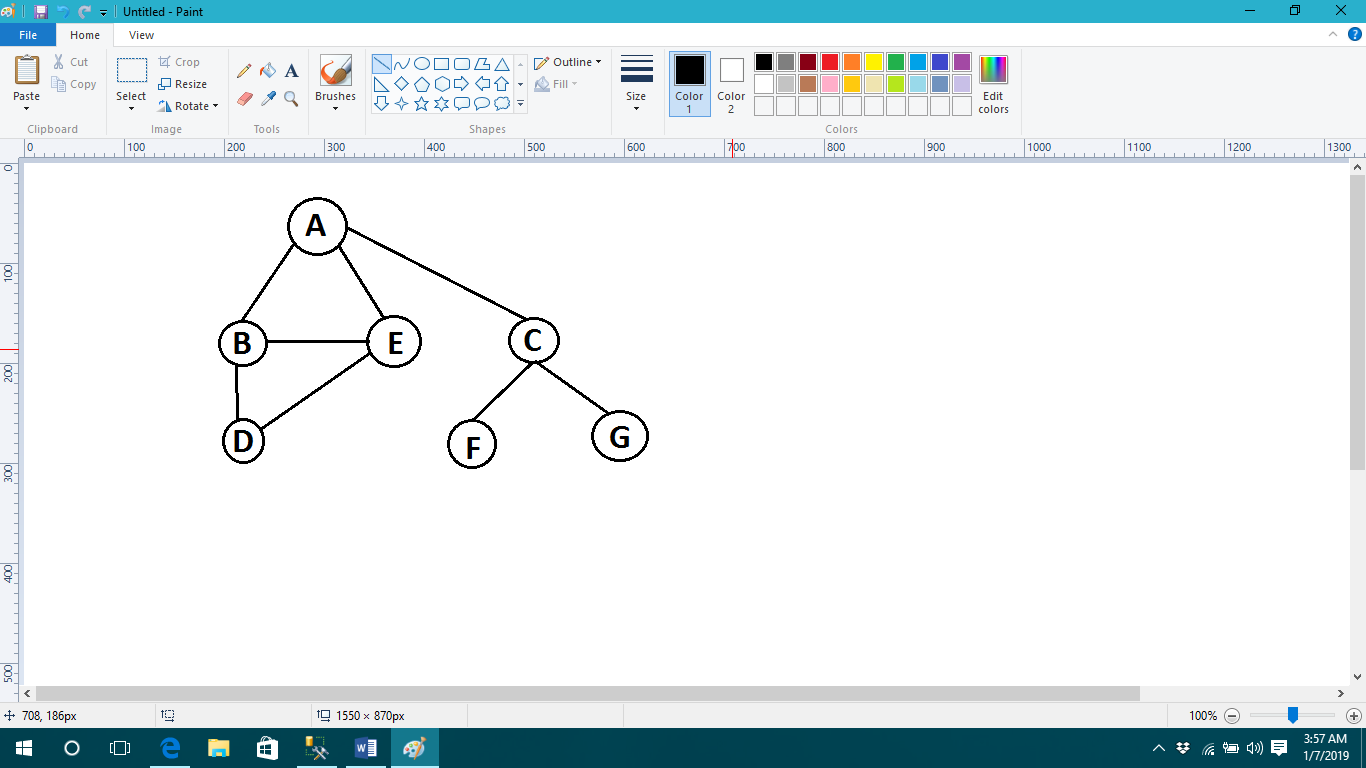
**IMPLEMENTATION OF DEPTH FIRST SEARCH IN PYTHON**

**EXERCISE:**

**Object 01:**

Consider the following graph:



Represent a graph by using any data structure .Apply DFS if the starting node is “A” and goal node is “G”. Show result in form of open & close list

**Source Code:**

graph = {'A':['B','E','C'],'B':['D'],'C':['F','G'],'D':[ ],'E':['D'],'F':[ ],'G':[ ]}

def recursive\_dfs(graph, start, path=[]):

print(path)

path=path+[start]

for node in graph[start]:

if not node in path:

path=recursive\_dfs(graph, node, path)

return path

def iterative\_dfs(graph, start, path=[]):

q=[start]

while q:

print(q)

v=q.pop(0)

if v not in path:

path=path+[v]

q=graph[v]+q

return path

print('OPEN LIST')

iterative\_dfs(graph, 'A')

print('CLOSE LIST')

print('FINAL OUTPUT ', recursive\_dfs(graph, 'A'))

**Output:**

OPEN LIST

['A']

['B', 'E', 'C']

['D', 'E', 'C']

['E', 'C']

['D', 'C']

['C']

['F', 'G']

['G']

CLOSE LIST

[]

['A']

['A', 'B']

['A', 'B', 'D']

['A', 'B', 'D', 'E']

['A', 'B', 'D', 'E', 'C']

['A', 'B', 'D', 'E', 'C', 'F']

FINAL OUTPUT ['A', 'B', 'D', 'E', 'C', 'F', 'G']

**Object 02:**

By using graph ,ask user to state starting and goal node .Apply DFS and print complete path from source to destination.

**Source Code:**

graph = {'A':['B','E','C'],'B':['D'],'C':['F','G'],'D':[ ],'E':['D'],'F':[ ],'G':[ ]}

def iterative\_dfs(graph, start,end, path=[]):

q=[start]

while q:

v=q.pop(0)

if v not in path:

path=path+[v]

q=graph[v]+q

return path

a=input('Enter initial point: ')

b=input('Enter destination point: ')

print('OUTPUT',iterative\_dfs(graph,a,b))

**Output:**

Enter initial point: A

Enter destination point: C

OUTPUT ['A', 'B', 'E', 'C']