##### Introduction

**LAB # 4**

#### Breadth First Search

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the neighbor nodes first, before moving to the next level neighbours.

BFS and its application in finding connected components of graphs were invented in 1945 by Konrad Zuse, in his (rejected) Ph.D. thesis on the Plankalkül programming language, but this was not published until 1972. It was reinvented in 1959 by E. F. Moore, who used it to find the shortest path out of a maze, and discovered independently by C. Y. Lee as a wire routing algorithm (published 1961).

![A picture containing diagram

Description automatically generated]()

The above image depicts the end-to-end process of Breadth-First Search Algorithm. Let me explain this in more depth.

1. Assign ‘a’ as the root node and insert it into the Queue.
2. Extract node ‘a’ from the queue and insert the child nodes of ‘a’, i.e., ‘b’ and ‘c’.
3. Print node ‘a’.
4. The queue is not empty and has node ‘b’ and ‘c’. Since ‘b’ is the first node in the queue, let’s extract it and insert the child nodes of ‘b’, i.e., node ‘d’ and ‘e’.
5. Repeat these steps until the queue gets empty. Note that the nodes that are already visited should not be added to the queue again.

Now let’s look at the pseudocode of Breadth-First Search algorithm.

## **Breadth-First Search Algorithm Pseudocode**

Here’s the pseudocode to implement the Breadth-First Search Algorithm:

|  |  |
| --- | --- |
|  | Input: s as the source node  BFS (G, s)  let Q be queue.  Q.enqueue( s )  mark s as visited  while ( Q is not empty)  v = Q.dequeue( )  for all neighbors w of v in Graph G  if w is not visited  Q.enqueue( w )  mark w as visited |

In the above code, the following steps are executed:

1. (G, s) is input, here G is the graph and s is the root node
2. A queue ‘Q’ is created and initialized with the source node ‘s’
3. All child nodes of ‘s’ are marked
4. Extract ‘s’ from queue and visit the child nodes
5. Process all the child nodes of v
6. Stores w (child nodes) in Q to further visit its child nodes
7. Continue till ‘Q’ is empty

**A picture containing text, clipart

Description automatically generated**

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

**Applications Of Breadth-First Search Algorithm**

Breadth-first Search is a simple graph traversal method that has a surprising range of applications. Here are a few interesting ways in which Bread-First Search is being used:

**Crawlers in Search Engines:** Breadth-First Search is one of the main algorithms used for indexing web pages. The algorithm starts traversing from the source page and follows all the links associated with the page. Here each web page will be considered as a node in a graph.

**GPS Navigation systems:** Breadth-First Search is one of the best algorithms used to find neighboring locations by using the GPS system.

**Broadcasting:** Networking makes use of what we call as packets for communication. These packets follow a traversal method to reach various networking nodes. It is being used as an algorithm that is used to communicate broadcasted packets across all the nodes in a network.

##### Lab Tasks

**Task 01:** Implement Breadth First Search algorithm in python for the given graph.

![A picture containing clock

Description automatically generated]()

**Task 02:** Implement Breadth First Search algorithm in python for the given graph with the help of Networkx library.

![Diagram

Description automatically generated]()