Objective: Implementation and analysis of Depth first search

Depth First Search

Traversal means visiting all the nodes of a graph. Depth first traversal or Depth first Search is a recursive algorithm for searching all the vertices of a graph or tree data structure.

DFS algorithm

A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

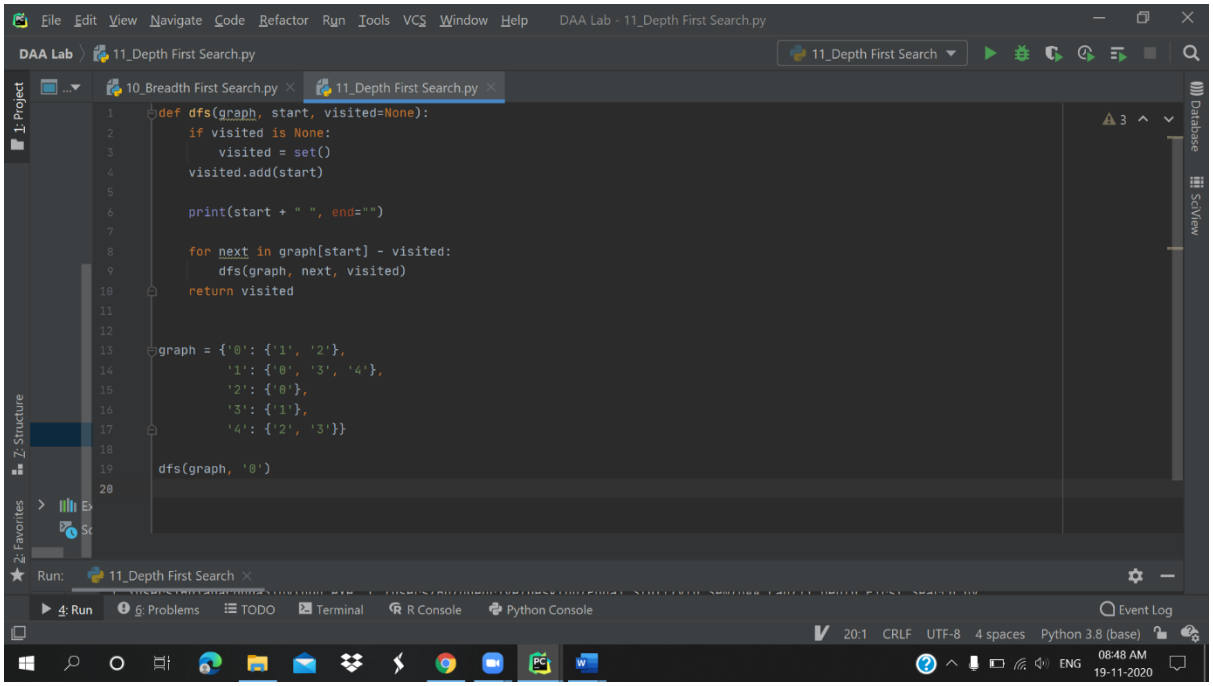
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

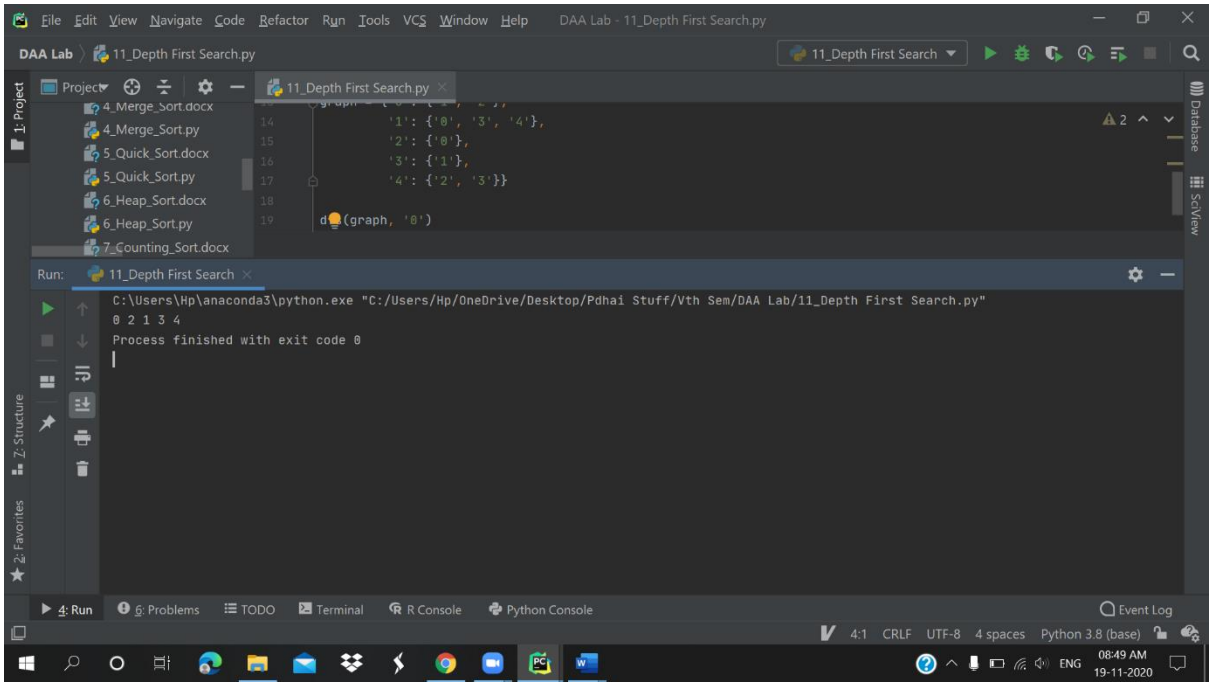
Code:

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)

    print(start + " ", end="")

    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited


graph = {'0': {'1', '2'},
         '1': {'0', '3', '4'},
         '2': {'0'},
         '3': {'1'},
         '4': {'2', '3'}}

dfs(graph, '0')
```



Output:

0 2 1 3 4



Time Complexities:

The time complexity of the DFS algorithm is represented in the form of O(V + E), where V is the number of nodes and E is the number of edges.
The space complexity of the algorithm is O(V).

DFS Applications:

- For finding the path
- To test if the graph is bipartite
- For finding the strongly connected components of a graph
- For detecting cycles in a graph