

Exercise - 4(BFS & DFS)

Team

Ritesh Rai(333)

Harshil Agaral(325)

Puneet Sharma(331)

Prakhar Vijay(337)

Vivek Burugadda(316)

BFS source code

```
from collections import defaultdict
```

```
# This class represents a directed graph
```

```
# using adjacency list representation
```

```
class Graph:
```

```
    # Constructor
```

```
    def __init__(self):
```

```
        # default dictionary to store graph
```

```
        self.graph = defaultdict(list)
```

```
    # function to add an edge to graph
```

```
    def addEdge(self,u,v):
```

```
        self.graph[u].append(v)
```

```
    # Function to print a BFS of graph
```

```
    def BFS(self, s):
```

```
        # Mark all the vertices as not visited
```

```
        visited = [False] * (max(self.graph) + 1)
```

```
        # Create a queue for BFS
```

```
        queue = []
```

```
# Mark the source node as
# visited and enqueue it
queue.append(s)
visited[s] = True
```

```
while queue:
```

```
    # Dequeue a vertex from
    # queue and print it
    s = queue.pop(0)
    print (s, end = " ")
```

```
    # Get all adjacent vertices of the
    # dequeued vertex s. If a adjacent
    # has not been visited, then mark it
    # visited and enqueue it
    for i in self.graph[s]:
        if visited[i] == False:
            queue.append(i)
            visited[i] = True
```

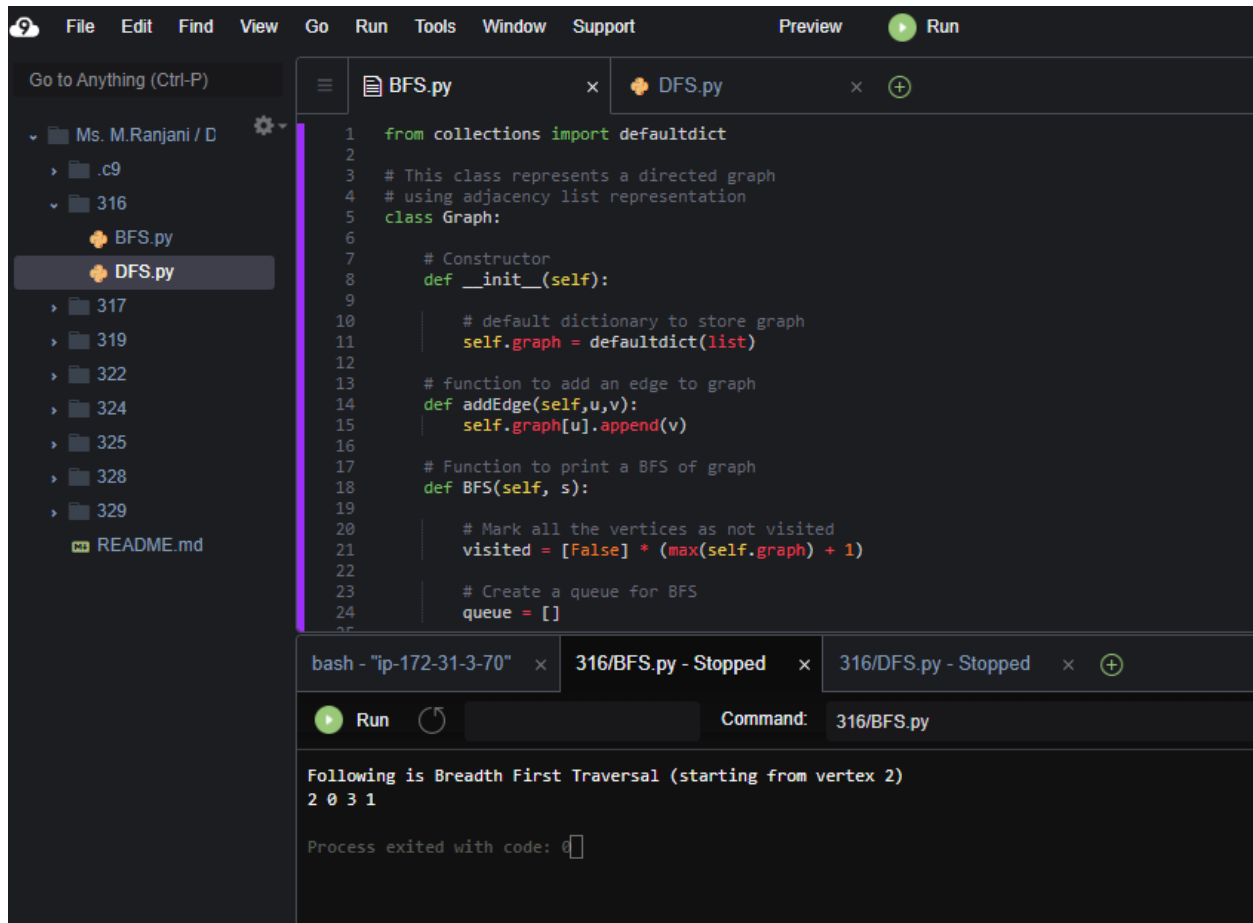
```
# Driver code
```

```
# Create a graph given in
# the above diagram
g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)
```

```
print ("Following is Breadth First Traversal"
      " (starting from vertex 2)")
```

g.BFS(2)

Output



The screenshot shows a code editor with two tabs: `BFS.py` and `DFS.py`. The `DFS.py` tab is active, displaying the following Python code:

```
1 from collections import defaultdict
2
3 # This class represents a directed graph
4 # using adjacency list representation
5 class Graph:
6
7     # Constructor
8     def __init__(self):
9
10         # default dictionary to store graph
11         self.graph = defaultdict(list)
12
13     # function to add an edge to graph
14     def addEdge(self,u,v):
15         self.graph[u].append(v)
16
17     # Function to print a BFS of graph
18     def BFS(self, s):
19
20         # Mark all the vertices as not visited
21         visited = [False] * (max(self.graph) + 1)
22
23         # Create a queue for BFS
24         queue = []
```

Below the code editor, there are two terminal windows. The first terminal window shows the command `bash - "ip-172-31-3-70"` and the output `316/BFS.py - Stopped`. The second terminal window shows the command `316/DFS.py - Stopped` and the output `316/DFS.py - Stopped`. The output of the `BFS` process is as follows:

```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
Process exited with code: 0
```

DFS source code

from collections import defaultdict

This class represents a directed graph using
adjacency list representation

class Graph:

Constructor

def __init__(self):

```

# default dictionary to store graph
self.graph = defaultdict(list)

# function to add an edge to graph
def addEdge(self, u, v):
    self.graph[u].append(v)

# A function used by DFS
def DFSUtil(self, v, visited):

    # Mark the current node as visited
    # and print it
    visited.add(v)
    print(v, end=' ')

    # Recur for all the vertices
    # adjacent to this vertex
    for neighbour in self.graph[v]:
        if neighbour not in visited:
            self.DFSUtil(neighbour, visited)

# The function to do DFS traversal. It uses
# recursive DFSUtil()
def DFS(self, v):

    # Create a set to store visited vertices
    visited = set()

    # Call the recursive helper function
    # to print DFS traversal
    self.DFSUtil(v, visited)

# Driver code

```

Create a graph given

in the above diagram

```
g = Graph()
```

```
g.addEdge(0, 1)
```

```
g.addEdge(0, 2)
```

```
g.addEdge(1, 2)
```

```
g.addEdge(2, 0)
```

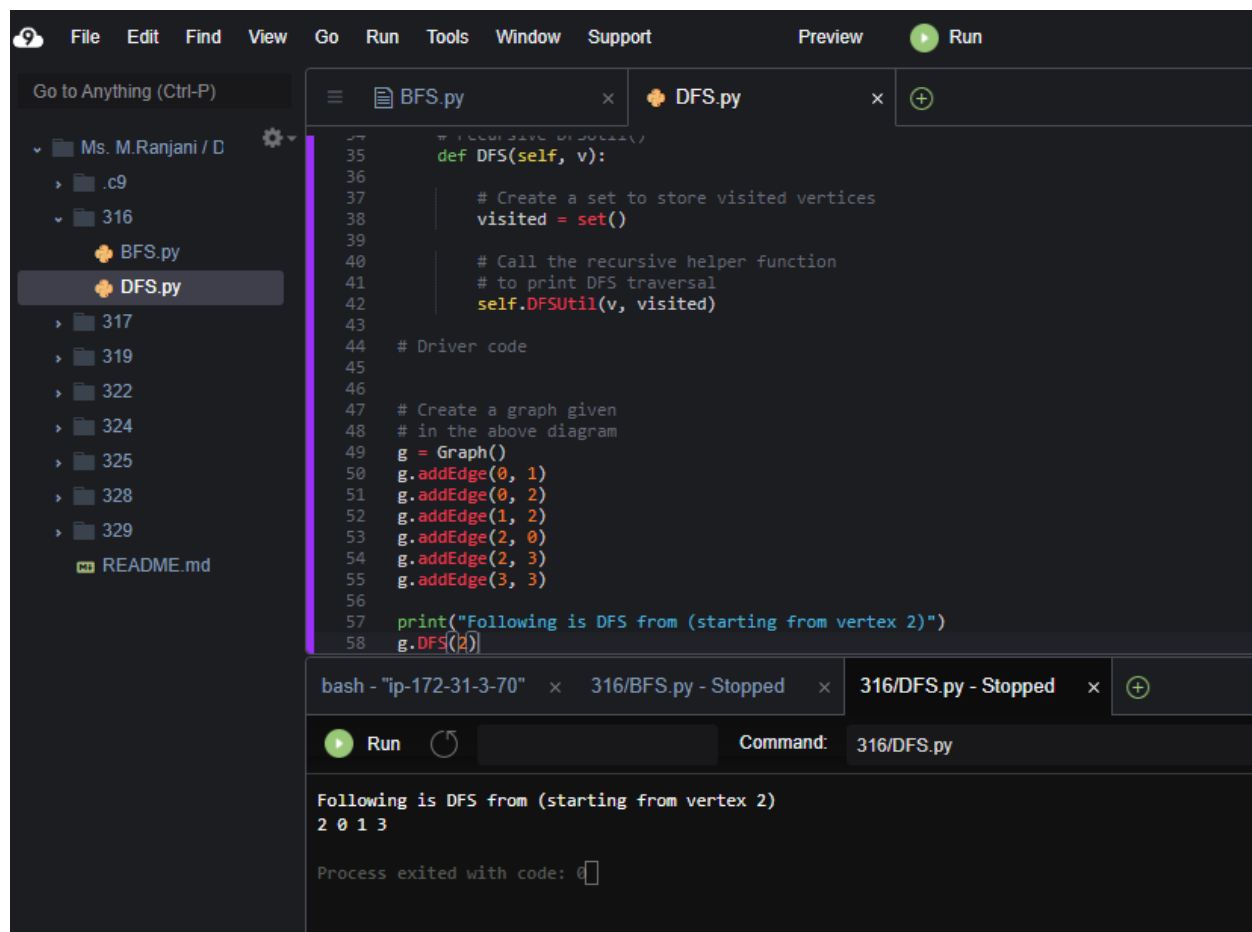
```
g.addEdge(2, 3)
```

```
g.addEdge(3, 3)
```

```
print("Following is DFS from (starting from vertex 2)")
```

```
g.DFS(2)
```

Output



```
34 # Recursive DFS util()
35 def DFS(self, v):
36     # Create a set to store visited vertices
37     visited = set()
38     # Call the recursive helper function
39     # to print DFS traversal
40     self.DFSUtil(v, visited)
41
42 # Driver code
43
44 # Create a graph given
45 # in the above diagram
46 g = Graph()
47 g.addEdge(0, 1)
48 g.addEdge(0, 2)
49 g.addEdge(1, 2)
50 g.addEdge(2, 0)
51 g.addEdge(2, 3)
52 g.addEdge(3, 3)
53
54 print("Following is DFS from (starting from vertex 2)")
55 g.DFS(2)
```

bash - "ip-172-31-3-70" x 316/BFS.py - Stopped x 316/DFS.py - Stopped x

Run Command: 316/DFS.py

Following is DFS from (starting from vertex 2)
2 0 1 3

Process exited with code: 0