Report-02
BFS Algorithom Poblem Solving

CSE-0408 Summer 2021

MD.SHERAJUDDAWLA SUMON BISWAS
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
sherajuddawlasumon@gmail.com

***Abstract*—As discussed earlier, Breadth-First Search (BFS) is an algorithm used for traversing graphs or trees. Traversing means visiting each node of the graph. Breadth-First Search is a recursive algorithm to search all the vertices of a graph or a tree. BFS in python can be implemented by using data structures like a dictionary and lists. Breadth-First Search in tree and graph is almost the same. The only difference is that the graph may contain cycles, so we may traverse to the same node again.**

***Index Terms*—BFS**

I. Introduction

Breadth First Traversal (or Search) for a graph is similar to Breadth First Traversal of a tree (See method 2 of this post). The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.

II. VARIANTS OF BEST FIRST SEARCH

The two variants of Best First Search are Greedy Best First Search and A* Best First Search.

Greedy BFS: Algorithm selects the path which appears to be the best, it can be known as the combination of depth-first search and breadthfirst search. Greedy BFS makes use of Heuristic function and search and allows us to take advantages of both algorithms.

A* BFS: Is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.).

III. ALGORITHM FOR BFS

Step 1: Choose the starting node and insert it into queue.
Step 2: Find the vertices that have direct edges with the vertex(node).
Step 3: Insert all the vertices found in step 3 into queue.
Step 4: Remove the first vertex(node) in queue
Step 5: Continue this process until all the vertices are visited.

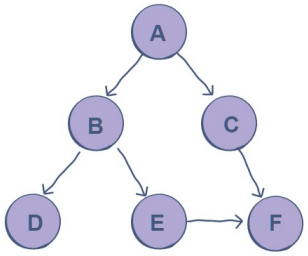

Fig. 1. Example of a BFS

```
In [12]: graph = {
           'A' : ['B','C'],
           'B' : ['D', 'E'],
           'C' : ['F'],
           'D' : [],
           'E' : ['F'],
           'F' : []
         }

         visited = [] # List to keep track of visited nodes.
         queue = []     #Initialize a queue

         def bfs(visited, graph, node):
           visited.append(node)
           queue.append(node)

           while queue:
             s = queue.pop(0)
             print (s, end = " ")

             for neighbour in graph[s]:
               if neighbour not in visited:
                 visited.append(neighbour)
                 queue.append(neighbour)

         # Driver Code
         bfs(visited, graph, 'A')
         A B C D E F
```

Fig. 2. Example of a BFS Algorithm

IV. CONCLUSION

The BFS algorithm is useful for analyzing the nodes in a graph and constructing the shortest path of traversing through these.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

REFERENCES

[1] Shoewu, O., & Idowu, O. A. (2012). Development of attendance management system using biometrics. The Pacific Journal of Science and Technology, 13(1), 300-307.

[2] Arulogun, O. T., Olatunbosun, A., Fakolujo, O. A., & Olaniyi, O. M. (2013). RFID-based students attendance management system. International Journal of Scientific & Engineering Research, 4(2), 1-9.