Acad. Year 2021-22

**Traversal Visualization Application**



**Traversal Visualization Application**

**Abdul Aziz Farid**

**Hamra Tariq**

**Faculty of Information Technology Department of Computer Science**

**Salim Habib University, Karachi, Pakistan**

**Year 2021/22**

Submitted by

ABDUL AZIZ FARID (F18CSC11)   
FASIH HUSSAIN (F18CSC16)   
HAMRA TARIQ (F18CSC47)

Faculty of Information Technology Department of Computer Science

Machine Learning Course Project Report Presented to

**Mr. Waseem and Ms. Samita** Faculty of Information Technology Department of Computer Science

Salim Habib University, Karachi, Pakistan

Year 2022

Table of Contents

[1. Introduction 4](#_Toc96876042)

[2. Methodology 4](#_Toc96876043)

[3. Dependencies 5](#_Toc96876044)

[4. Source Code 5](#_Toc96876045)

[5. Output and Results 5](#_Toc96876046)

# Introduction

In a world dominated by visualization, filters and effects, its also easier to learn using pictures and visual effects rather than textbooks and boring videos. In our course project of Artificial Intelligence, we aim to build an application that will be able to visualize the working of uninformed search algorithms. This project is specifically targeted for teachers aiming to give a better understanding of how the different algorithms work to find the goal node. It allows the students to witness the flow of the traversal in a live simulation instead of slides, hence boosting the understanding of students.

# Methodology

We have developed a basic application of an uninformed search where user will get the visualization for the map of Romania where they can enter the start and goal node and see the visualization from the start node to the next node all the way to the destination according to the algorithm chosen (BFS/DFS).

To walk through the methodology of implementation, first step is to install the dependencies listed in *Section 4*

*.*

Once all the dependencies are in place, we have first used Tkinter library to instantize a GUI window for all the components to be in place. In this window, we have made a canvas to display all the shapes (nodes), texts (labels) and buttons.

Once the canvas has been instantized we have set up a few basic settings, such as the position and properties of buttons, labels and have loaded the map of Romania ranging from Arad to Bucharest through shapes provided by Tkinter all of this is done in the “initialize\_vertex” method in our code. Our program now consists of 6 buttons providing different functionalities and 31 nodes representing the cities of Romania.

To connect these nodes, we’ve made a custom function “draw\_connector” which draws a line from node 1 to node 2 provided in the arguments. To ensure this isn’t just being depicted visually but our program also understands this connection, we have another function “connect\_connector” which in laymen’s terms basically tells the program all the children a particular node has.

Next, we make a method for each of these buttons to perform the functionality.

The “Start” button enables the user to select a single starting node which would be used as the root node for the traversal using the different algorithms. Similarly, the “Goal” button enables the user to select the goal node for the algorithm which would be our destination.

One of the major issues we had while implementing the start/goal system, was making the program have the optimal user experience. To do this, we went an extra mile to achieve the functionality of the Start and Goal button remaining pressed until a node has been selected to show and guide the user through the process of selection of nodes.

To simplify things, instead of selecting nodes through old school methods, we once again went an extra mile and come up with a logic that allows the user to click on the node that they wish to select as the starting or the destination node. This functionality has been written in the “clicked” method in the code attached.

Another feature we added was to find the shortest path to a goal city instead of the user selecting a goal node. For example, if the user wishes to find the shortest path from Arad to Bucharest using DFS and BFS, instead of selecting the specific Bucharest (which appears multiple times due to multiple routes), the user can simply select Bucharest from the drop down of all cities in our map of Romania, and choose their algorithm which would then traverse according to the algorithm until the first time (shortest path) the goal city appears.

Finally, to actually perform the traversal, the BFS and DFS button each implemented with their own logic simply does the magic. Both of these methods, work by accessing the start and goal node set by the user, and start traversing through Queues and Stack respectively to achieve the desired order of traversal. If in case the user does not select the start and goal node, the program would traverse the entire map of Romania

# Dependencies

* Tkinter
* Time
* Numpy

# Source Code

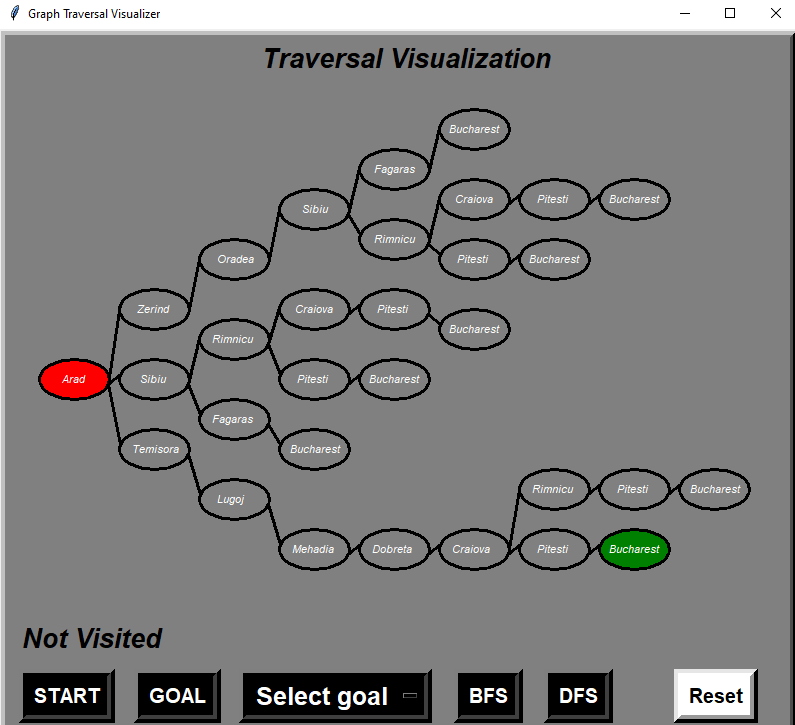
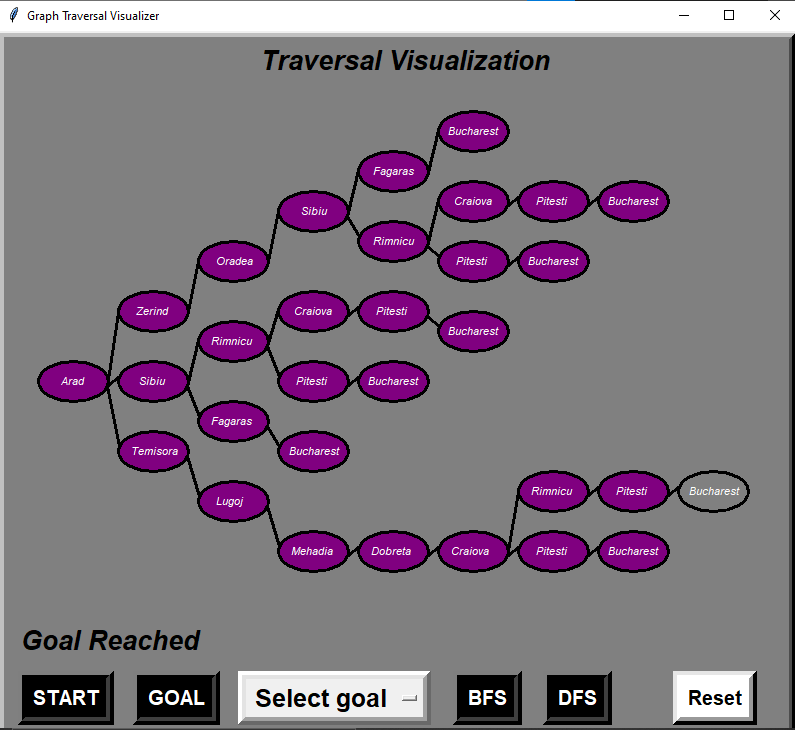
The entire code has been kept up to date on our github repository which can be accessed by clicking [here](https://github.com/hamra-siddiqui/traversal-algorithm).

# Output and Results

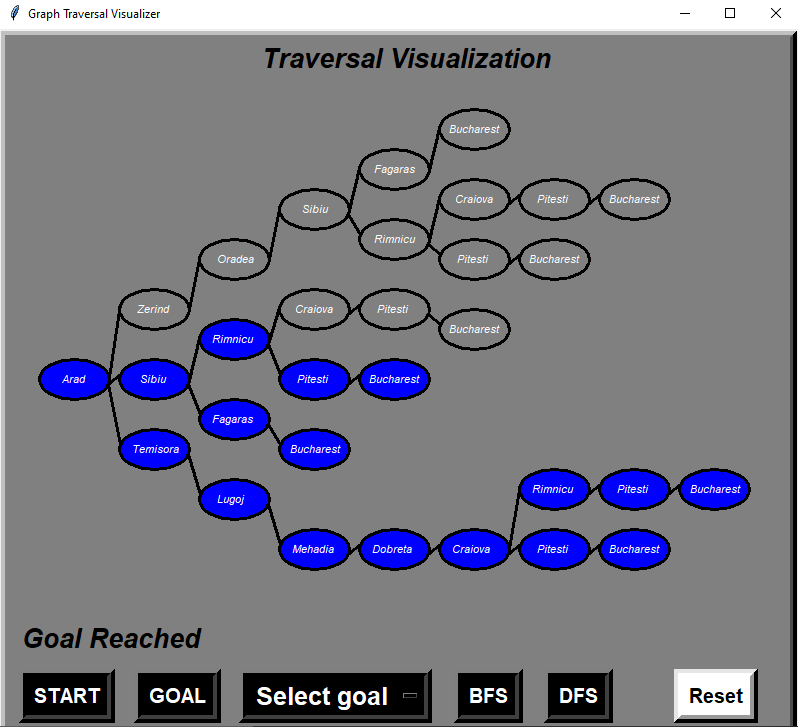
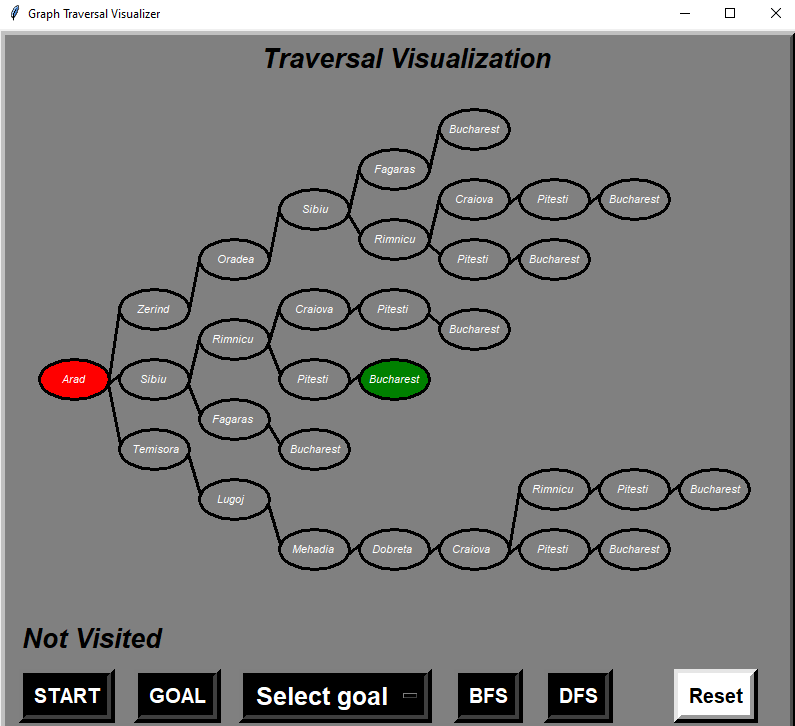
The results of our project came out good. We successfully implemented BFS and DFS with visualization given correct results.

1. Arad to Bucharest selected via clicks (BFS/DFS)

* BFS:

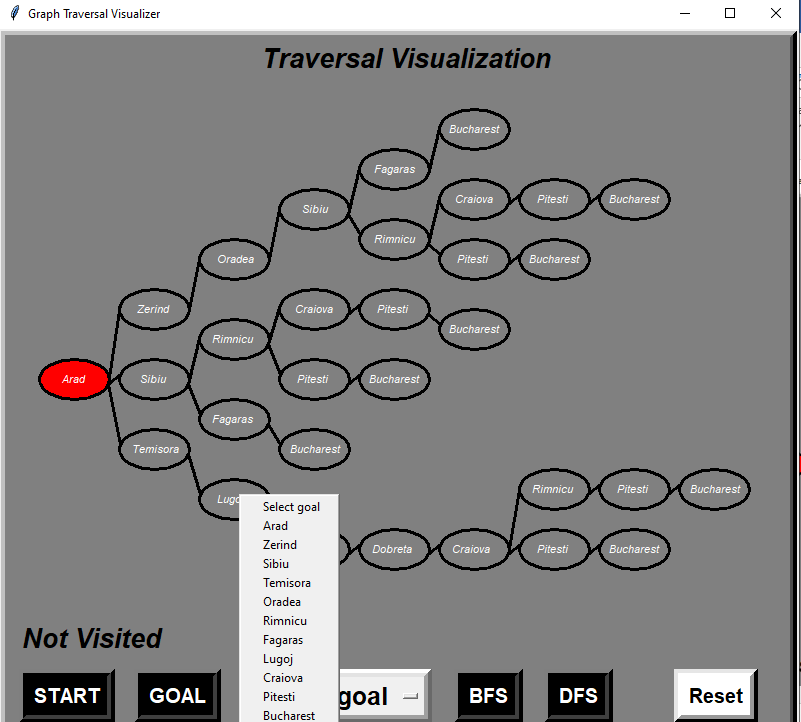
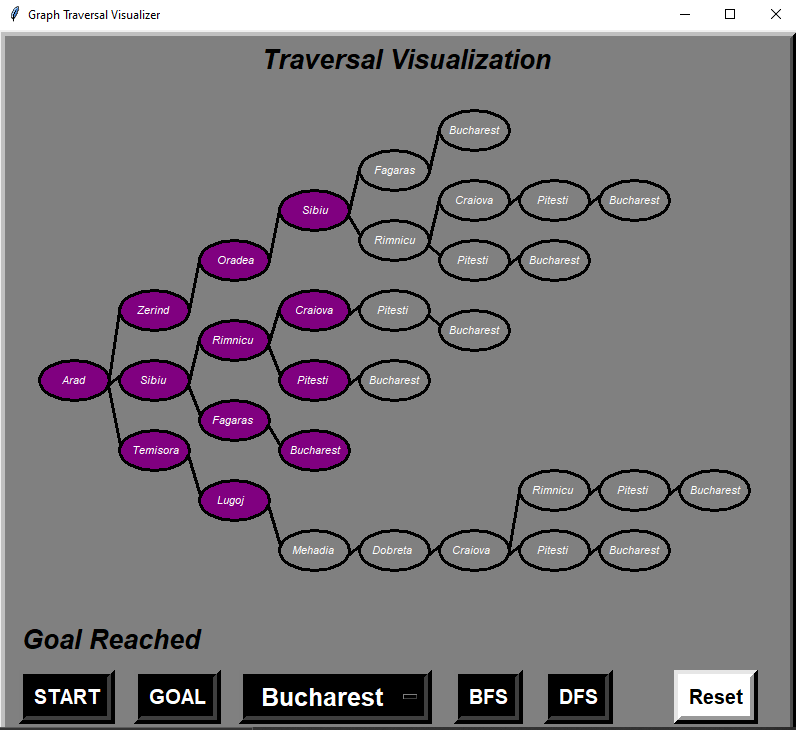
 

* DFS:



1. Arad to Bucharest via city selected from dropdown (BFS/DFS)

* BFS

* DFS

