Objective: Implementation and analysis of Breadth first search

Breadth First Search

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

BFS algorithm

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

- Visited
- Not Visited

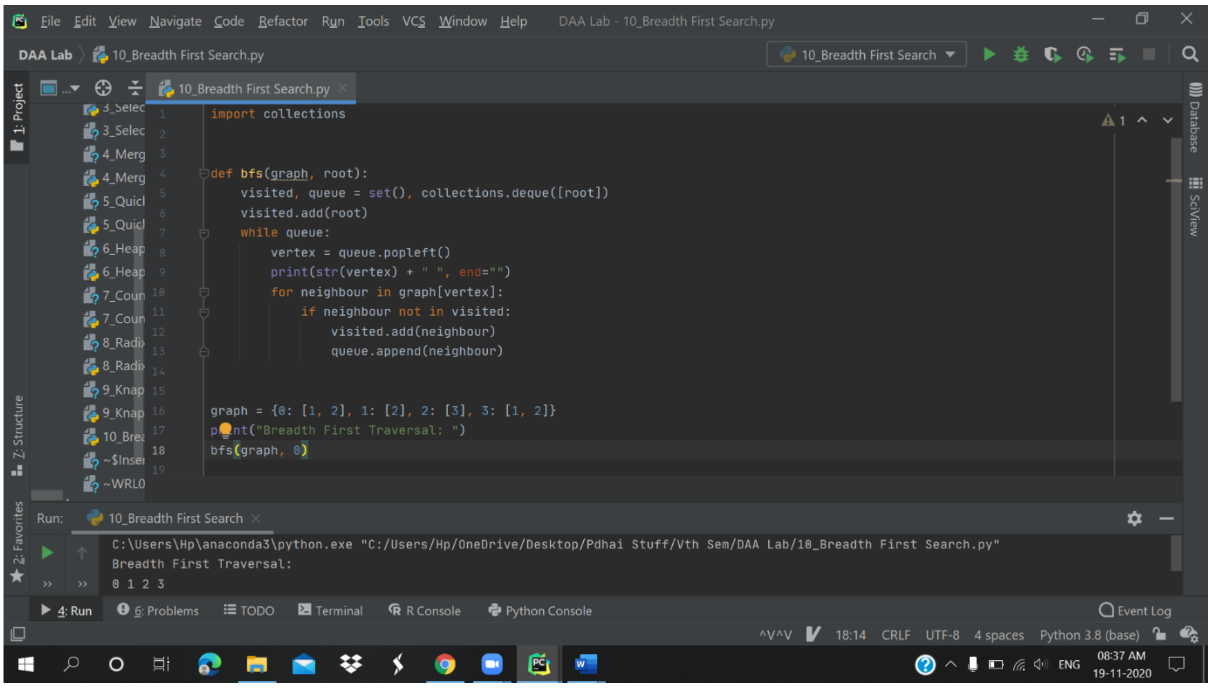
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

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

Code:

```python
import collections
def bfs(graph, root):
    visited, queue = set(), collections.deque([root])
    visited.add(root)
    while queue:
        vertex = queue.popleft()
        print(str(vertex) + " ", end="")
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)


graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
print("Breadth First Traversal: ")
bfs(graph, 0)
```

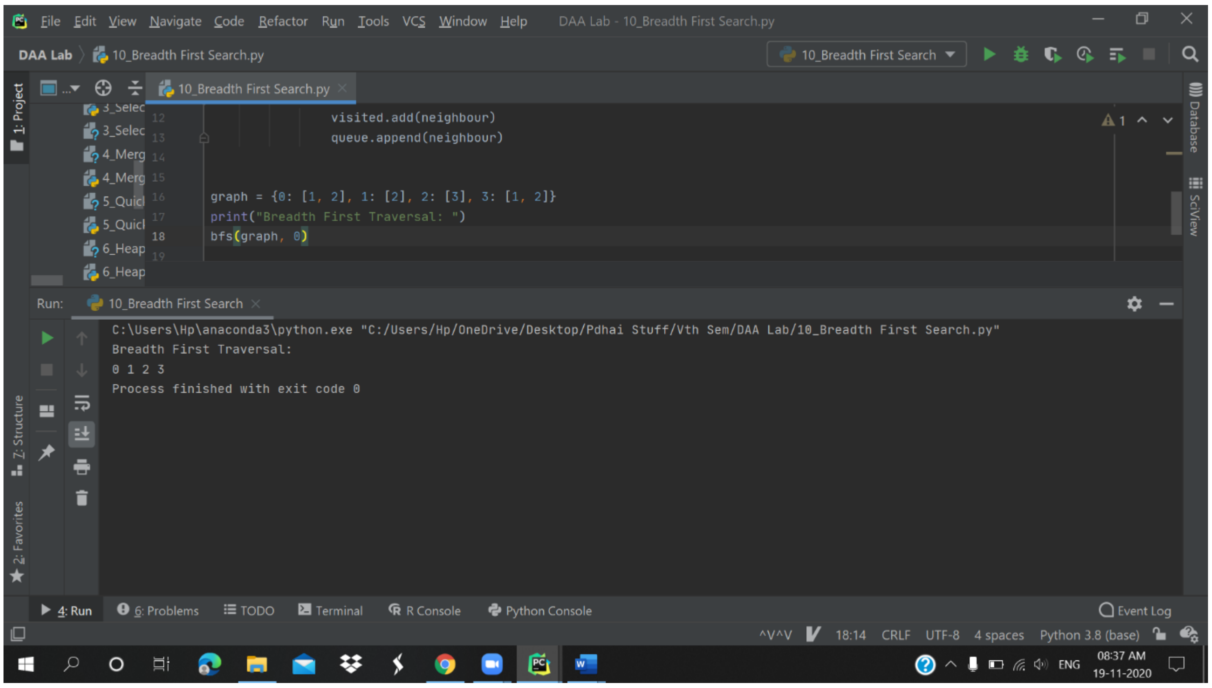

File Edit View Navigate Code Refactor Run Tools VCS Window Help
DAA Lab - 10_Breadth First Search.py
DAA Lab
10_Breadth First Search.py
10_Breadth First Search
Run: 10_Breadth First Search
4: Run
6: Problems
TODO
Terminal
R Console
Python Console
Event Log
18:14 CRLF UTF-8 4 spaces Python 3.8 (base)
08:37 AM
19-11-2020

```python
import collections


def bfs(graph, root):
    visited, queue = set(), collections.deque([root])
    visited.add(root)
    while queue:
        vertex = queue.popleft()
        print(str(vertex) + " ", end="")
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)


graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
print("Breadth First Traversal: ")
bfs(graph, 0)
```

```
C:\Users\Hp\anaconda3\python.exe "C:/Users/Hp/OneDrive/Desktop/Pdhai Stuff/Vth Sem/DAA Lab/10_Breadth First Search.py"
Breadth First Traversal: 
0 1 2 3
```

Output:

```
Breadth First Traversal:

0 1 2 3
```



```
                visited.add(neighbour)
                queue.append(neighbour)


graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
print("Breadth First Traversal: ")
bfs(graph, 0)
```

```
C:\Users\Hp\anaconda3\python.exe "C:/Users/Hp/OneDrive/Desktop/Pdhai Stuff/Vth Sem/DAA Lab/10_Breadth First Search.py"
Breadth First Traversal:
0 1 2 3
Process finished with exit code 0
```

Time Complexities:

The time complexity of the BFS 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).

BFS Applications:

- To build index by search index
- For GPS navigation
- Path finding algorithms
- In Ford-Fulkerson algorithm to find maximum flow in a network
- Cycle detection in an undirected graph
- In minimum spanning tree