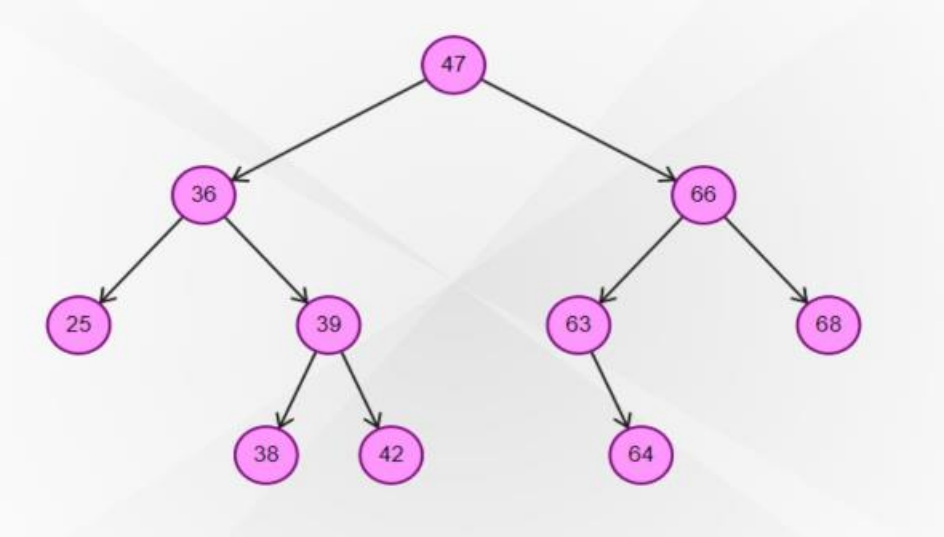
CSE-2204



BFS Algorithm Visualization

User Manual

May 13,2023

North Western University

Khulna, Bangladesh

Table of Contents

BFS Algorithm Visualization User Manual

1. Introduction

BFS, or Breadth-First Search, is a graph traversal algorithm used to explore and visit all the vertices of a graph in a breadthward motion. It starts at a given source vertex and systematically explores all the vertices of the graph that are reachable from the source vertex, one level at a time. BFS guarantees that it visits all the vertices in the graph at the same level of depth before moving on to the vertices at the next level.

Visualization of the BFS algorithm is a helpful way to understand its operation. It often involves representing the graph as a set of nodes (vertices) connected by edges. Each node is typically labeled with its level or distance from the source vertex. As the algorithm progresses, nodes are visited and marked, and the queue is updated accordingly. The visualization may include animations or step-by-step representations of the algorithm's execution, making it easier to comprehend the order in which the vertices are visited and the exploration of different levels.

What is BFS algorithm?

-BFS is a fundamental graph traversal algorithm and forms the basis for more advanced algorithms like Dijkstra's algorithm and A* search. Its simplicity and efficiency (O(V + E), where V is the number of vertices and E is the number of edges) make it a widely used technique in various applications, including network routing, web crawling, social network analysis, and more.

2. Objective

The objective of a BFS (Breadth-First Search) algorithm can be described briefly as follows:

1. Exploration of Graph: The primary objective of BFS is to explore and visit all the vertices of a graph. It systematically traverses the graph in a breadthward motion, visiting vertices at the same level of depth before moving on to the next level.

2. Finding Shortest Paths: BFS guarantees that it discovers the shortest path between the source vertex and any other vertex in an unweighted graph. By exploring the graph level by level, it ensures that the shortest path is found before exploring paths of greater length.

3. Connectivity Analysis: BFS can be used to determine the connectivity between vertices in a graph. By performing BFS from a source vertex, it identifies all the vertices that are reachable from that source, helping to find connected components or detect if the graph is connected.

4. Cycle Detection: BFS can detect cycles in a graph. If during the traversal, a visited vertex is encountered that is already marked as visited, then there exists a cycle in the graph. This property of BFS makes it useful for cycle detection and cycle analysis.

5. Bipartiteness Checking: BFS can also be employed to check if a graph is bipartite. A bipartite graph is one that can be divided into

two disjoint sets, such that all edges connect vertices from different sets. By assigning alternate colors to the vertices during BFS, if at any point, an edge is encountered that connects two vertices of the same color, then the graph is not bipartite.

6. Pathfinding in Unweighted Graphs: BFS can be utilized to find paths or routes between two vertices in an unweighted graph. By starting BFS from the source vertex and terminating the algorithm when the destination vertex is reached, the path between the two vertices can be reconstructed.

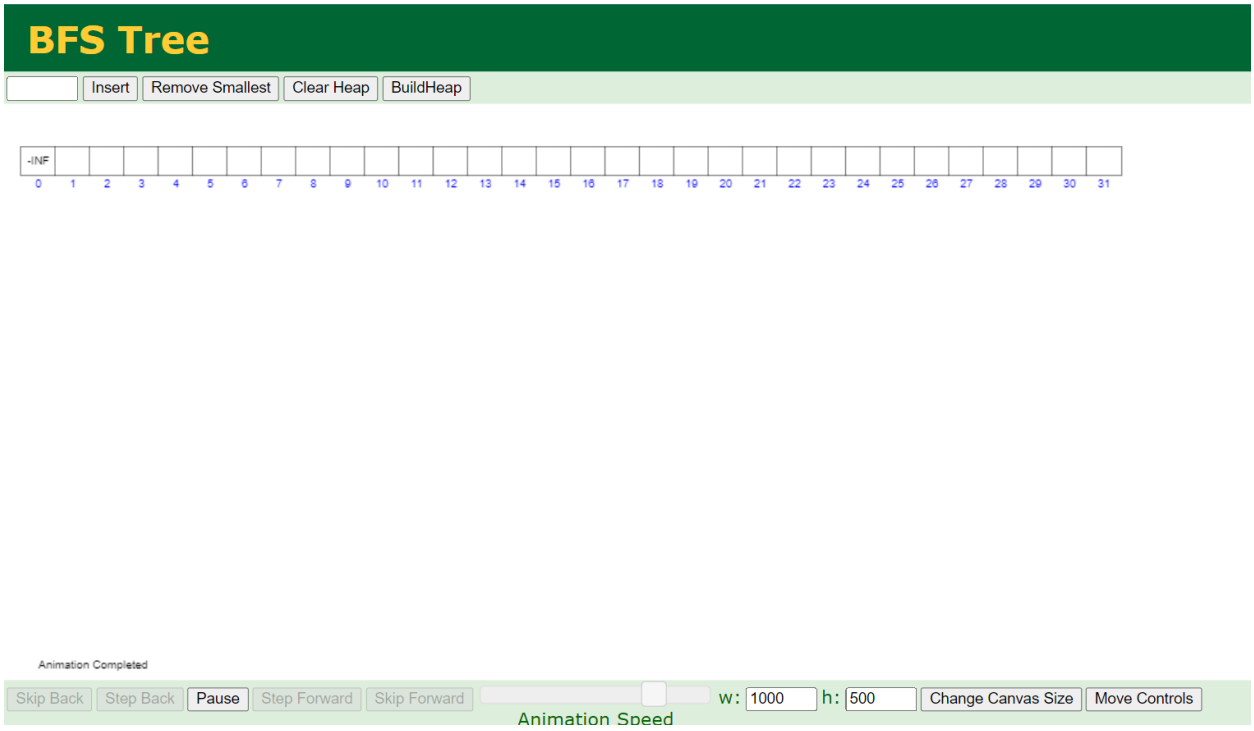
7. Tree and Graph Traversal: BFS is also a fundamental algorithm for tree and graph traversal. In tree structures, BFS can be used to traverse all the nodes in level order, visiting sibling nodes before moving on to their children. In a graph, BFS helps to systematically visit all the vertices, exploring the graph in a breadth-first manner.

Overall, the main objectives of a BFS algorithm include exploring the graph, finding shortest paths, analyzing connectivity, detecting cycles, checking bipartiteness, pathfinding, and facilitating tree and graph traversal.

3. Description

Home Page:

This the home page of BFS Algorithm Visualization Software.



This is the BFS Algorithm visualization elements from the user. Here we enter some values as input for BFS tree. Firstly we would give a input and then press 'Insert' button. It would arrange the input manually as a BFS tree. After given all the inputs, it arranged a full BFS tree for those input.

Test Case Input:

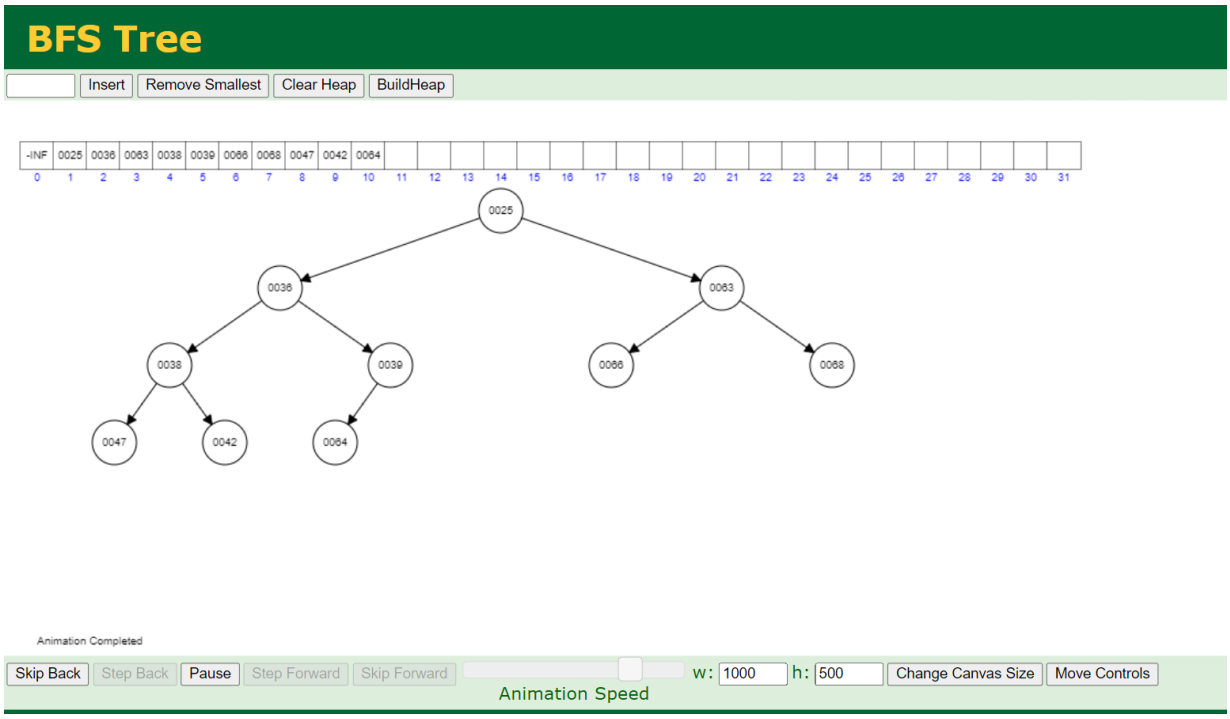
The algorithm employs a queue data structure to keep track of the vertices that need to be visited. It follows the FIFO (First-In-First-Out) principle, where the vertices are inserted at the end of the queue and removed from the front. The BFS algorithm proceeds as follows:

1. Begin with a source vertex and enqueue it into the queue.
2. Mark the source vertex as visited.
3. Dequeue a vertex from the front of the queue and visit it.
4. Enqueue all the unvisited neighboring vertices of the current vertex.
5. Mark the visited vertices as visited and repeat steps 3-5 until the queue becomes empty.

By exploring the graph level by level, BFS ensures that the shortest path between the source vertex and any other vertex is found, making it particularly useful for finding the shortest path or distance in unweighted graphs. It can also be used to solve various graph-related problems like finding connected components, detecting cycles, or checking bipartiteness.

Algorithm Visualization:

This is how BFS Algorithm Visualization Software visualizes the algorithm.



This is BFS algorithm visualization software. We inserting number one by one and it arranged a BFS visualization tree. Here we see that some numbers are given as input and after insert all the number it arranged a BFS visualization tree by manually.

4. Dependencies

1. HTML: The HTML code is used to create the user interface for the BFS algorithm visualization software. The code creates a web page with two sections: the insert section and the visualization section. In the input section, there are a input boxes for the user to enter the node value of the BFS graph. The user can then click the "Insert" button to execute the BFS algorithm. In the visualization section, the graph visualization is displayed that generated by the JavaScript code. The HTML code is linked to a CSS file for styling and a JavaScript file for implementing the BFS algorithm and the visualization.

2. CSS: The CSS code in the provided snippet is used to style and animate the elements of a visualization software that displays the BFS algorithm on a graph. The code styles the background color, font family, and color of the text. It also defines the layout of the container that holds the input and visualization sections, as well as the styles for the input section and the graph visualization section. The animation code is used to animate the edges, vertices as the algorithm progresses. The animation also highlights the connections between the vertices and edges, and displays the weight of each edge during the animation. Afterall, the CSS code is used to enhance the visual representation of the BFS algorithm on a graph, making it easier to understand and follow.

3. JavaScript: JavaScript takes input values from HTML input elements, runs the algorithm, and displays the graph and results.

Special Thanks to:

Md. Shymon Islam

Lecturer

Department of CSE

North Western University

Khulna, Bangladesh

Developed by:

Humaira Khatun

Student ID: 20212035010

Sadia Parvin

Student ID: 20212037010

Sanzida Sazzad Khushi

Student ID: 20212040010