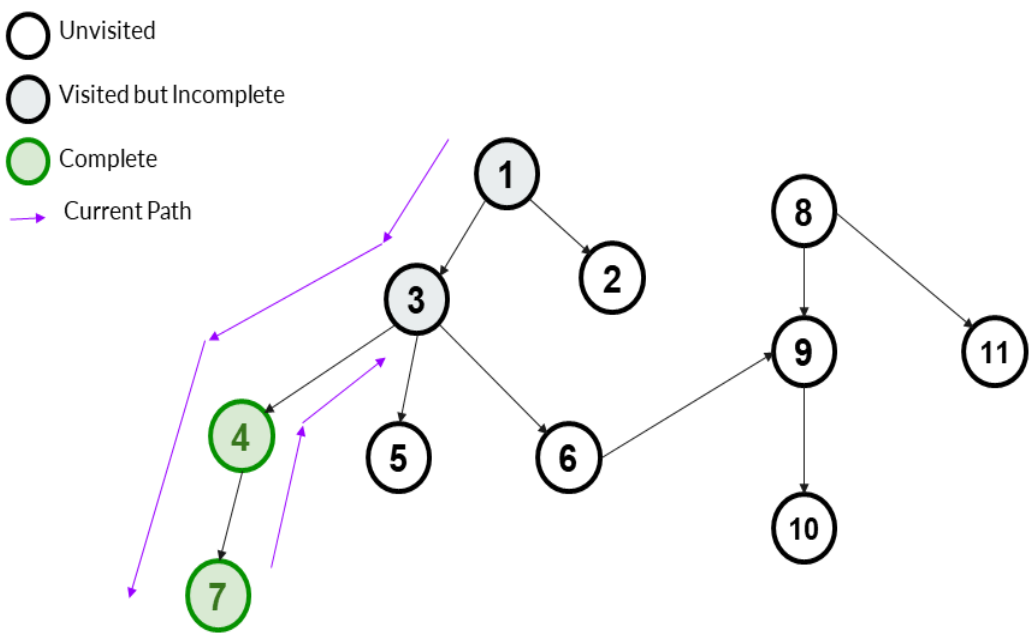


DFS VISUALISATION USER MANUAL

Md Shahariaz Anwar [20212002010]

Humairatul Bushra [20212033010]

Fatema Tuj Johra Lima [20212047010]

North Western University

12/05/2023

Table of Contents

Introduction

The DFS algorithm is a popular graph traversal algorithm used to explore and search through a graph data structure. It stands for Depth-First Search, and as its name suggests, it starts from the root node and explores as far as possible along each branch before backtracking. This algorithm is widely used in various applications such as network discovery, maze solving, and topological sorting.

Visualizing the DFS algorithm can be helpful in understanding how it works and how it explores the graph. By using animations and interactive tools, it's possible to see how the algorithm progresses step-by-step, how it chooses its path, and how it marks visited nodes. Visualizations can also help to identify potential issues in the algorithm, such as infinite loops or incorrect path selection. Overall, visualizing the DFS algorithm can be a valuable tool for both students and professionals who want to learn or improve their understanding of this essential algorithm.

In this article, we will explore how DFS works and provide a step-by-step visualization of the algorithm. We will also discuss the different applications of DFS and how it compares to other graph traversal algorithms.

Objectives

Here are some possible objectives for visualizing the DFS algorithm:

1. Demonstrate the steps of the DFS algorithm: The visualization should show the order in which vertices are visited during the DFS algorithm, as well as the status of each vertex (visited, unvisited, or in progress) at each step. This will help users understand how the DFS algorithm works and how it explores the graph.

2. Highlight the discovery and finishing times of each vertex: The visualization should also show the time at which each vertex is first discovered (i.e., when it is first visited during the DFS traversal) and the time at which it is finished (i.e., when all of its neighbors have been explored). This will help users understand how the DFS algorithm creates a depth-first search tree and how it can be used to classify edges in the graph.

3. Illustrate the different types of edges: The DFS algorithm can classify edges in a graph as tree edges, back edges, forward edges, or cross edges, depending on the order in which they are explored. The visualization should illustrate these different types of edges and show how they are identified during the DFS algorithm.

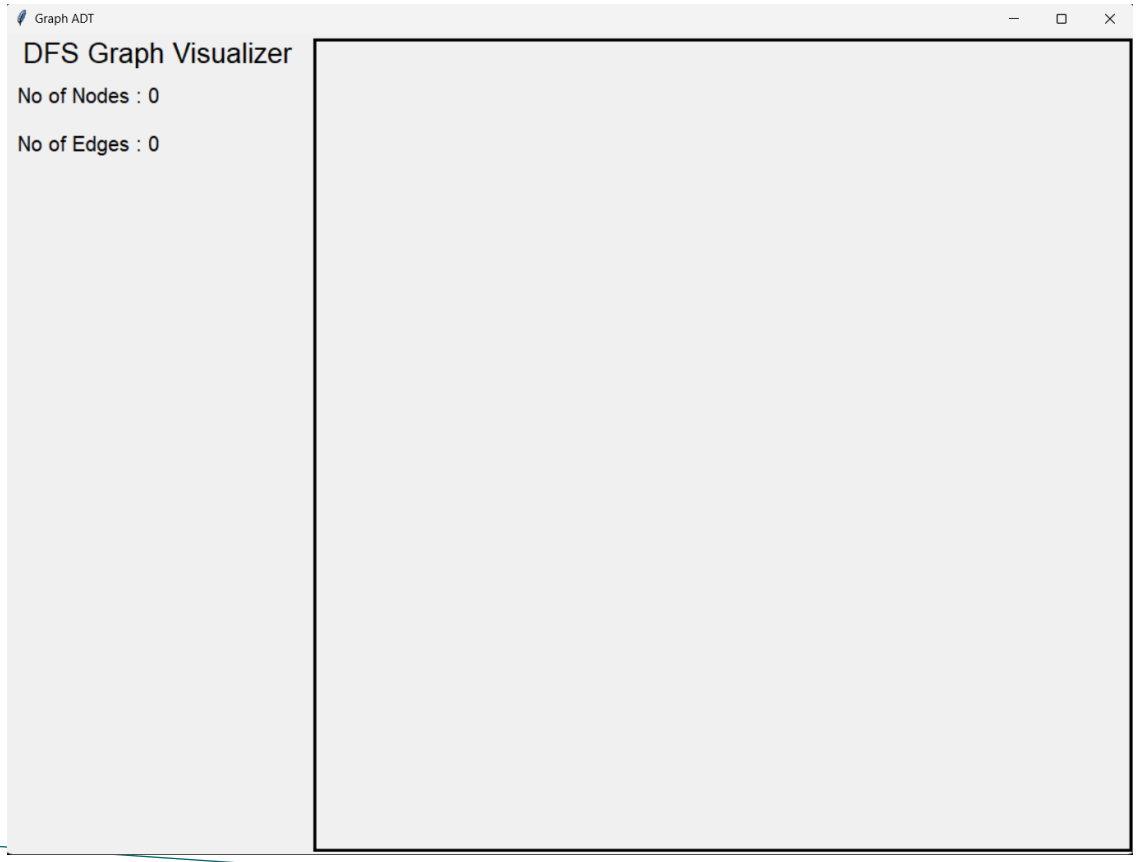
4. Allow users to interact with the visualization: To help users engage with the DFS algorithm and explore its behavior in different contexts, the visualization should allow users to interact with the graph, selecting different starting vertices, changing the order in which vertices are visited, or modifying the graph itself. This will make the visualization more engaging and useful as a teaching tool.

5. Provide context and guidance: Finally, the visualization should provide context and guidance for users who are new to the DFS algorithm or to graph theory more generally. This might include explanatory text, annotations, or links to additional resources that can help users deepen their understanding of the algorithm and its applications.

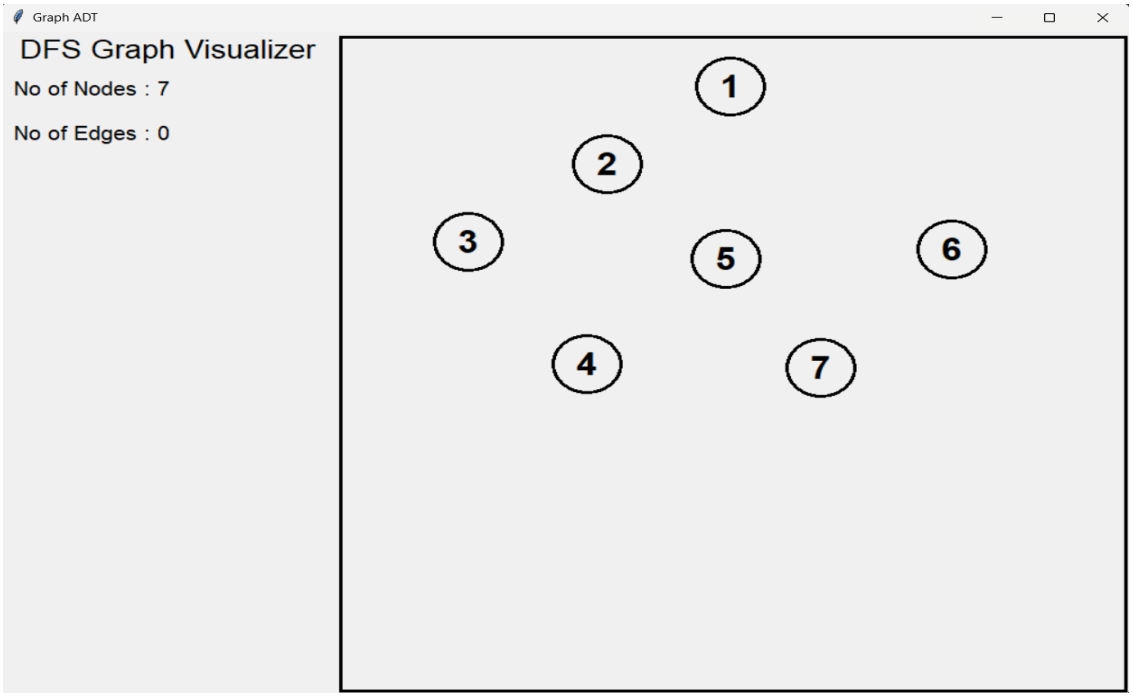
Description

This code is a python implementation of a graph visualization tool using the tkinter library. It creates a canvas where nodes and edges are drawn in real-time. The user can add nodes to the canvas by clicking on a location, and edges can be created between two nodes by clicking on them in succession.

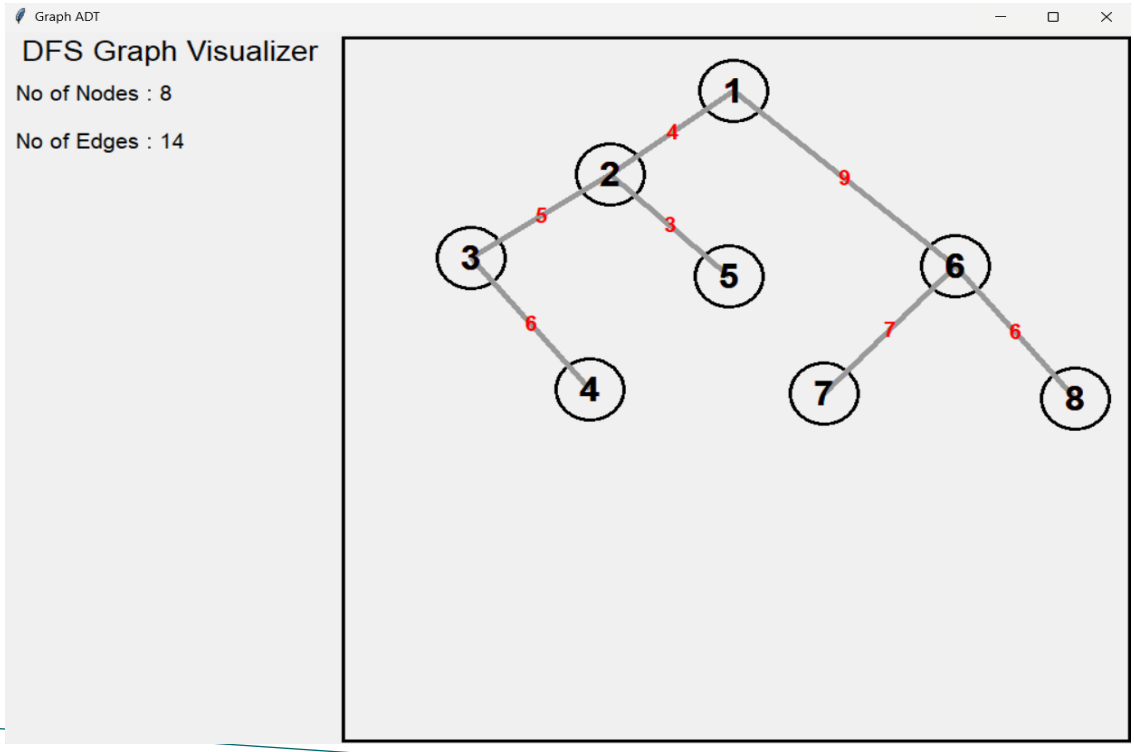
This is the home page of DFS algorithm



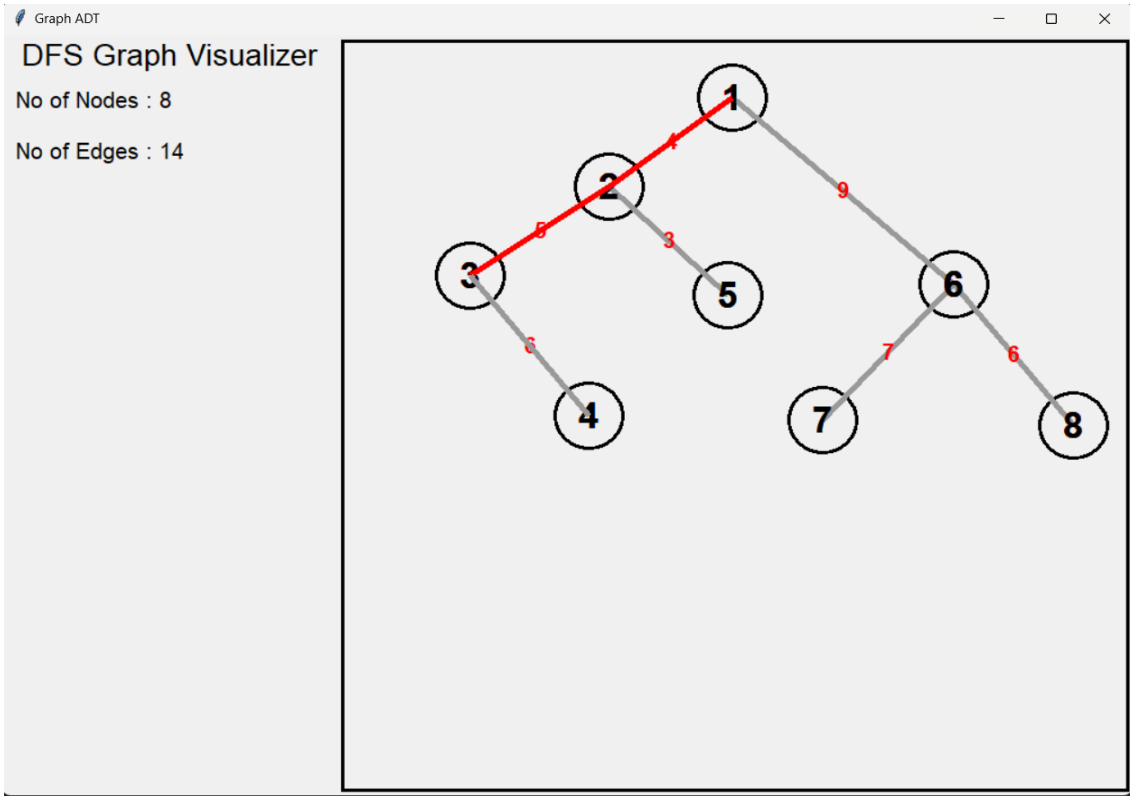
Click on the left side of the mouse to take node input. Here we have input some nodes



By clicking right site of the mouse connect a node to a another node and simultaneously inputting the distance from one node to another node.



If you press DFS on the keyboard, the nodes are visited in depth wise. That means the DFS algorithm is working properly.



Dependencies

The code dependencies are:

- `tkinter` library, imported with alias `tk`: It is used for creating graphical user interfaces in Python. The code uses it to create a window for displaying the graph visualization.

- `math` library: It provides mathematical functions for the program, used to compute the Euclidean distance between nodes.

- `time` library: It is not used in the code and can be removed.

- `Canvas` class from `tkinter` library: It provides the canvas where the nodes and edges are displayed.

- `ttk` class from `tkinter` library: It provides the GUI elements, such as `Label`, used to display the number of nodes and edges.

Special Thanks to:

Md. Shymon Islam

Lecturer Department Of CSE

North Western University

Khulna, Bangladesh

Developed By:

Md Shahariaz Anwar [20212002010]

Humairatul Bushra [20212033010]

Fatema Tuj Johra Lima [20212047010]