EAST WEST UNIVERSITY

Lab-1

Tanvir Mobasshir

ID: 2019-2-60-025

Summer 2022

Date: 10-06-2022

CSE366

Course: Artificial Intelligence

Faculty: Redwan Ahmed Rizvee

Section: 02

1. Problem Statement

In this assignment, it has been asked to get the shortest path and its distance from a source to a destination of an unweighted and undirected graph using BFS. Later, an exploration tree is to be founded.

So, the assignment includes two problems:

- Getting the shortest path and its distance
- Getting the exploration tree i.e., exploration order of each node.

2. Solution Approach

Initial Steps:

- Took user input for the graph, source and destination
- Created two different functions for BFS and displaying the exploration tree.

Data structure:

- A ***queue*** to keep track of the nodes that has been explored
- A ***distance (list)*** to keep the distance of each node from the source node
- A ***parent (list)*** to record parent of each node
- An ***explored_node (list)*** to keep record of the order in which each node was explored.
- An ***order (dictionary)*** which keeps order numbers as key and node as value. It helps to display the order.

Algorithm:

- Pushed source node to the queue
- Initialised ***distance list, parent list*** and ***explored_node list*** with ***-1***
- Initialised the order of source node to ***1*** in ***explored_node***.
- Traverse the queue while executing following action
  - Pop the first node from queue
  - Calculate distance of neighbouring nodes
  - Record parent nodes for each node that has been explored
  - Record orders of the explored nodes
  - Push newly explored node to the queue

3. Some Inputs and Outputs

Input 1

```
4 5
1 2
1 3
2 4
3 4
1 4
1 4
```

Output

```
1
1 4

Explored nodes in order:  1 2 3 4
node: 1, exploration order: 1
node: 2, exploration order: 2
node: 3, exploration order: 3
node: 4, exploration order: 4
```

Input 2

```
6 8
1 2
1 3
1 4
1 5
2 5
3 5
4 5
5 6
1 6
```

Output

```
2
1 5
5 6

Explored nodes in order:  1 2 3 4 5 6
node: 1, exploration order: 1
node: 2, exploration order: 2
node: 3, exploration order: 3
node: 4, exploration order: 4
node: 5, exploration order: 5
node: 6, exploration order: 6
```

Input 3

```
10 13
1 7
1 5
2 6
2 10
2 7
3 8
3 4
3 6
4 6
5 10
6 8
7 9
8 9
2 3
```

Output

```
2
2 6
6 3

Explored nodes in order:  2 6 10 7 3 4 8 5 1 9
node: 1, exploration order: 9
node: 2, exploration order: 1
node: 3, exploration order: 5
node: 4, exploration order: 6
node: 5, exploration order: 8
node: 6, exploration order: 2
node: 7, exploration order: 4
node: 8, exploration order: 7
node: 9, exploration order: 10
node: 10, exploration order: 3
```

Input 4

```
12 16
1 11
1 8
2 4
2 3
3 7
4 12
4 5
5 10
5 9
6 7
6 10
7 9
7 8
8 12
9 11
11 12
8 10
```

Output

```
3
8 7
7 6
6 10

Explored nodes in order:  8 1 7 12 11 3 6 9 4 2 10 5
node: 1, exploration order: 2
node: 2, exploration order: 10
node: 3, exploration order: 6
node: 4, exploration order: 9
node: 5, exploration order: 12
node: 6, exploration order: 7
node: 7, exploration order: 3
node: 8, exploration order: 1
node: 9, exploration order: 8
node: 10, exploration order: 11
node: 11, exploration order: 5
node: 12, exploration order: 4
```

4. Conclusion

Findings:

- This BFS Algorithm can be used to find the shortest path of all the nodes from a source of a graph.
- Time complexity for this Algorithm is $O(V + E)$, where V is vertices and E is edges.

Limitations:

- Whilst inserting the graph, the algo by default assumes that it will be an undirected and unweighted graph. Therefore, the Algorithm won't be applicable for weighted or directed graph.
- The Algorithm doesn't handle any exception that may occur while taking input.