TASK:1

Implementation of Graph search algorithms (**Breadth first search and Depth First Search**) using following constraints.

Aim: To Implement of Graph search algorithms (Breadth first search and Depth First Search) using Python.

Task 1A

Algorithm:

BFS

Step 1: Start by putting any one of the graph's vertices at the back of the queue.

Step 2: Now take the front item of the queue and add it to the visited list.

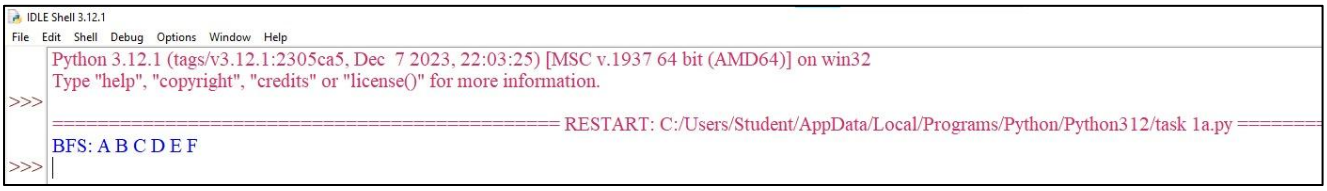
Step 3: Create a list of that vertex's adjacent nodes. Add those which are not within the visited list to the rear of the queue.

Step 4: Keep continuing steps two and three till the queue is empty.

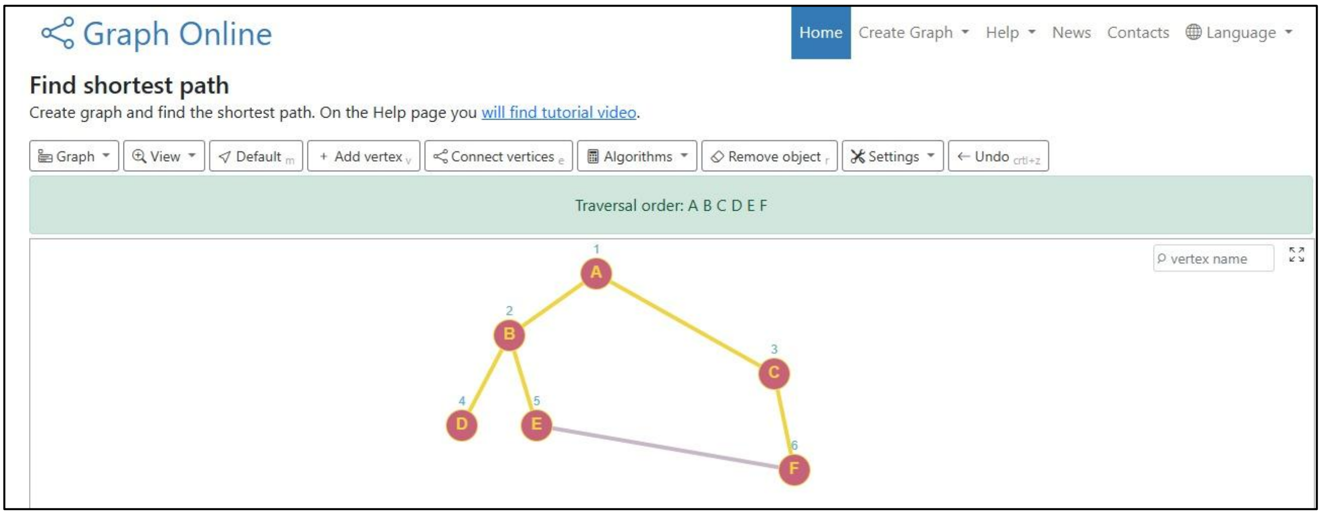
Program

```
from collections import deque

def bfs(graph, start):

  queue, visited = deque([start]), set() print("BFS:",

  end=" ")

  while queue:

    node = queue.popleft() if node not in visited: print(node, end=" ")

  visited.add(node) queue.extend(neighbor for neighbor in graph[node] if neighbor

  not in visited) print()

# Example graph graph

= {

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

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

  'C': ['A', 'F'],

  'D': ['B'],

  'E': ['B', 'F'],
    'F': ['C', 'E']

}

bfs(graph, 'A')
```

Output:



```
    Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    ============================================ RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python312/task 1a.py ========
    BFS: A B C D E F
>>> |
```



Find shortest path

Create graph and find the shortest path. On the Help page you will find tutorial video.

Traversal order: A B C D E F

Task1 B
Algorithm

DFS –

Step 1: Declare a queue and insert the starting Vertex.

Step 2: Initialize a visited array and mark the starting Vertex as visited.

Step3: Remove the First vertex of queue.

Step 4: Mark that vertex as visited

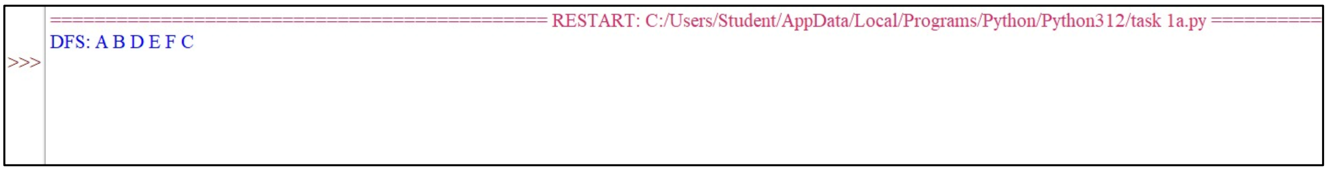
Step 5: Insert all the unvisited neighbors of the vertex into queue.

Step 6: stop.

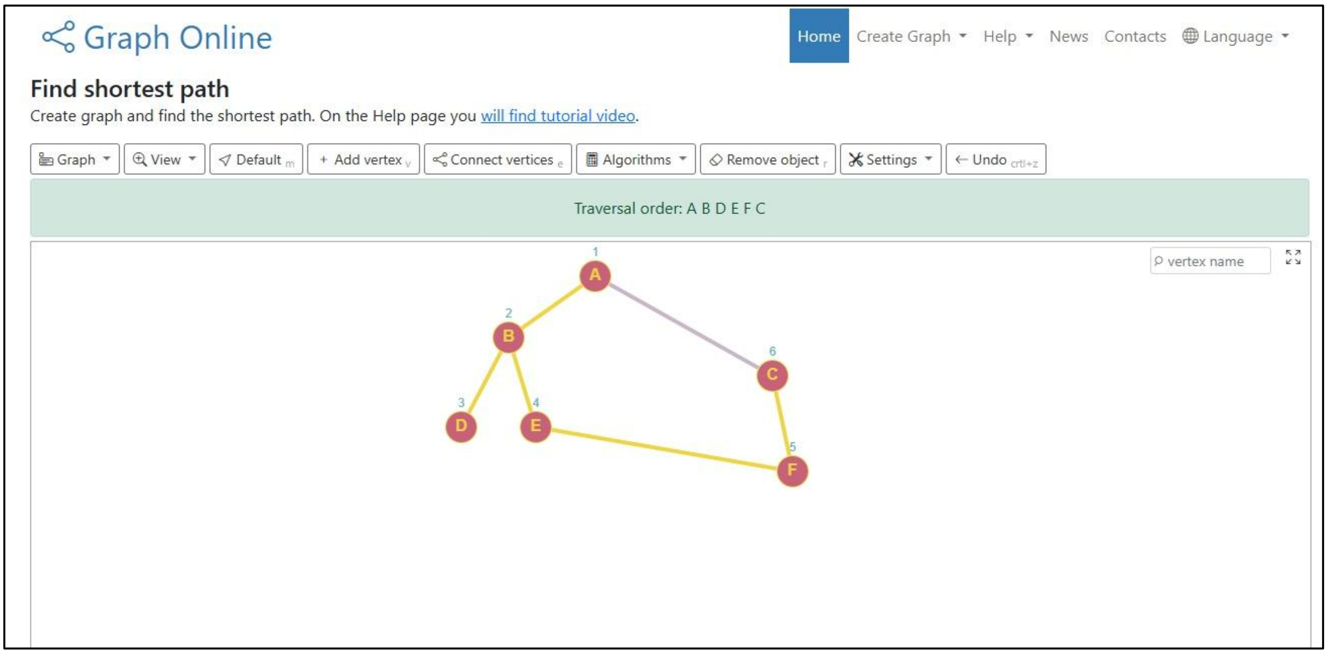
Program

```
from collections import deque

def dfs(graph, start):

    stack, visited = [start], set()
    print("DFS:", end=" ")
    while stack:
     node = stack.pop()
    if node not in visited: print(node, end=" ") visited.add(node)

  stack.extend(reversed([neighbor for neighbor in graph[node] if neighbor not in visited]))

  print()

# Example graph graph

= {

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

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

  'C': ['A', 'F'],

  'D': ['B'],

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

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

}

dfs(graph, 'A')
```

Output:



```
============================================ RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python312/task 1a.py ==========
DFS: A B D E F C
>>>
```



Find shortest path

Create graph and find the shortest path. On the Help page you will find tutorial video.

Traversal order: A B D E F C

Result:

Thus the Implementation of Graph search algorithms (Breadth first search and Depth First Search) using Python was successfully executed and output was verified.