**Chapter 1**

**INTRODUCTION**

Algorithms and data structures as an essential part of knowledge in a framework of computer science have their stable position in computer science curricula, since every computer scientist and every professional programmer should have the basic knowledge from the area . With the increasing number of students in Central European higher education systems in last decades , introduction of appropriate methods into the process of their education is also required. Our scope here is the higher education In the field of computer science. So, within the paper, we discuss the extension of standard methods of teaching algorithms,using the whiteboard or slides, with the algorithm visualizations. According to they can be used to attract students' attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization effectiveness are available nowadays, and the results are quite encouraging.

**Chapter 2**

**Problem Defination**

**2.1 Problem Statement:**

* To ensure that the visualization clearly conveys the concept.
* Automating the visualization process.
* Ensure that the students are able to experiment with and learn data structures in their own way

**2.1.1 Goals and Objectives:**

The motivation behind this project is to study how the operations on data structure are performed. So that students can learn various algorithms through animation. To get a clear knowledge about various data structures and their operations on it. It will make data structures learning more interesting. The main goal of this project is to implement a system for various sorting algorithms ,Path Finding, and Binary search tree makes learners understand how it works.

This project is for educational purposes. The main objective of this project is to help beginners to be able to visualize the basic algorithms and get a better understanding of the underlying operations. And obviously, it is needless to say that anyone who is willing to contribute is invited to use their creativity in making the visualizations even better and more attractive. One can add fresh algorithms and visualization of their choice too.

**2.1.2 Statement of Scope:**

In algorithm visualizer is developed to help the student to learn and study the Data Structure Algorithms.

**Chapter 3**

**System Requirement**

* 1. **Software Specification:**

1. Operating System:Windows 10 Pro 21H2.
2. Programming Language: JavaScript,HTML,Css.
   1. **Hardware Specification:**

1. CPU Speed:
2. RAM:

**Chapter 4**

**Feasibility Study**

A feasibility study was an evaluation of a proposal designed to determine the difficulty in carrying out designed tasks.Generally, a feasibility study precedes technical development and project implementation.

* **Economic Feasibility:**

To develop the proposed system, development costs be affordable . When implemented in real life, it does not need any maintenance.so the project is economically feasible.

* **Technical Feasibility:**

This website is specially made for desktop browsers.So, the website does not face any compatibility issues.Hence, the system is technically feasible.

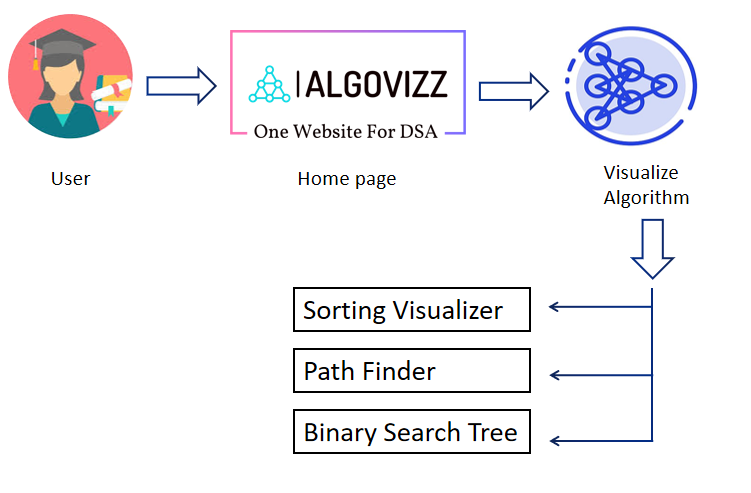
* **Behavioral Feasibility:**

Behavioral feasibility determines how much effort will go into the proposed system and into educating and training the users on the new system.The user interface is very simple and easy to understand. If a user finds any feature difficult to use, the Help option is easily accessible.So there is no training required for using the Algorithm Visualizer

**Chapter 5**

**System Design**

**5.1 Architecture:**

****

**Fig.5.1 Architecture**

**5.2 Modules:**

**Module 1: Homepage**

1. Tabs to choose between the three modules.

**Module 2: Sorting Algorithm**

1. Change the size of array.
2. Adject the speed of visualization.
3. Generate the new array.
4. Bubble,Insertion,Selection,Merge,Quick,Heap Sort.
5. Create the user define array.
6. Complexity for the algorithm.
7. The array is represented in the form of a (Downward) bar graph.

**Module 3: Pathfinder**

1. Grids/obstacles create function.
2. Clear grid.
3. dijkstra's ,A\*, BFS, DFS Algorithms.
4. Visualize the path finding algorithm.

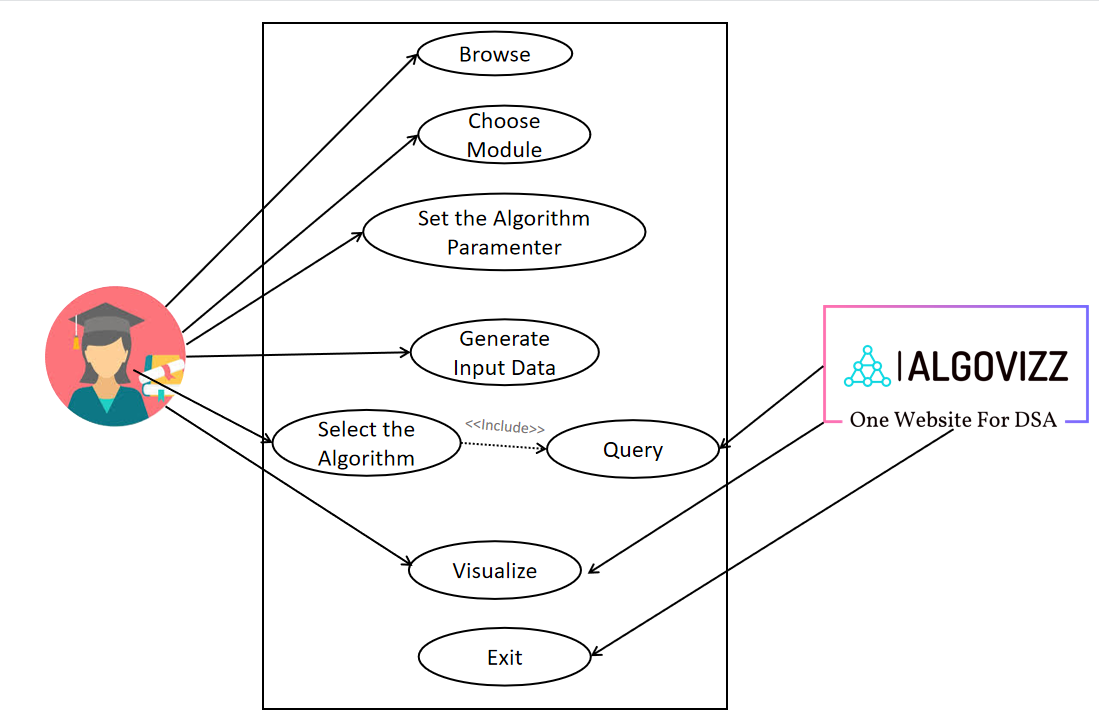
**Module 4: Binary Search Tree**

1. Insert Node.
2. Delete Node.

**5.3 Diagrams:**

**5.3.1 Use Case Diagram:**

The Use-case diagram shows the main interactions between the user and the algorithm visualizer. The user starts by selecting an algorithm to visualize and setting any necessary parameters. They then generate input data and execute the algorithm step-by-step, with the visualizer showing the state of the algorithm at each step. Finally, the user can view the complete execution and exit the visualizer.



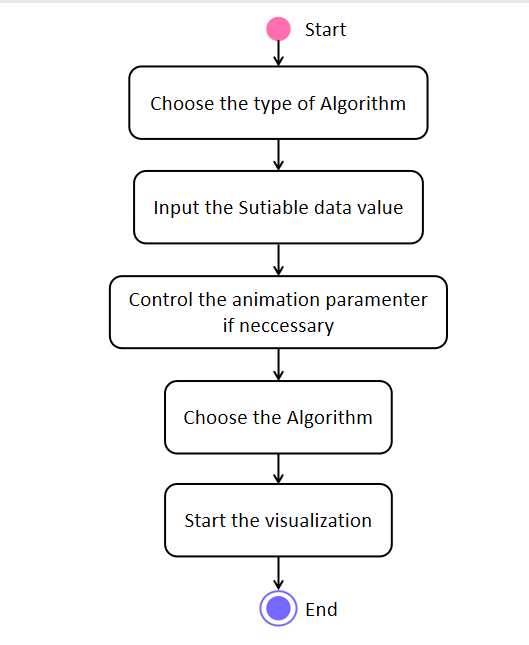
**Fig.5.3.1:Use Case Diagram**

**5.3.2 Class Diagram:**

This class diagram depicts the main classes and their relationships in an algorithm visualizer. The Algorithm class represents the algorithm being visualized, with name and parameters attributes and a input data list. The AlgorithmVisual class manages the execution of the algorithm, with a algorithm attribute and methods to set input data, parameters, and execute the algorithm step by step. The State class represents the current state of the algorithm, with attributes for data, position, and a boolean for whether the algorithm has finished executing. Finally, the AlgorithmFactory class can be used to create instances of different algorithms to visualize.

**5.3.3 Activity Diagram:**

This activity diagram shows the main steps involved in visualizing an algorithm. The user interacts with the algorithm visualizer to initialize the algorithm, set parameters, and input data. Then the algorithm is generated, executed, and the state is updated. If the algorithm is finished, the visualizer displays the final state and exits. If not, the visualizer updates the state and continues executing the algorithm.



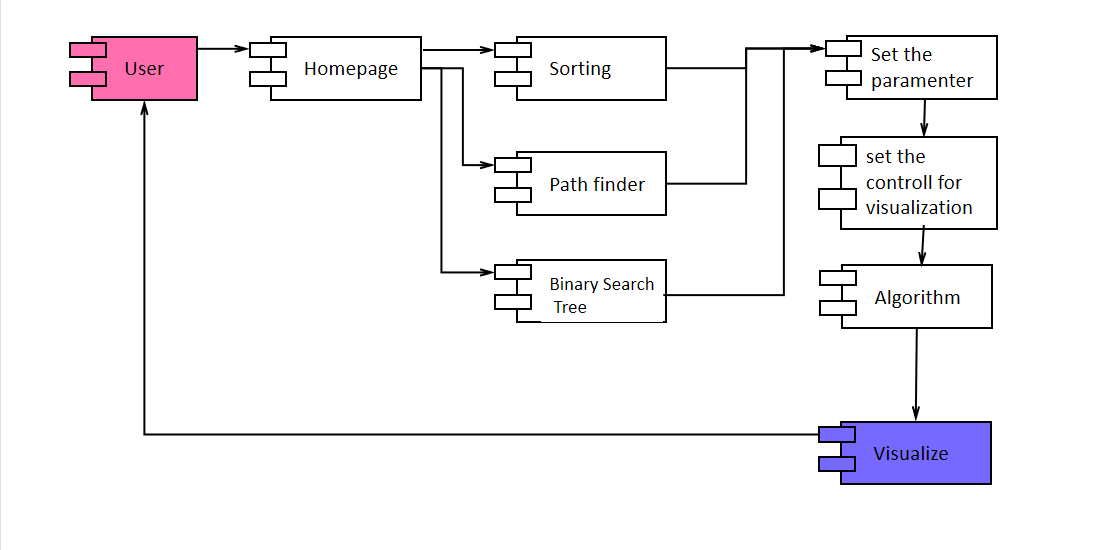
**Fig.5.3.3:Activity Diagram**

**5.3.4 Sequence Diagram:**

This sequence diagram shows the sequence of messages and interactions between the user and the algorithm visualizer. The user initializes the algorithm, sets parameters, and input data. Then, the visualizer generates the input, executes the algorithm, updates the state, and checks if the algorithm has finished. If it hasn't finished, the visualizer updates the state and continues executing the algorithm. Once the algorithm is finished, the visualizer displays the final state and exits.

**5.3.5 Component Diagram:**

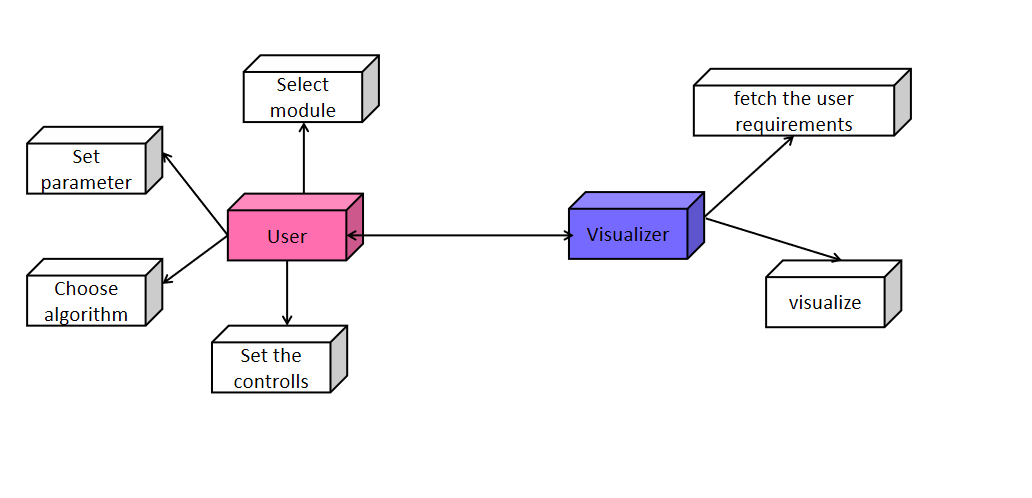
This component diagram shows the main components and their relationships in an algorithm visualizer. The Client component interacts with the AlgorithmVisual component to execute algorithms with input data and parameters. The AlgorithmVisual component manages the algorithm execution and updates the state. The Visualizer component displays the state using the Renderer component, which renders the state as a visual representation. The algorithm component represents the algorithm being executed, with name and parameters attributes and a input data list. The AlgorithmVisual component creates an instance of the appropriate algorithm to execute based on user input, and updates its state accordingly.



**Fig.5.3.5: Component Diagram**

**5.3.6 Deployment Diagram:**

This deployment diagram shows the components and their physical deployment in an algorithm visualizer system. The User Machine component represents the machine or device used by the user to interact with the algorithm visualizer. The Algorithm Visuaizer App component is the application that runs on the user's machine and provides the user interface for interacting with the algorithm visualizer. The Algorithm Server component is the server that executes the algorithms and provides the results to the Algorithm Visualizer App component. The Algorithm component represents the algorithms that are executed on the Algorithm Server, and the Renderer component renders the visual representations of the algorithm's state.

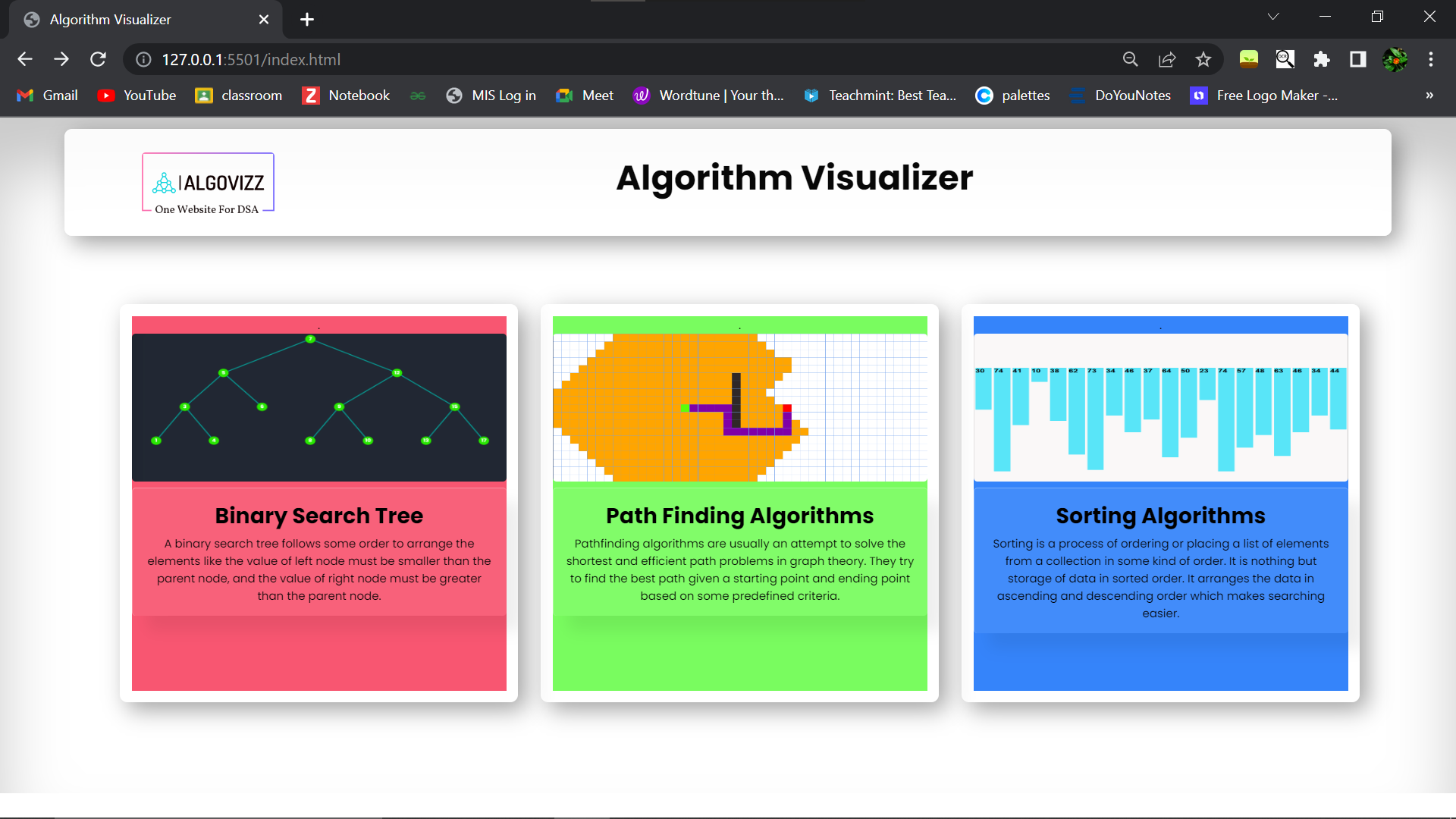


**Fig.5.3.6.Deployment Diagram**

**Chapter 5**

**Screenshots**

* **Home Page:**

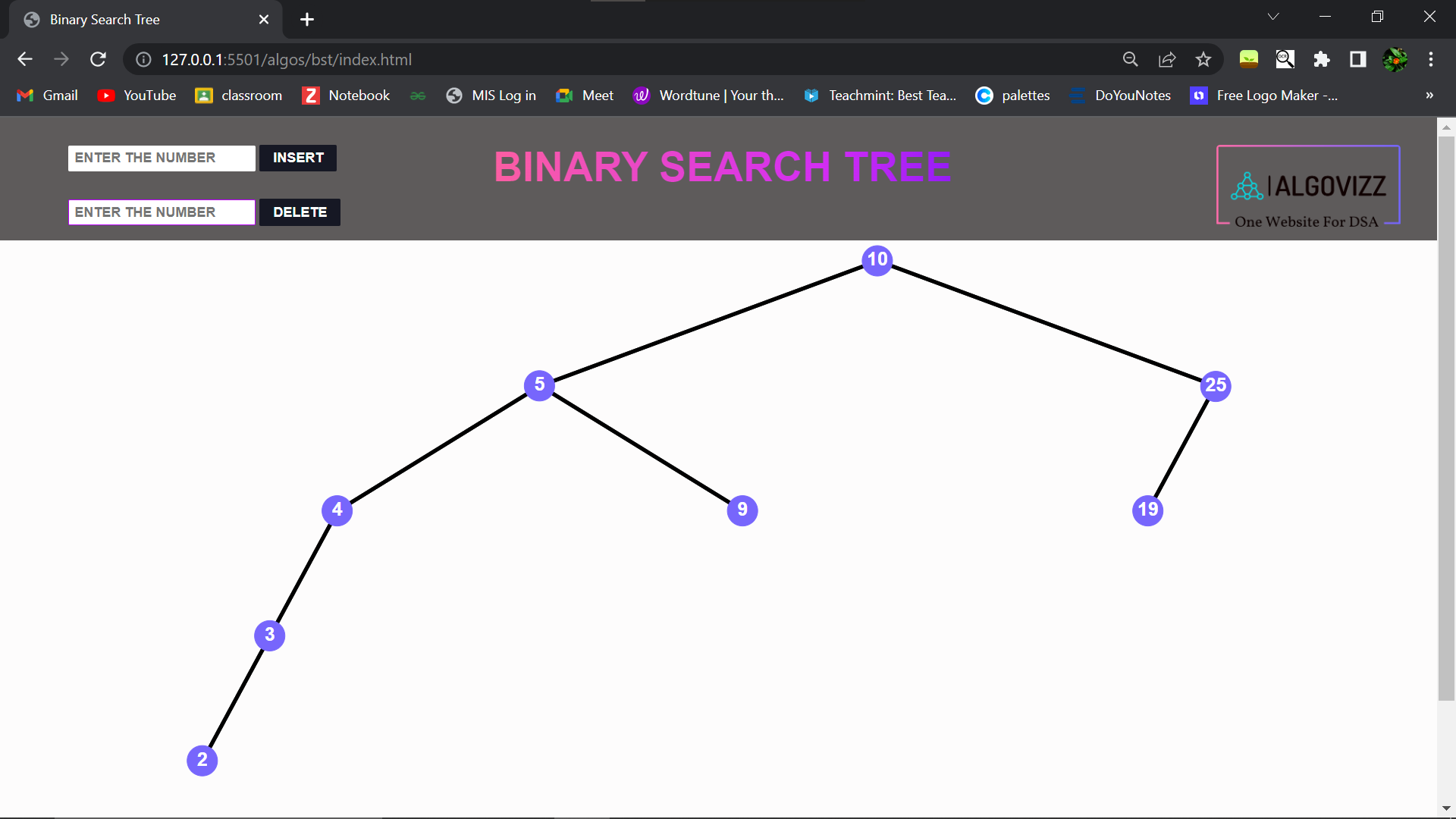
****

* **Binary Search Tree:**

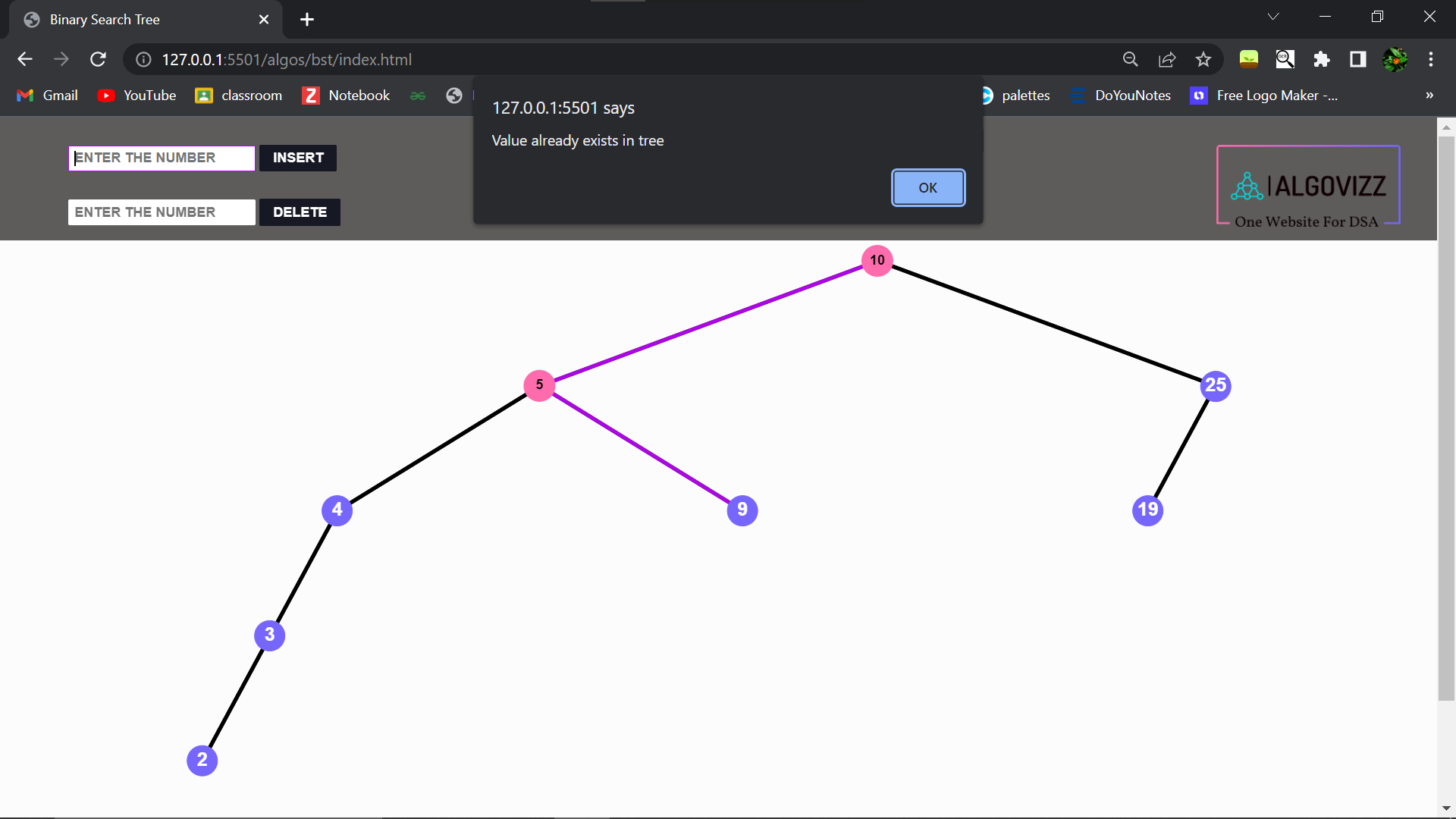
Blank Canvas

****

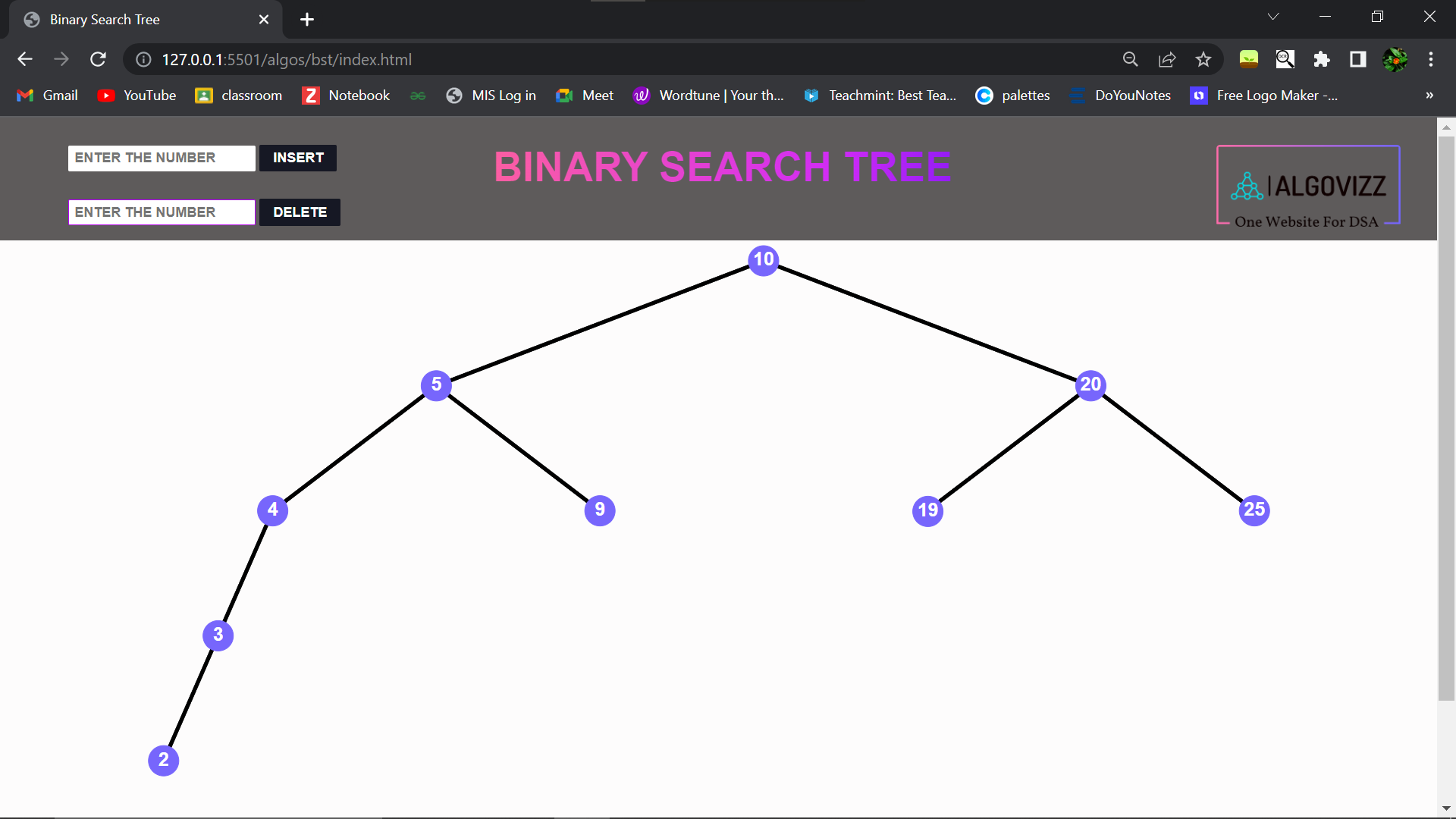
After Inserting a few nodes:

****

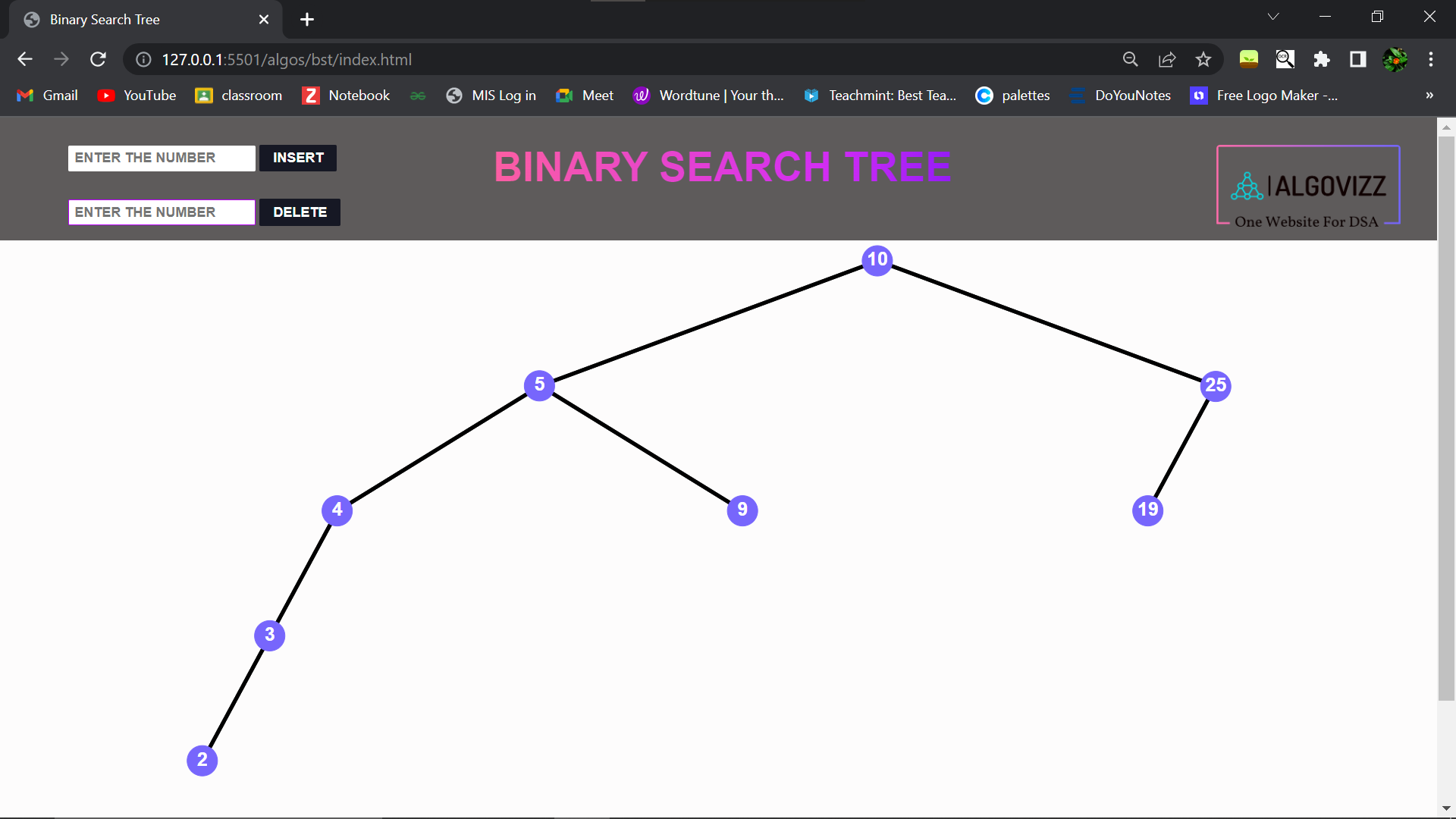
Already exiting error:



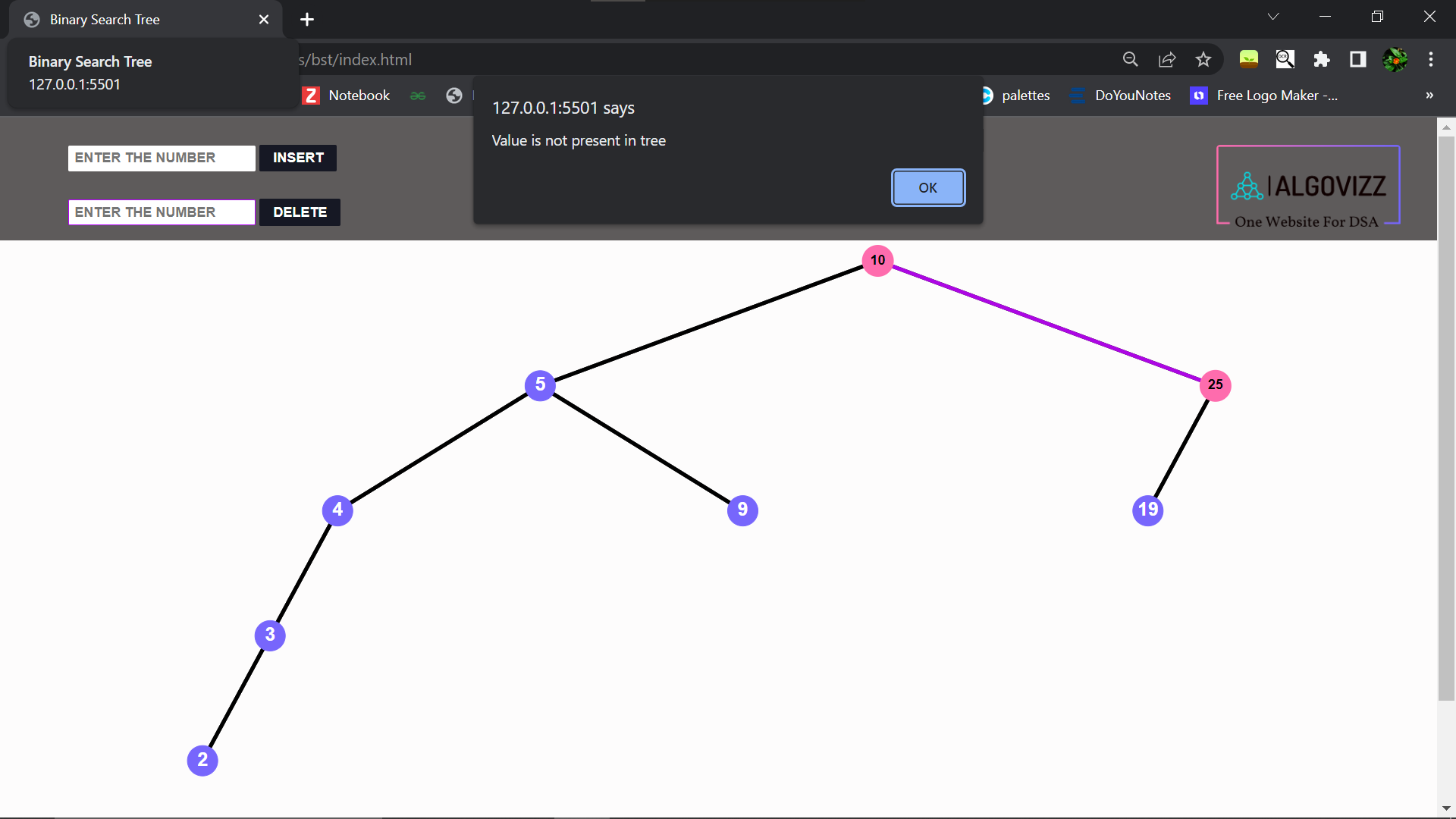
Deleting node with one children node:



Deleting the node with two children node:

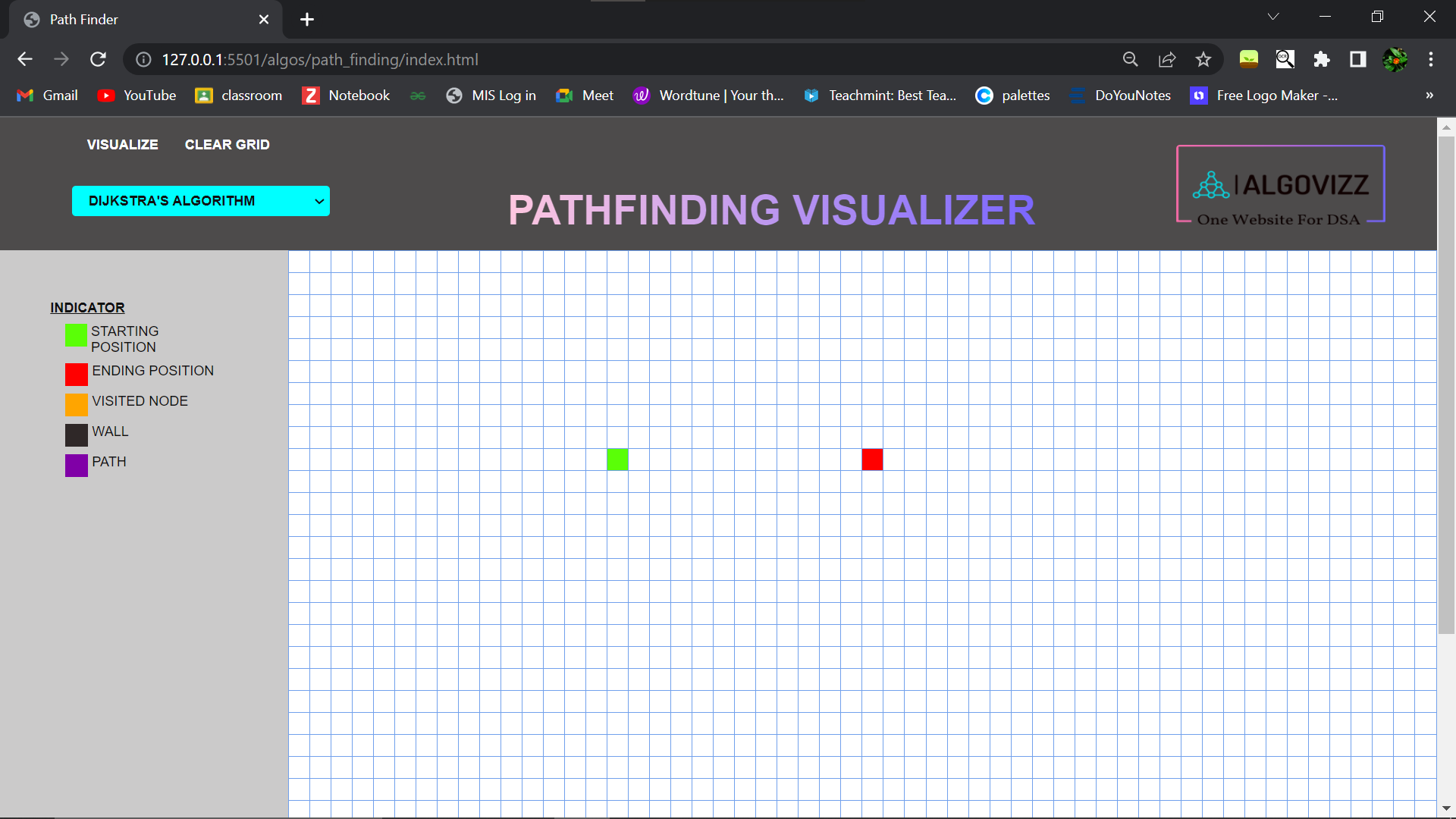


Node not Present error:

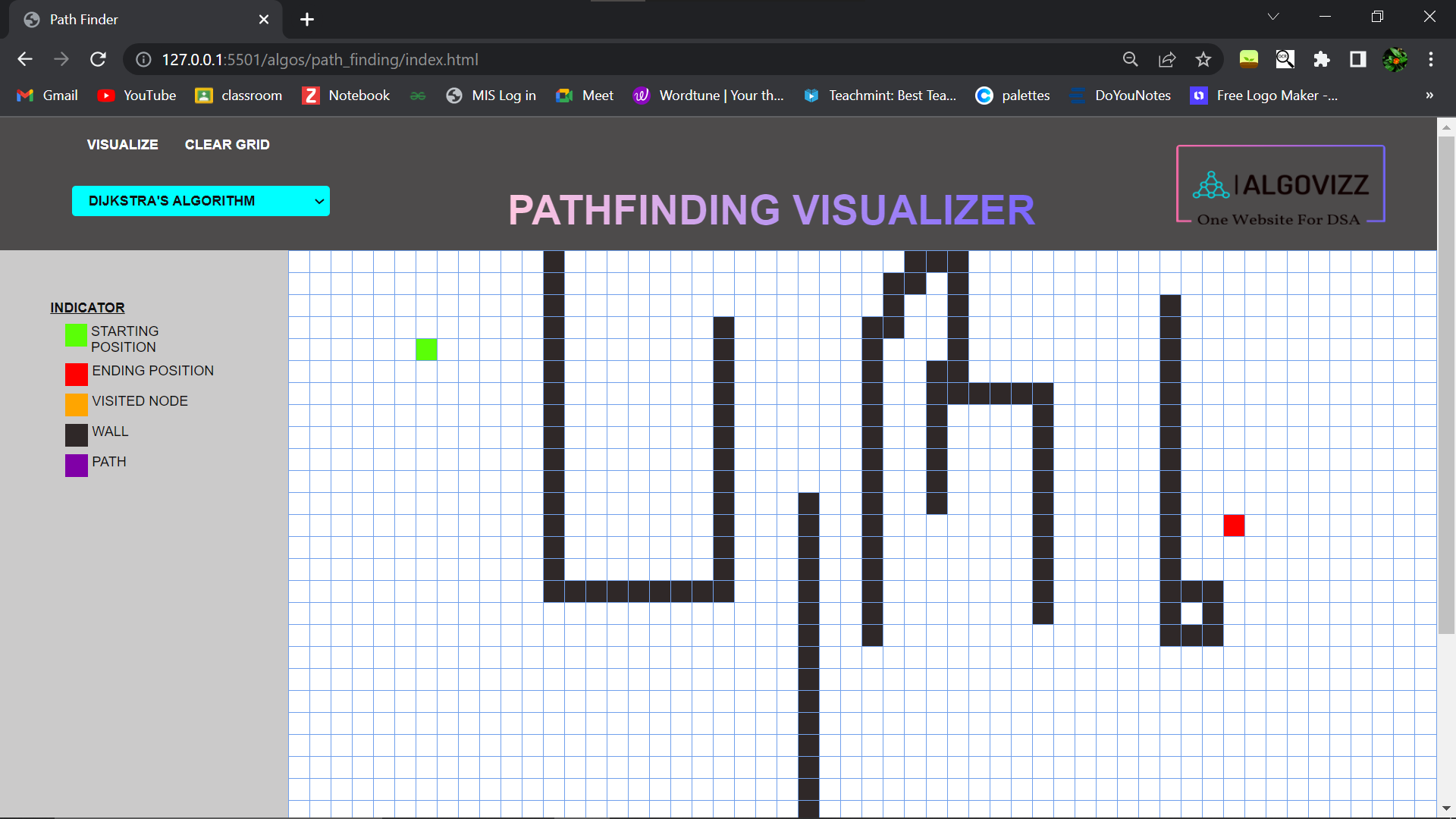


* **Path FInder:**

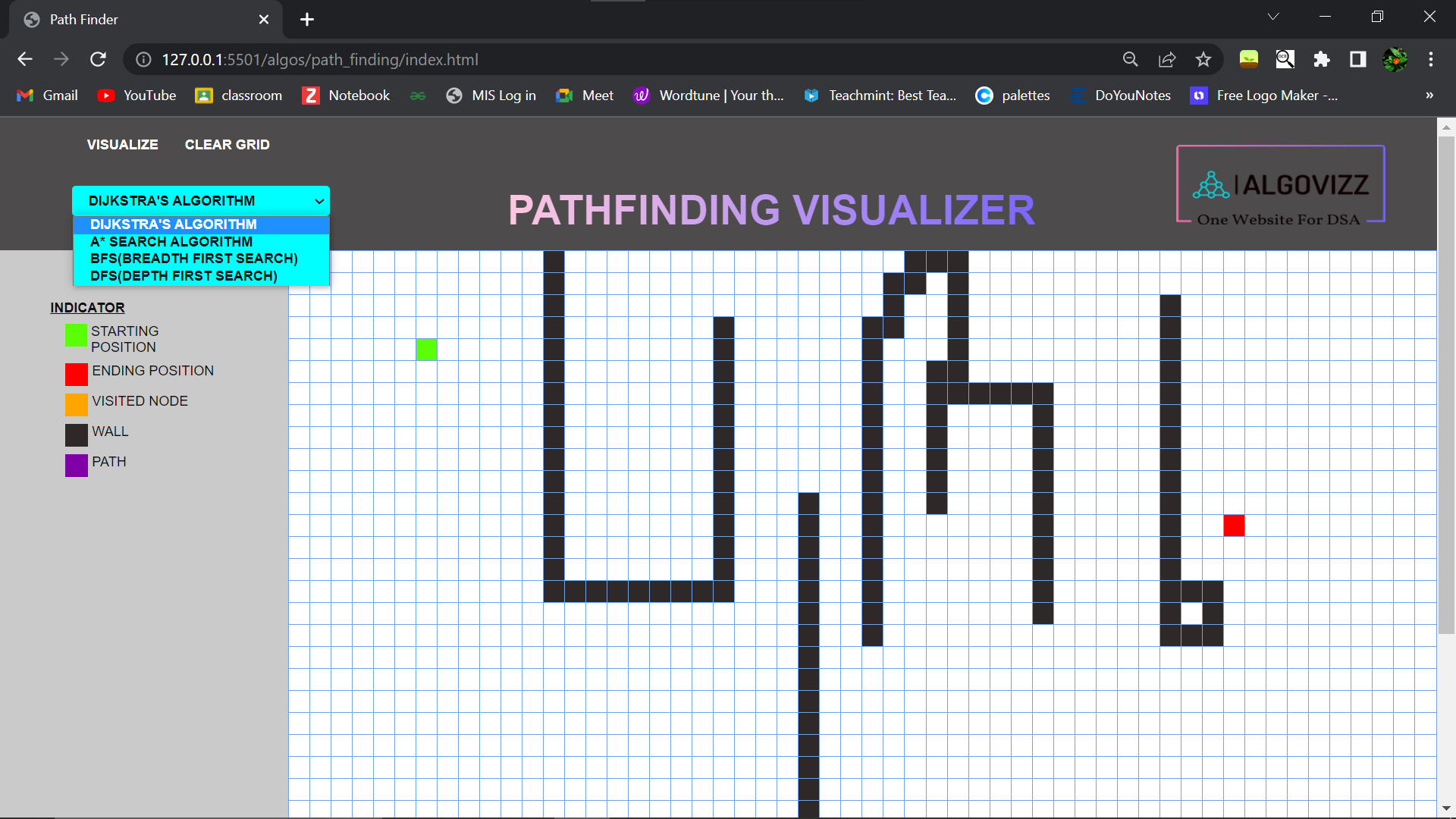
Plain Canvas:



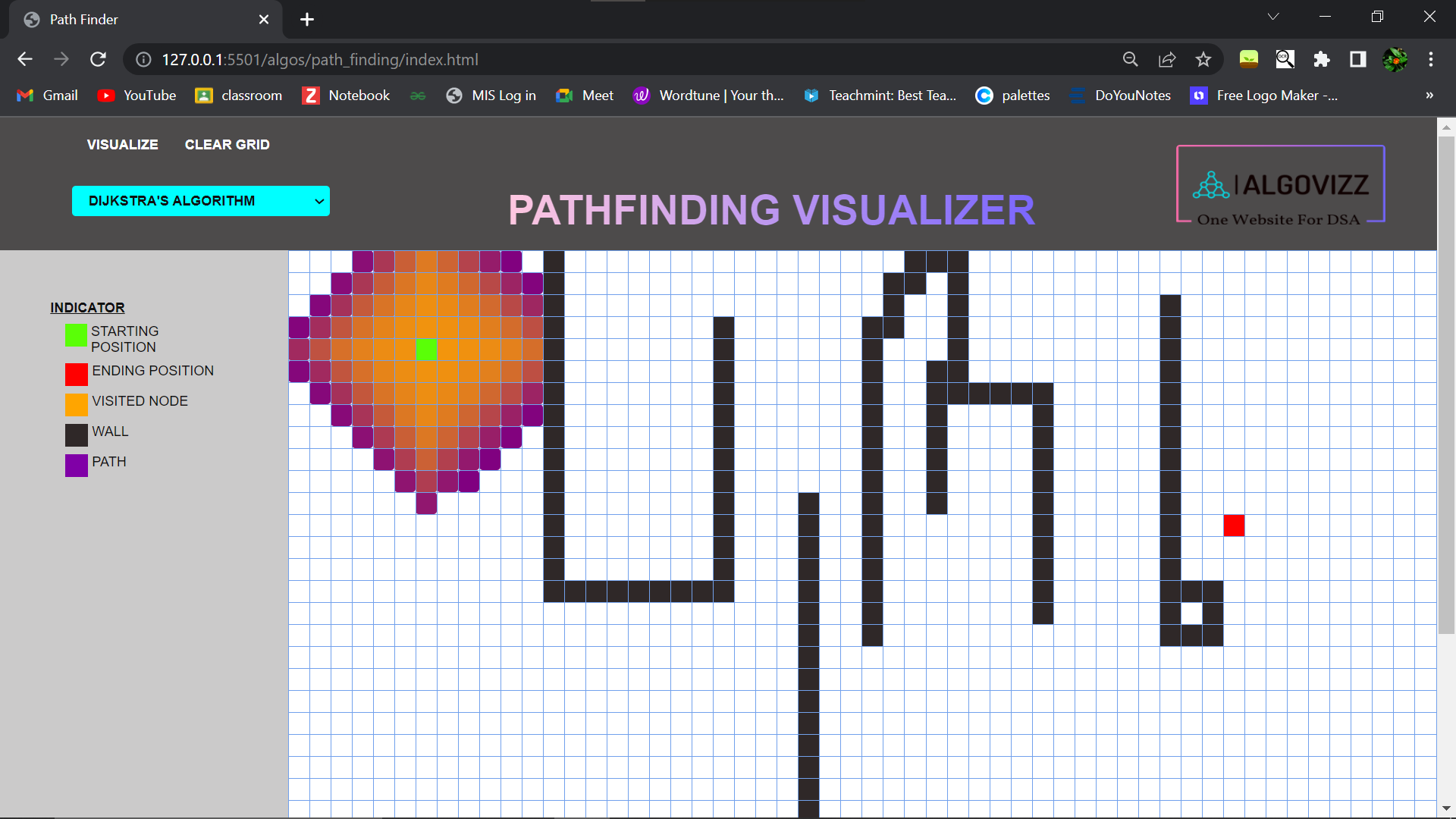
Changing the position and creating the grid:

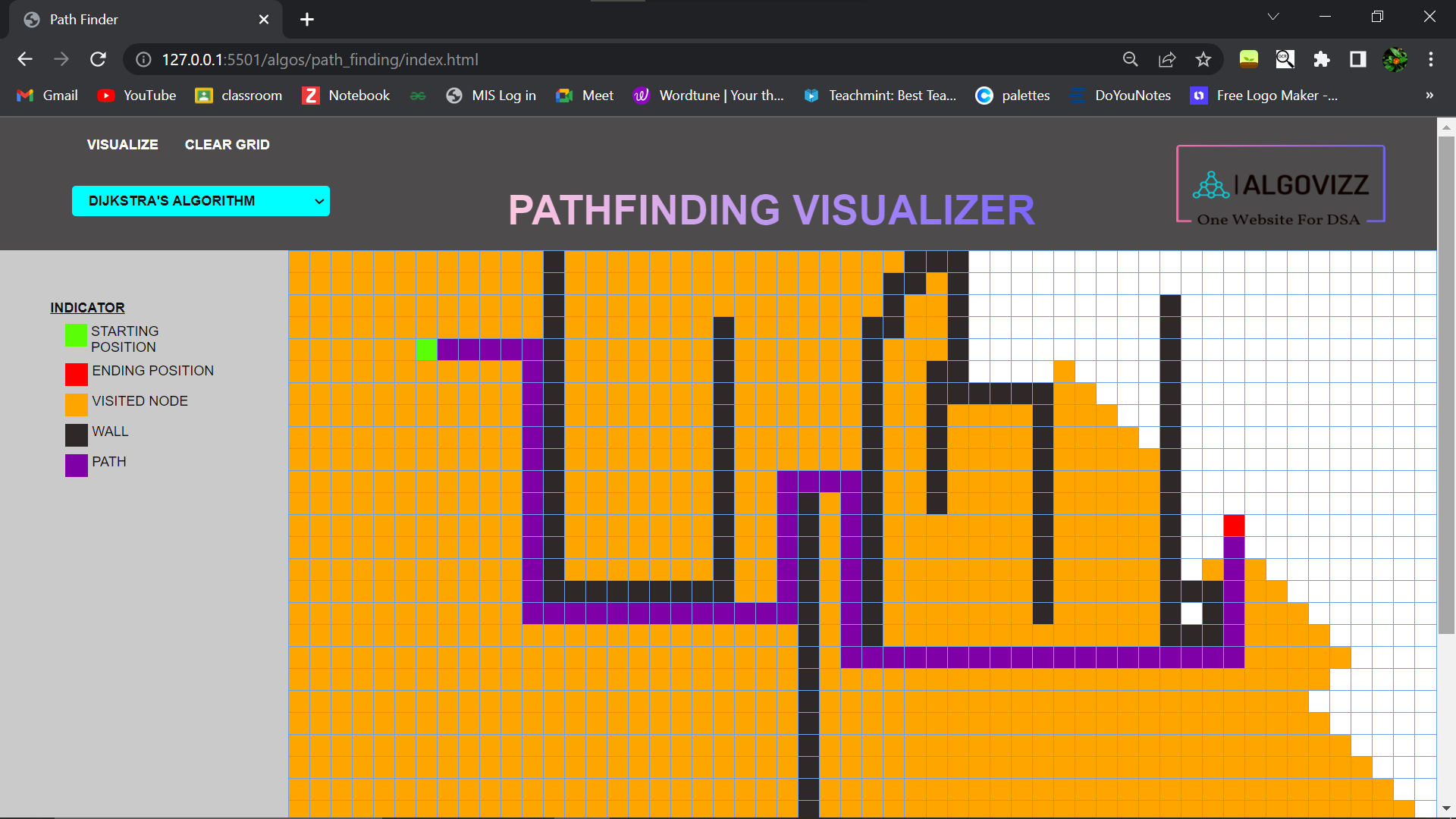


Algorithm List:

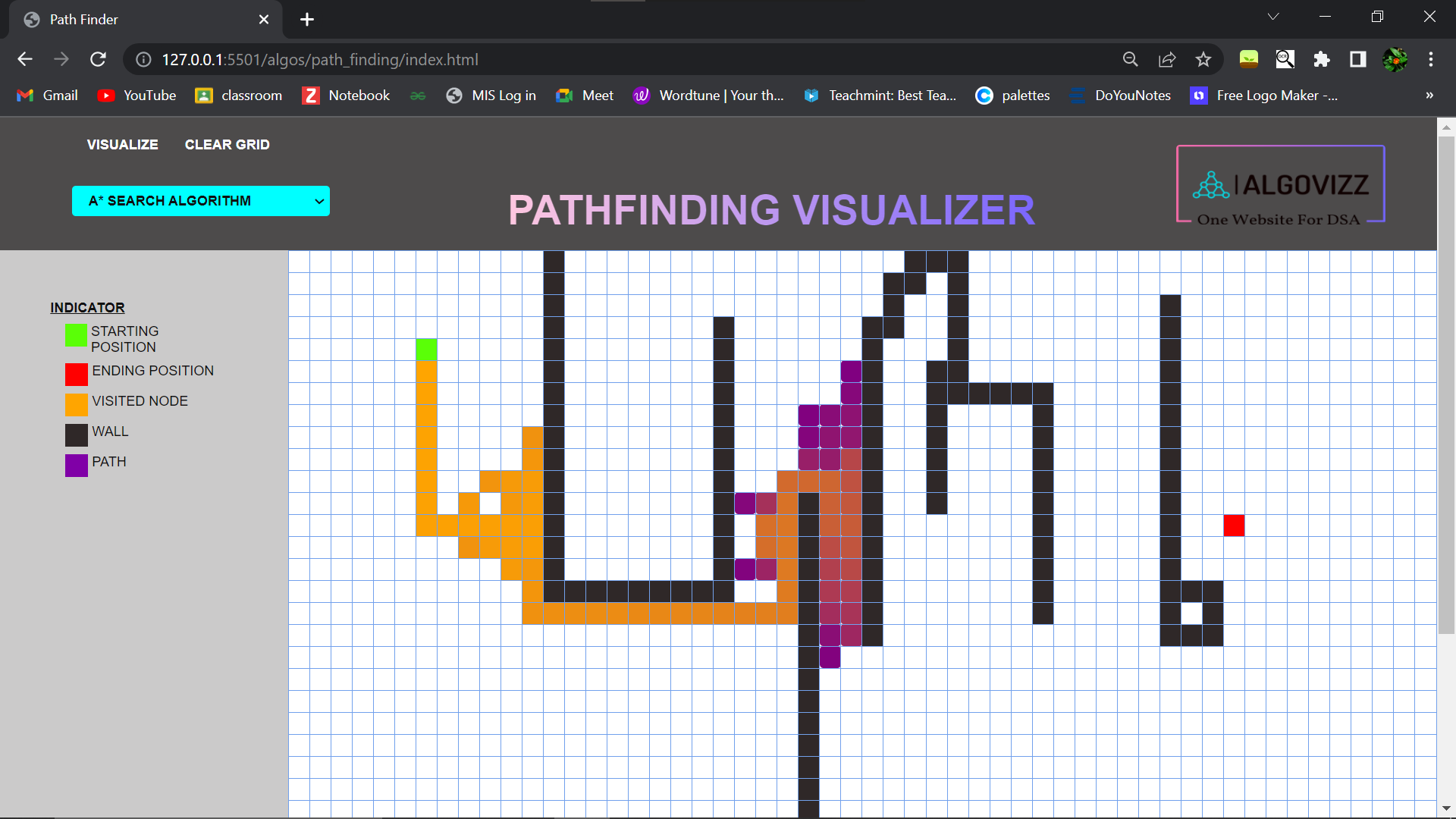


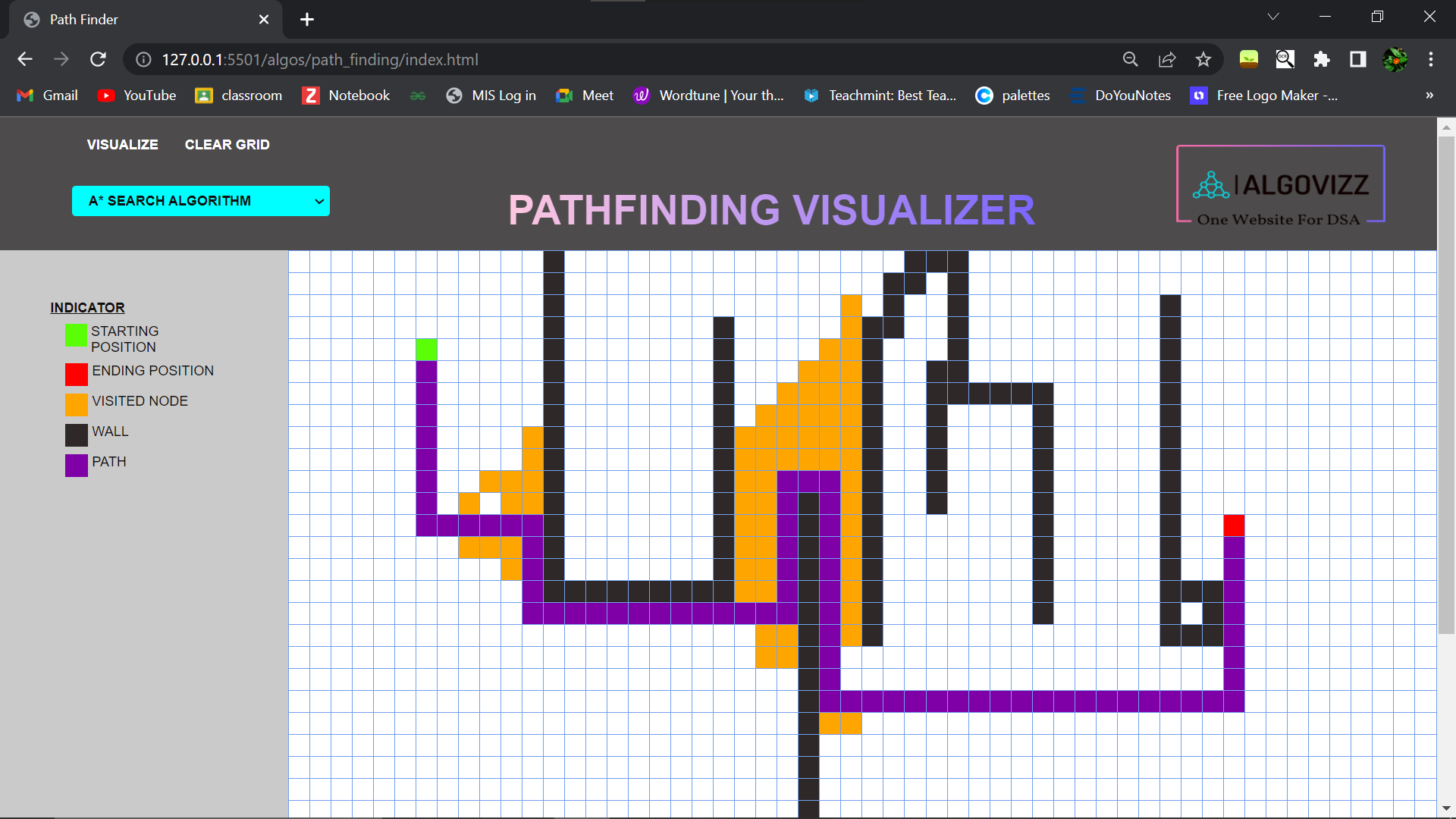
Dijstra’s Algirthm:



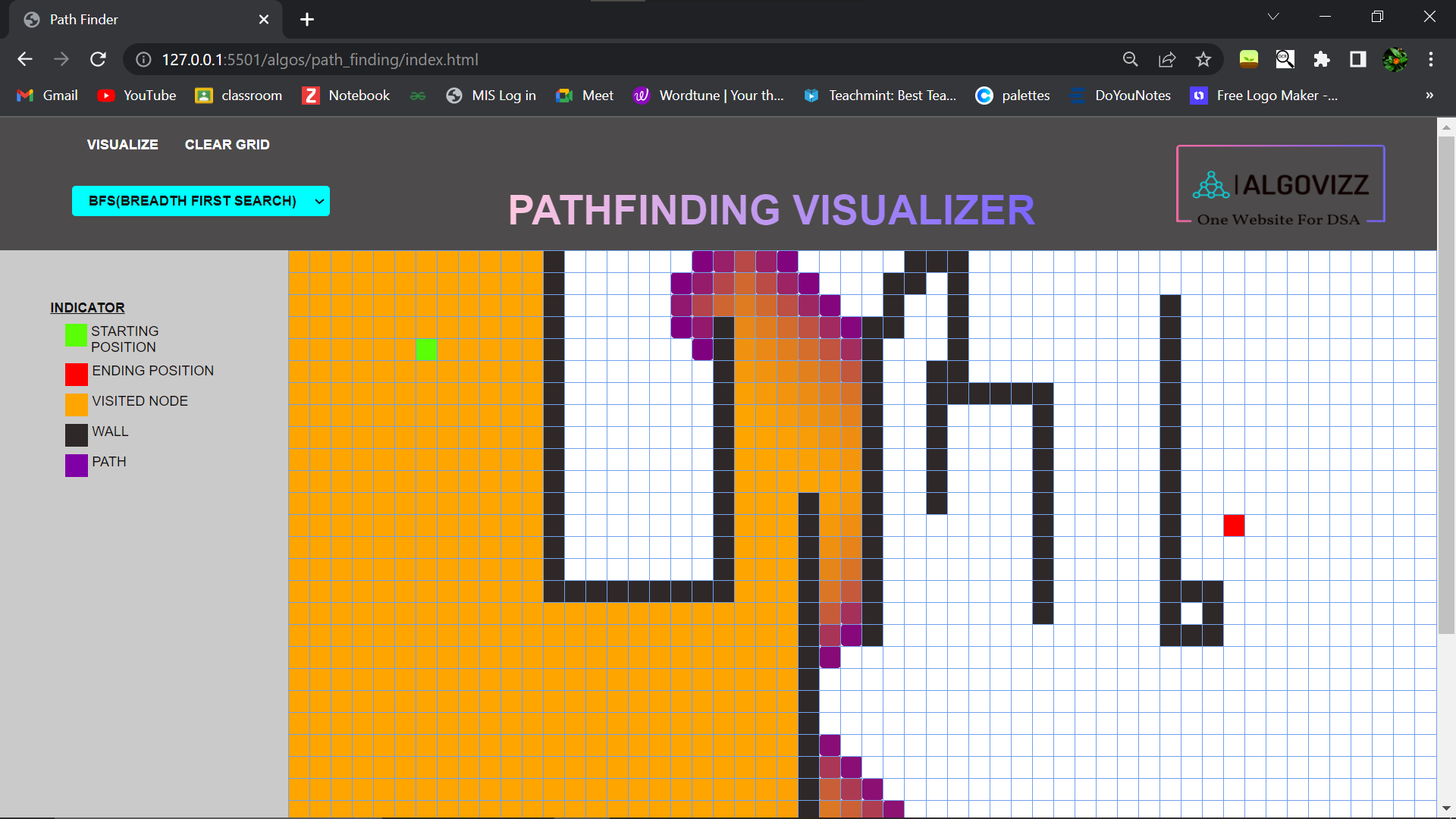


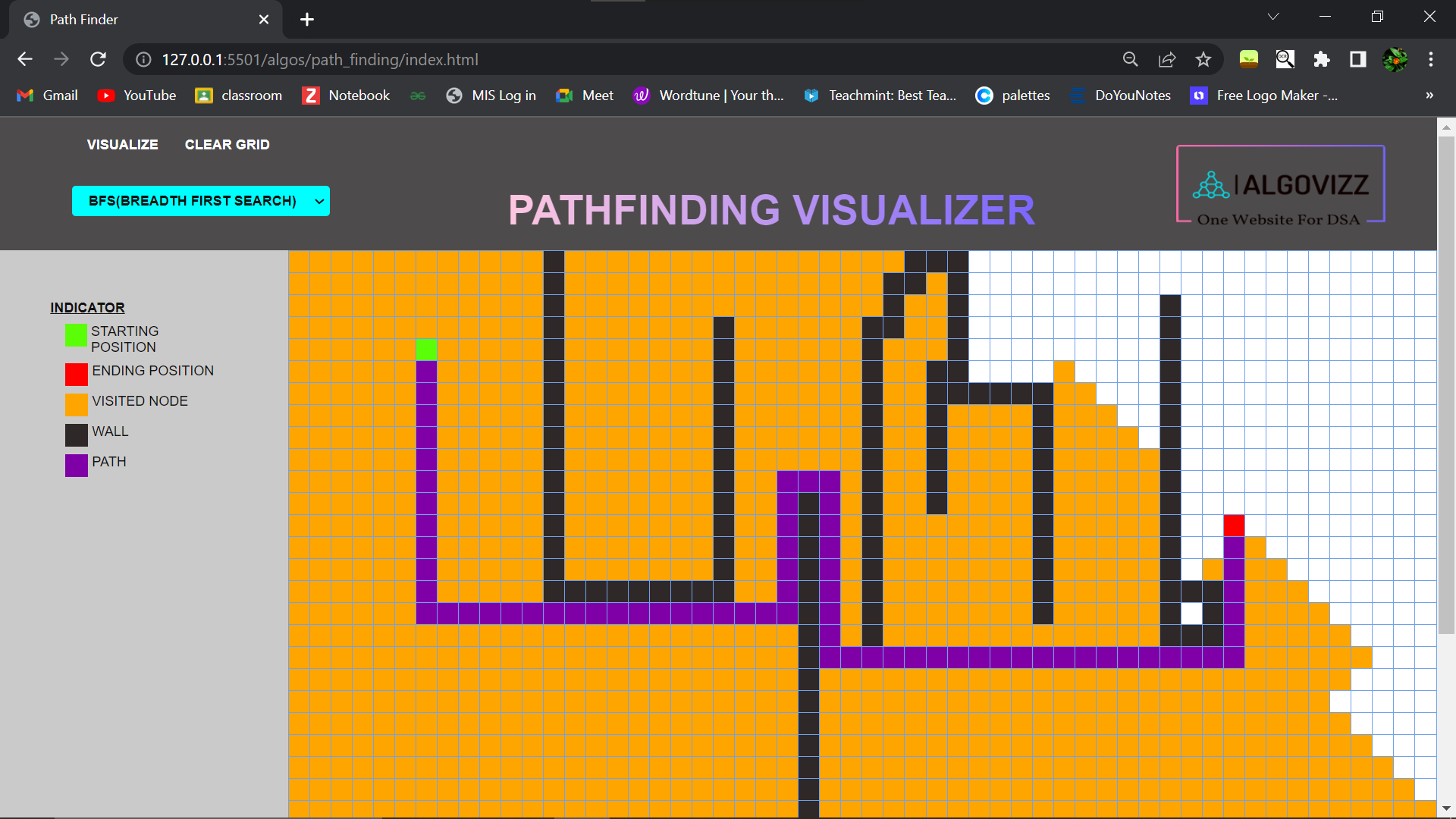
A\* Algirthm:



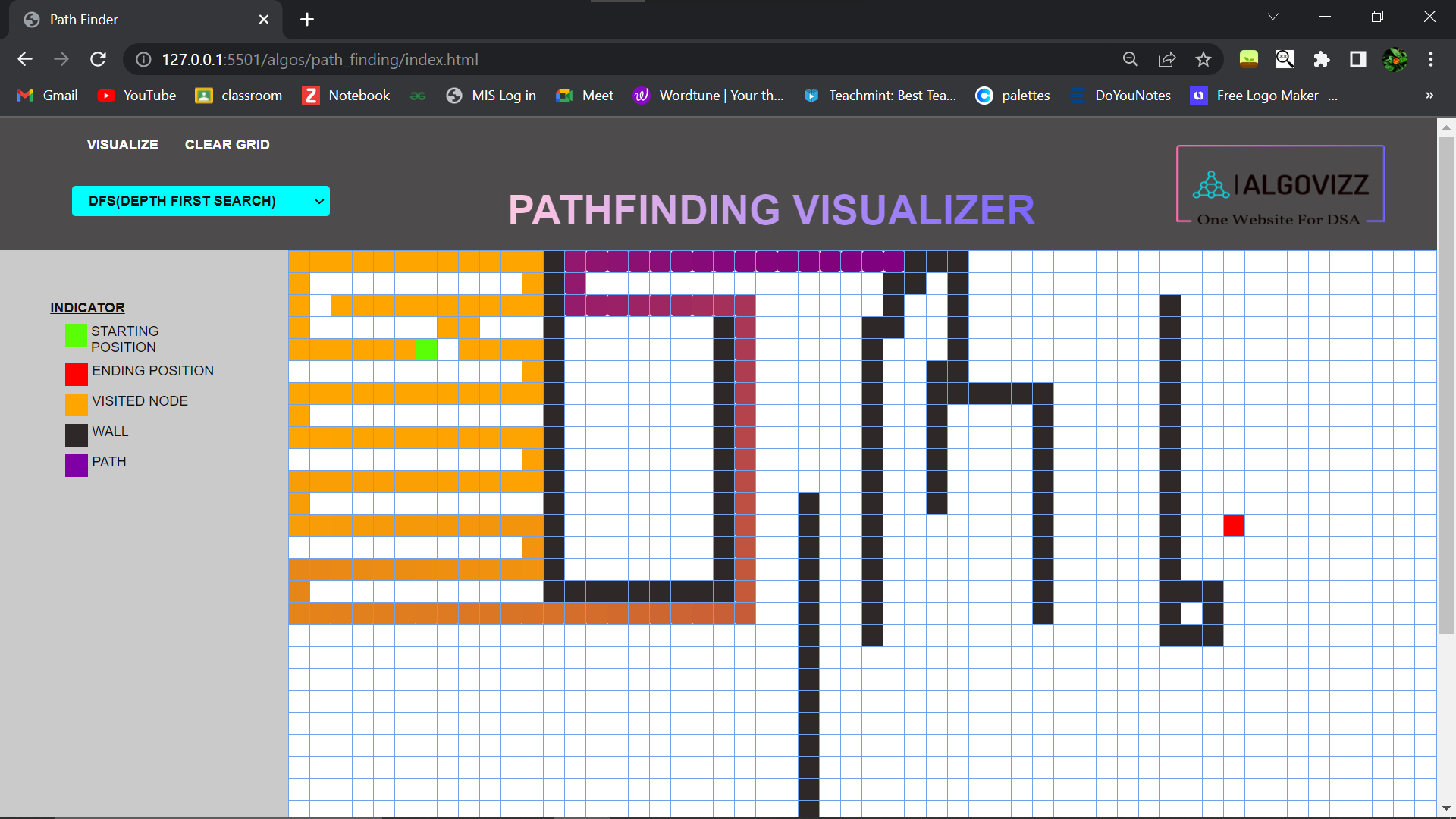


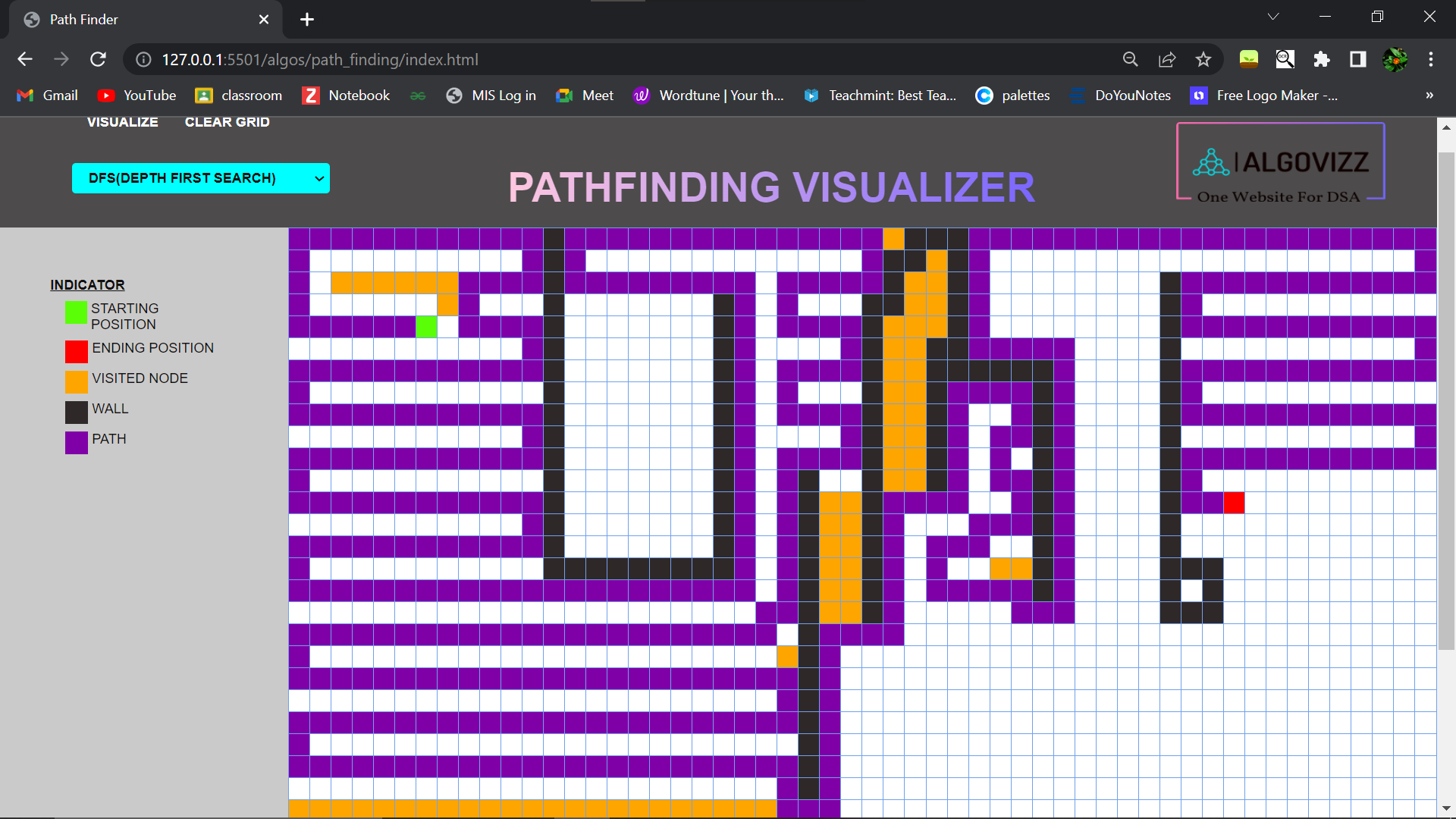
BFS Algirthm:



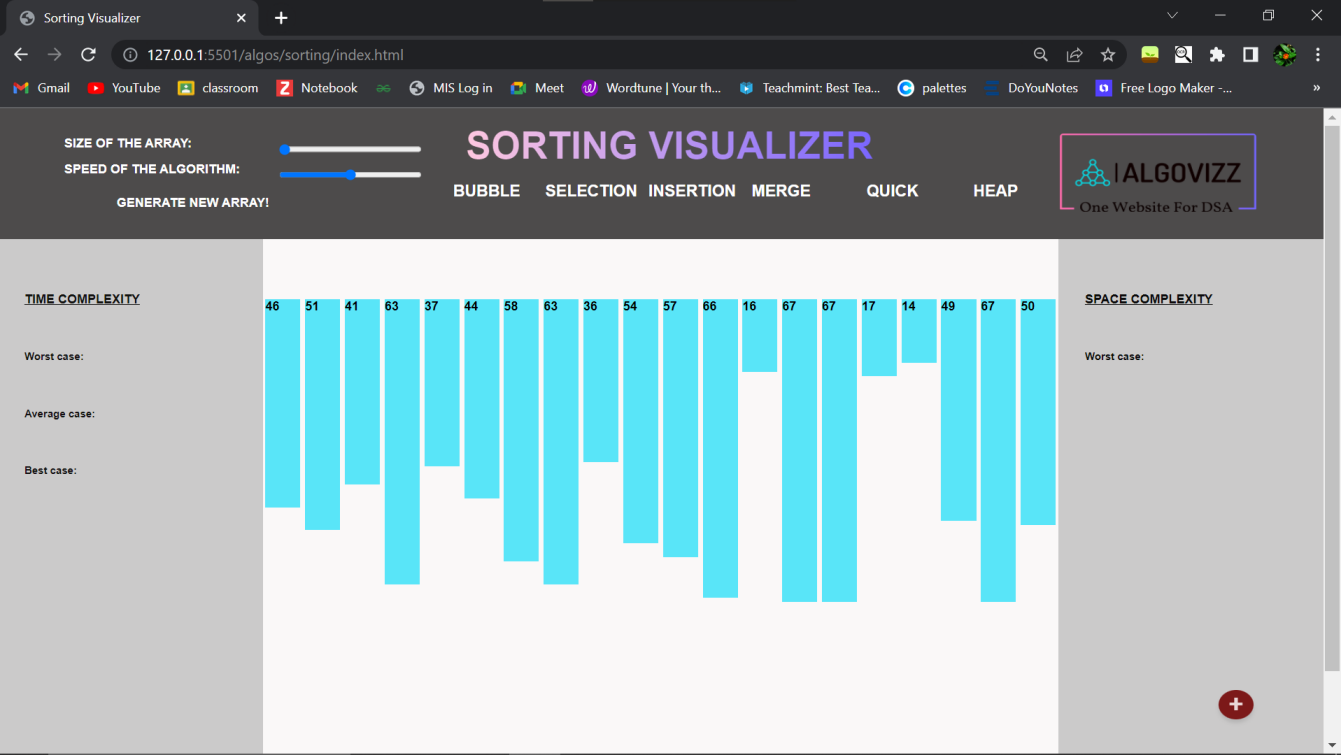


DFS Algorithm:

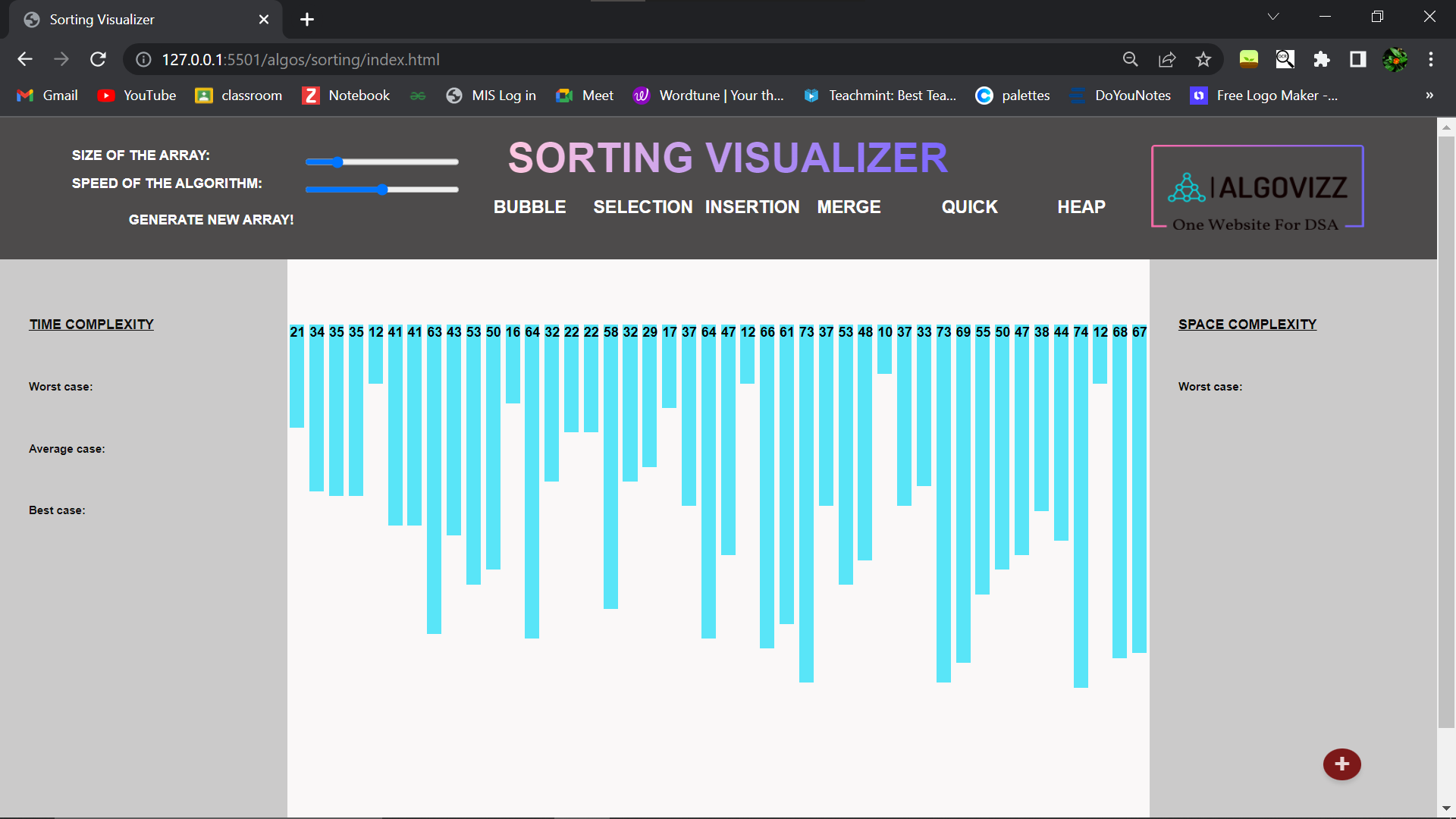




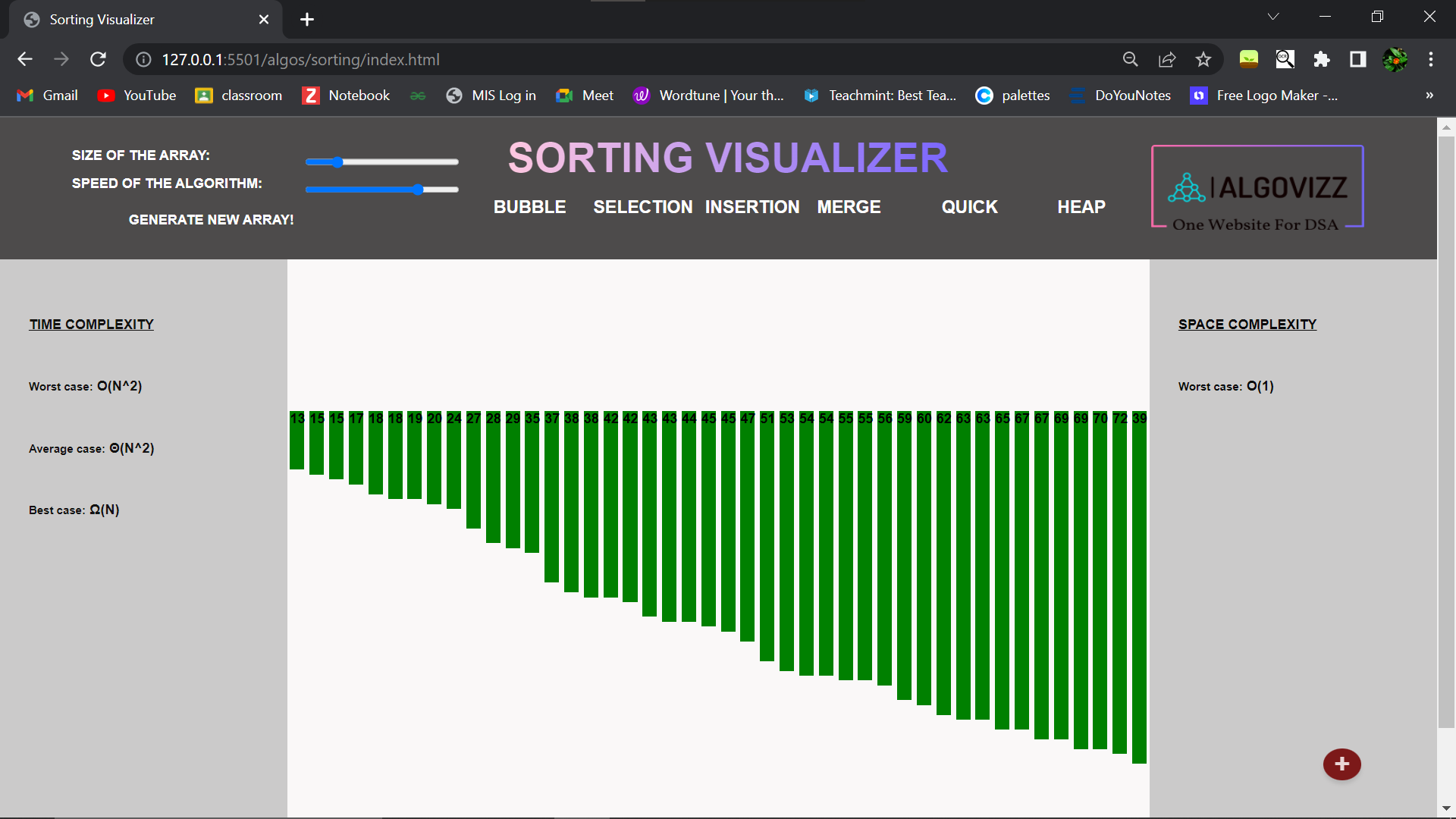
* **Sorting Algorithms:**



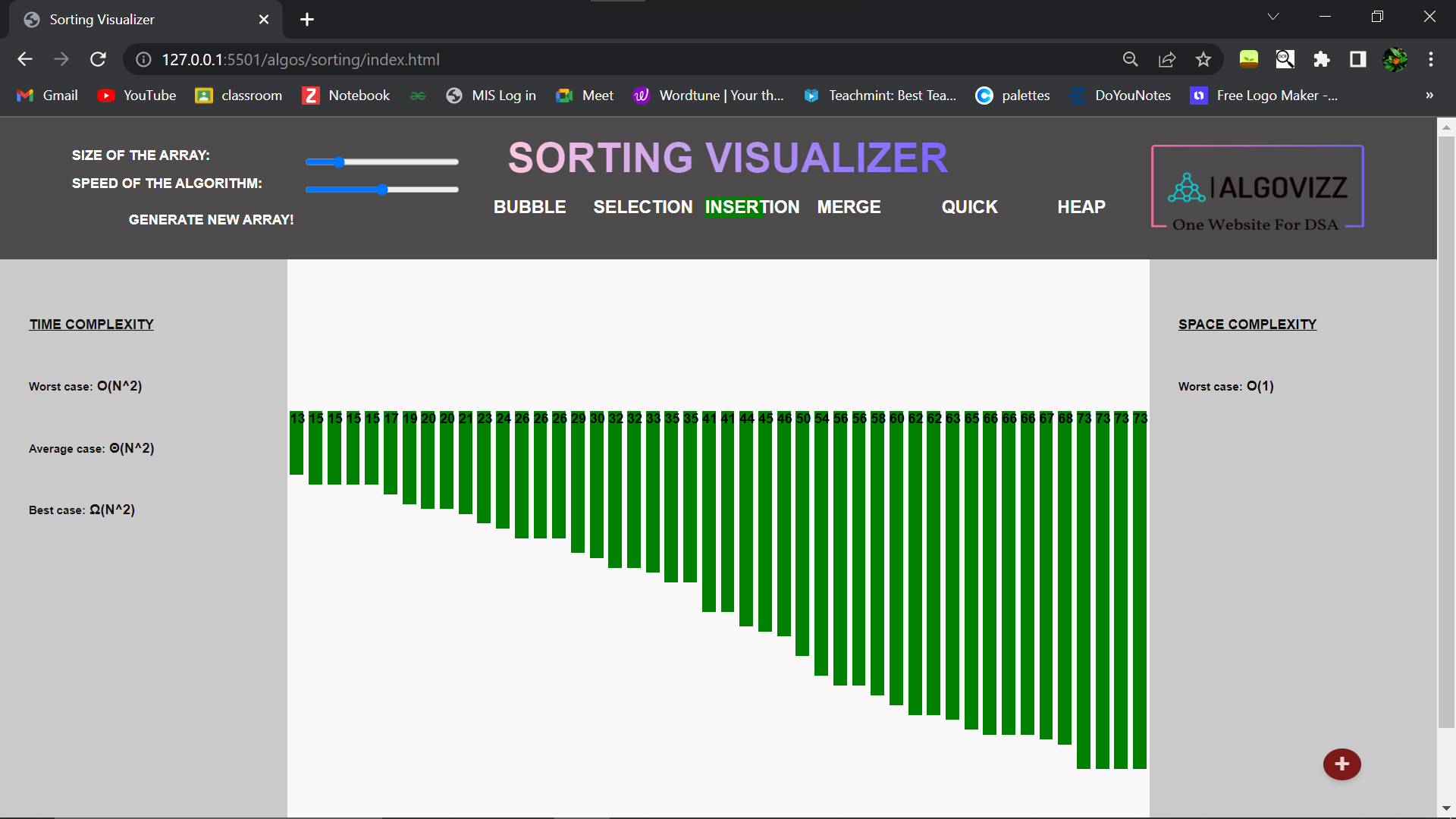
Change the size of array:



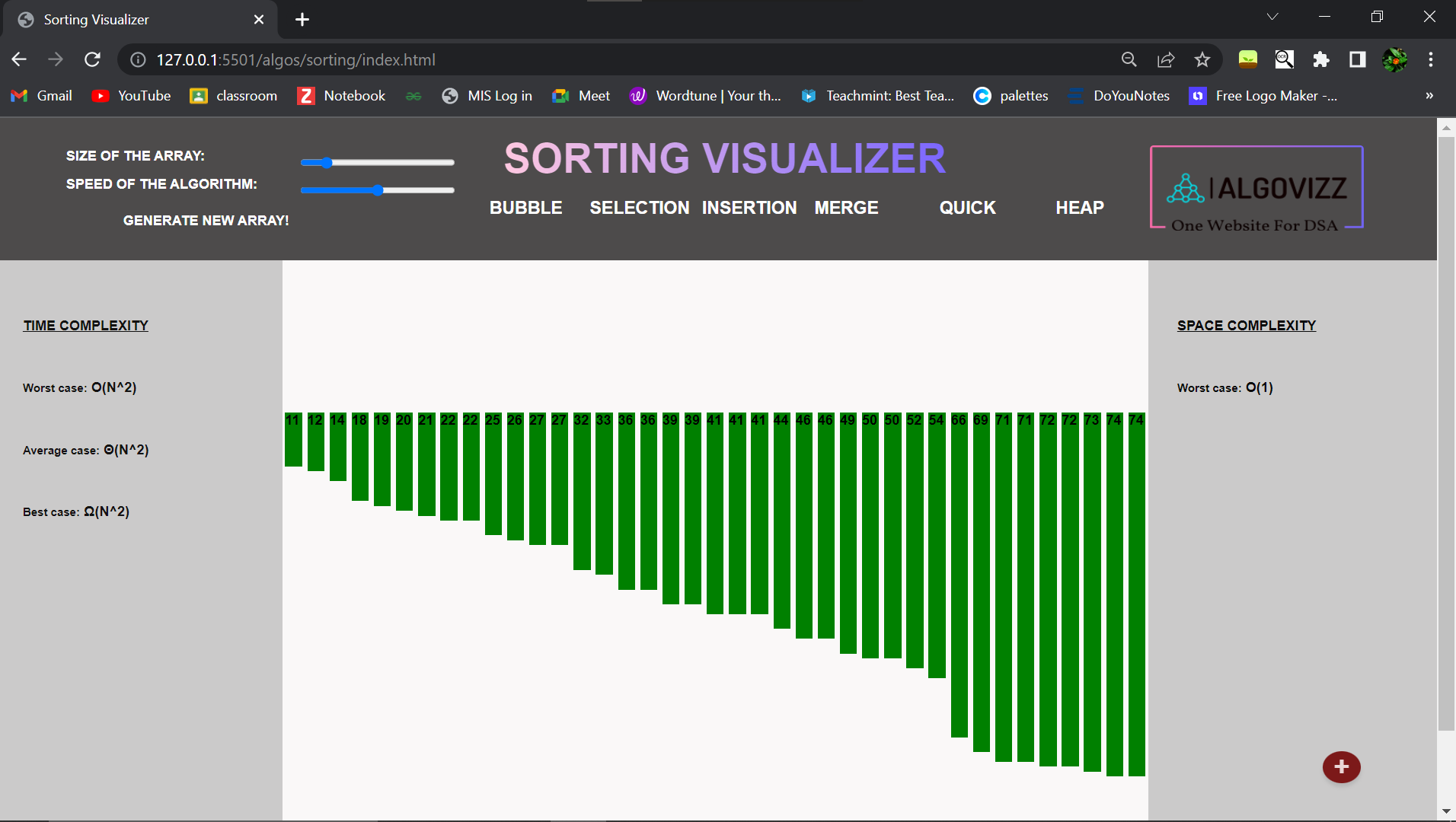
Bubble sort:



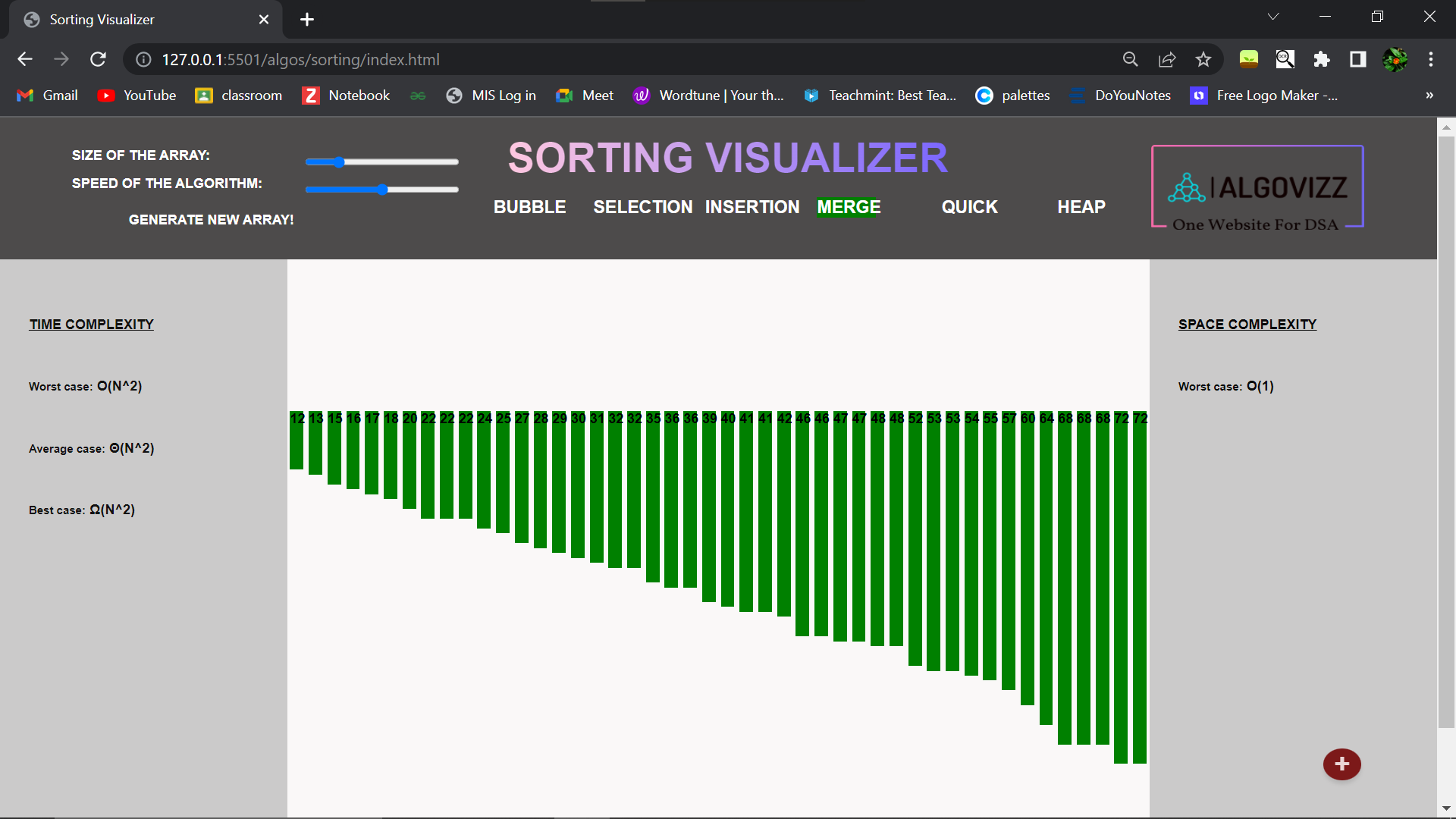
Insertion Sort:



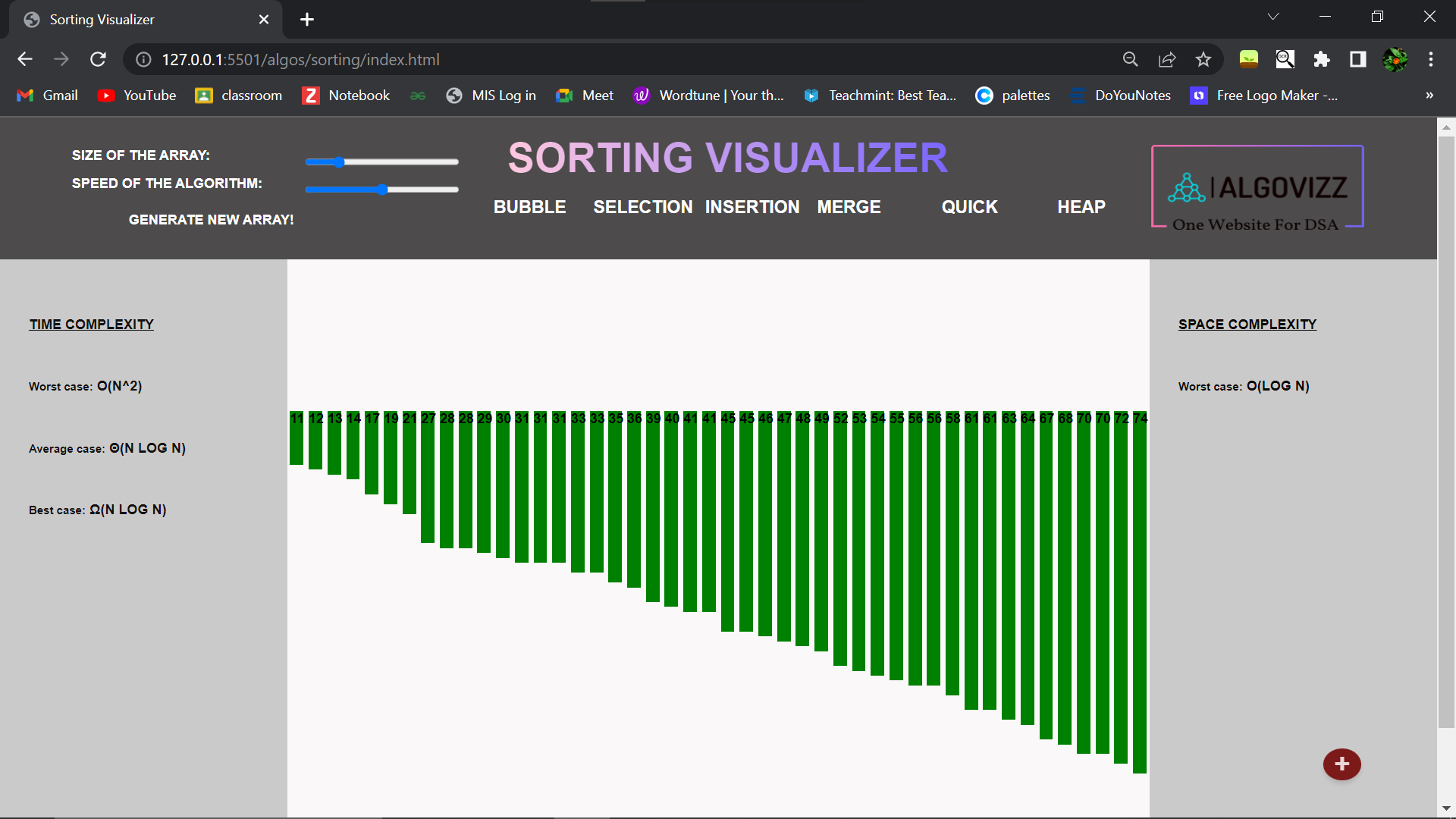
Selection sort



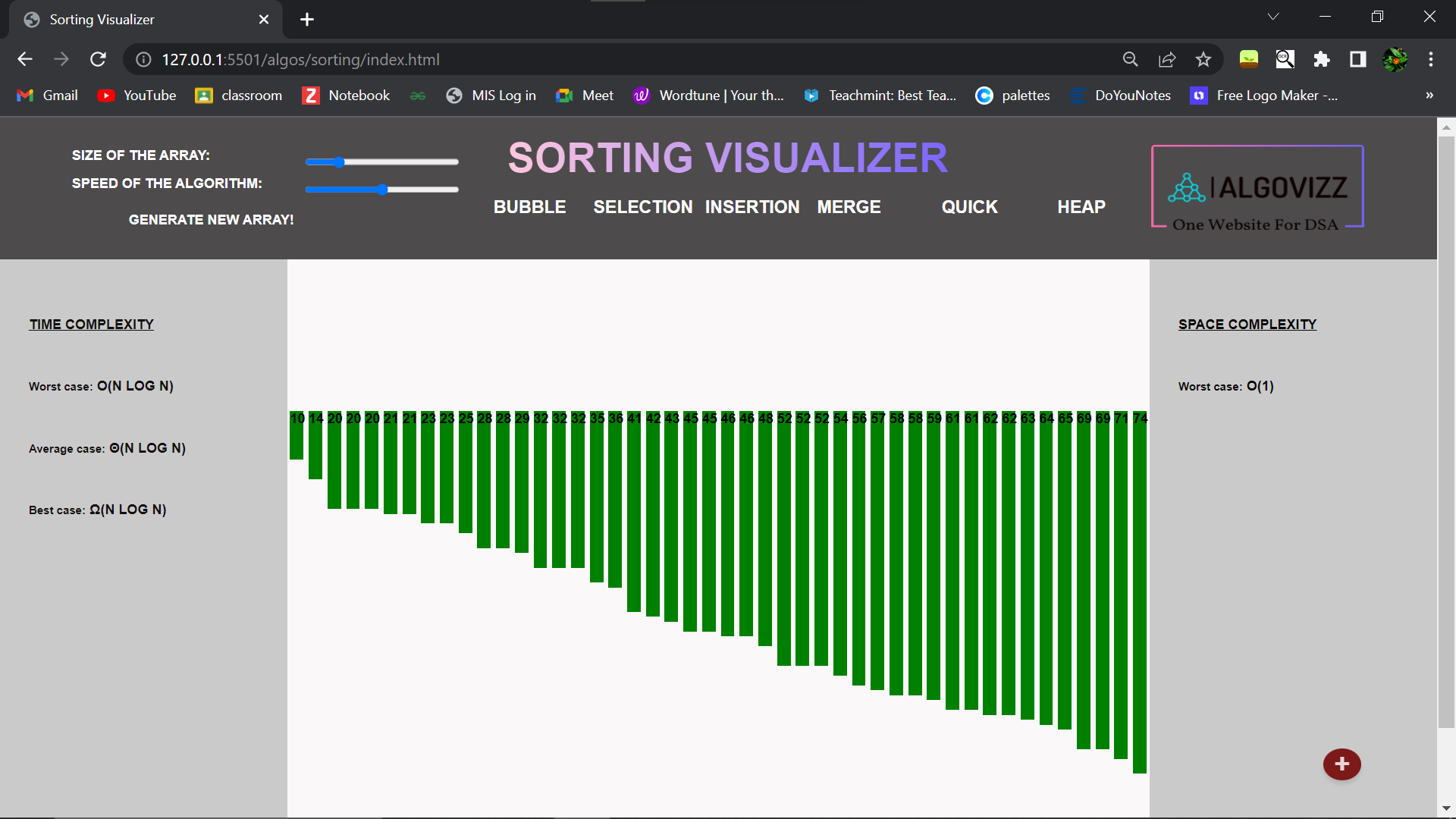
Merge Sort:



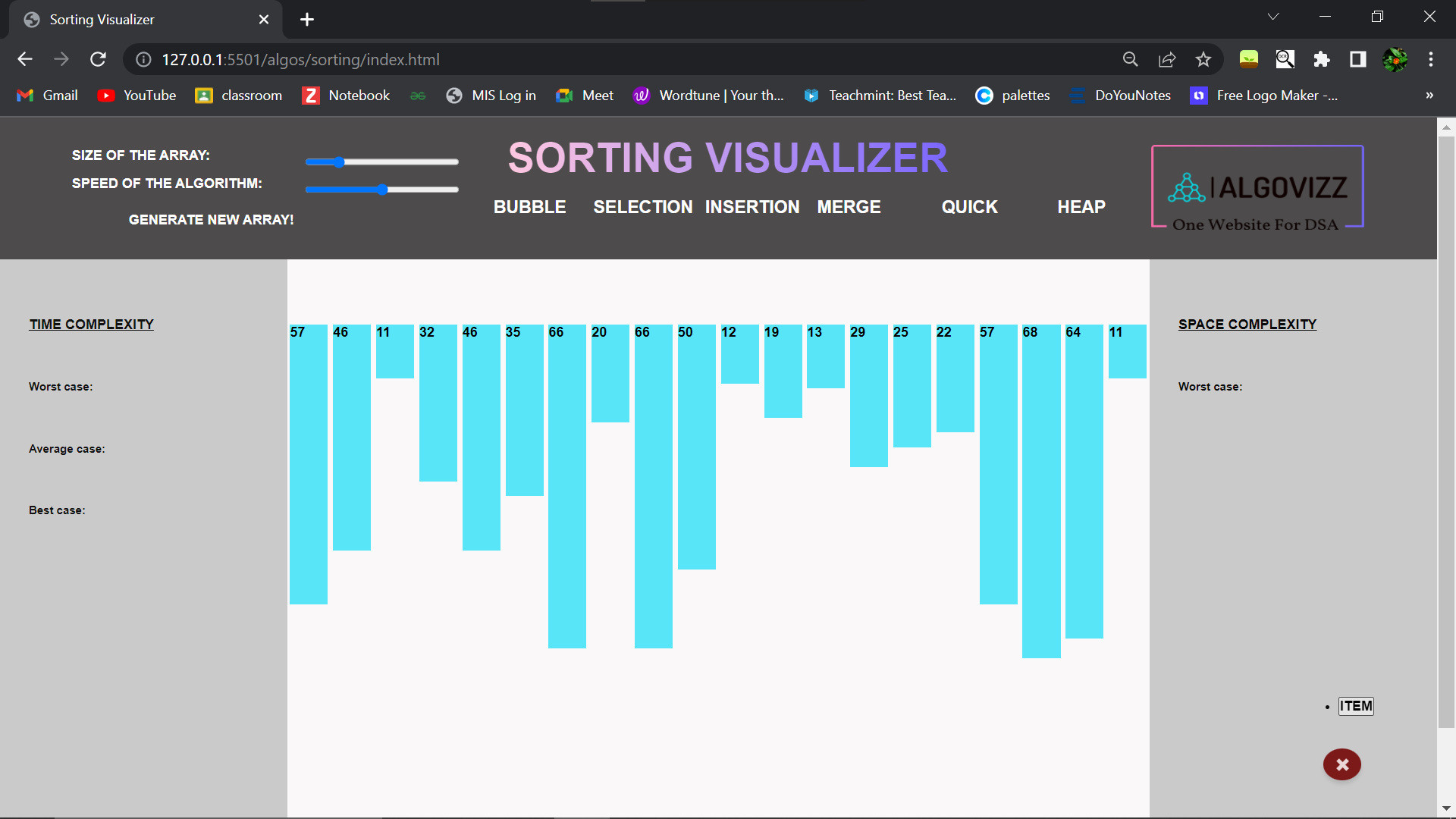
Quick sort

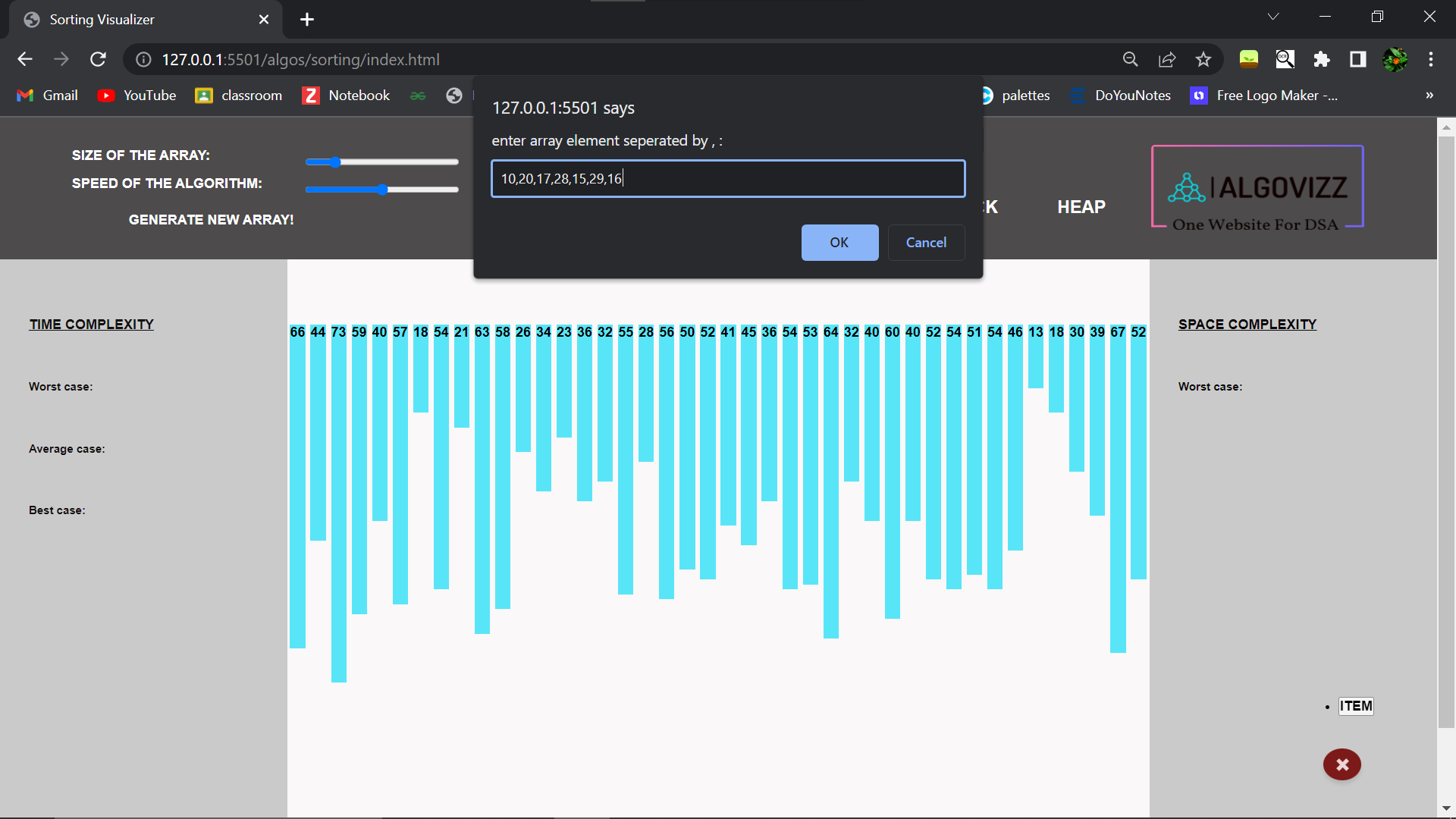


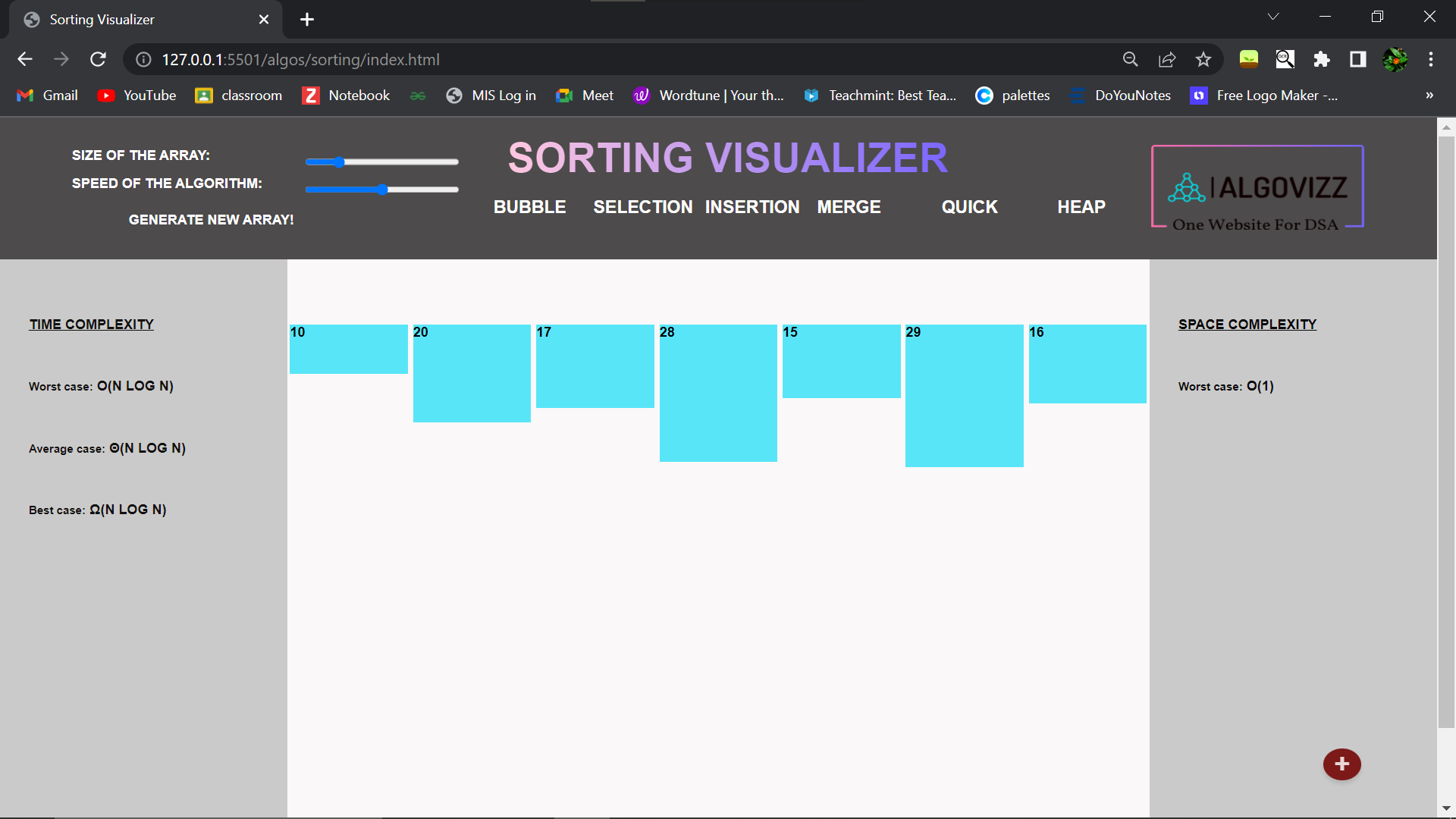
Heap Sort:



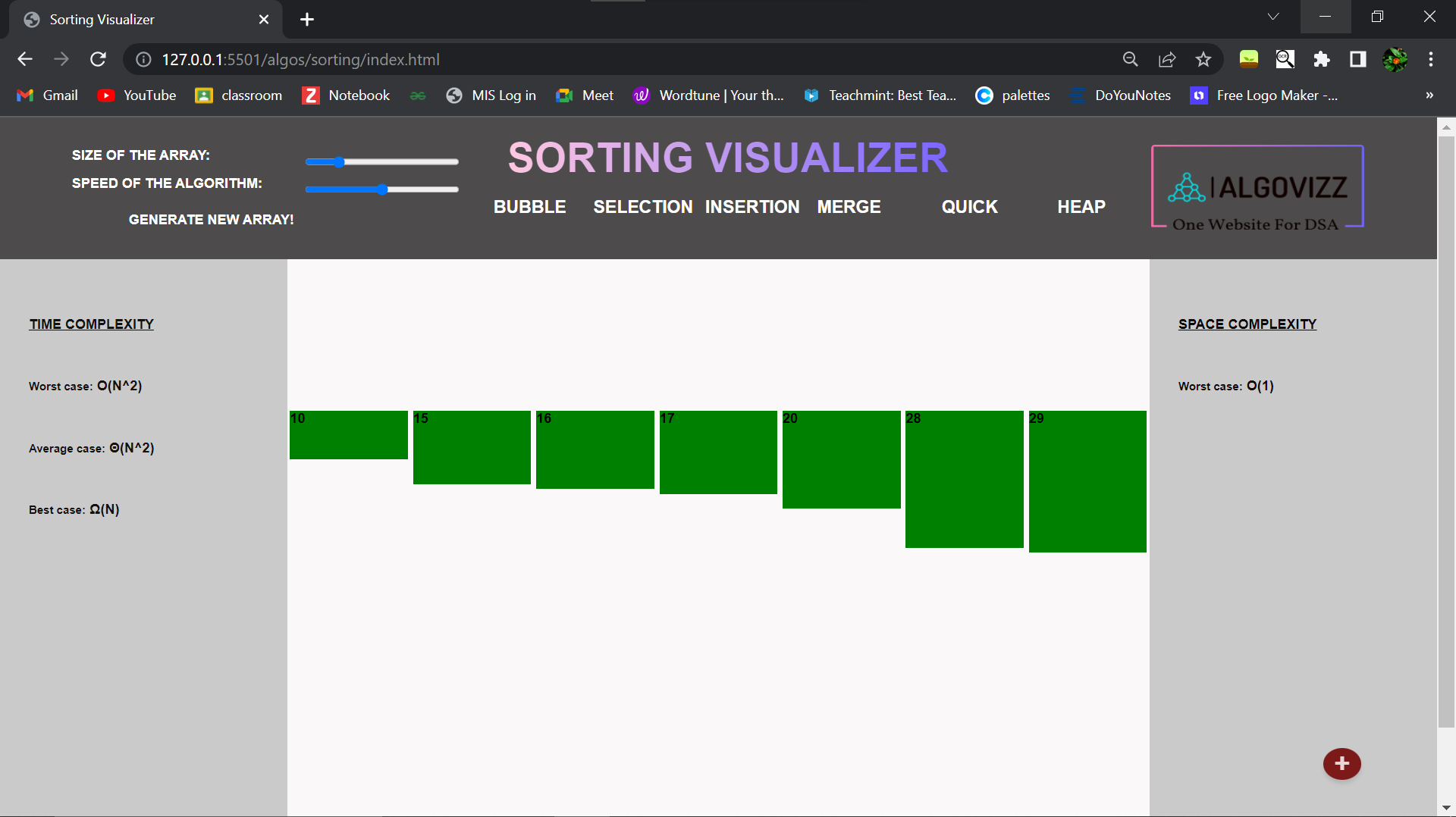
Genertate user-define array:







Sorted array:



**Chapter 7**

**Testing**

* 1. **What is Testing?**

The purpose of testing is to determine whether a system or its components meets the specified requirements. Software testing is an important part of the software development life cycle, identifying errors or defects early. Testing involves observing how the system or component performs under specific conditions, and comparing the actual results with what is expected.In testing, we ensure that the system or components meet the requirements and are error-free. Using it reduces the risk of bugs or failures in production. Stakeholders are also assured that the system or components are reliable.

* 1. **Type of Testing:**
* **What is White Box Testing?**

White box testing, also known as clear box testing, is a testing technique that involves examining the internal structure and design of the software being tested. It requires knowledge of the internal workings of the code and is typically done by developers or testers with programming experience. The purpose of white box testing is to ensure that the code is written correctly and functions as expected.

* **What is Black Box Testing?**

Black box testing, on the other hand, is a testing technique that involves testing the software without any knowledge of its internal workings. It is typically done by testers who do not have access to the source code and focuses on the external behavior of the software. The purpose of black box testing is to ensure that the software meets the specified requirements and functions as expected from the end-user's perspective.

Block box testing refers to a testing technique that involves testing a specific module or component of the software without any knowledge of its internal workings. It is often used in combination with both white box and black box testing to provide a comprehensive testing approach. The purpose of black-box testing is to ensure that individual modules or components of the software meet the specified requirements and function as expected.

* 1. **Test Cases:**
* **Test Cases for Home Page:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Testcase**  **\_ID** | **Test case  Objective** | **Step** | **Input** | **Expected  Result** | **Actual Result** | **Status** |
| 1 | TC\_1 | Verify if the all tab icon represents its corresponding tab. | view on  Home page | N/A | All icon should  display correctly | All icon is displaying  Correctly | Pass |
| 2 | TC\_2 | Verify that there is any grammatical or spelling mistake | view on  Home page | N/A | All spelling should be correct & no grammatical mistake is present | No mistake is present in this make | pass |
| 3 | TC\_3 | Check that the logo is visible and the content is readable | Logo  {at the left-top corner} | N/A | Logo must be visible that the content is readable | Logo is visible such that the content is readable | Pass |
| 4 | TC\_4 | Verify that the linking is working properly | Click on the link | Link | Each link must be heading towards the  Correspondig module | Each link is heading towards the corresponding module | Pass |

* **Test Cases for Sorting Visualizer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Testcase**  **\_ID** | **Test case  Objective** | **Step** | **Input** | **Expected  Result** | **Actual Result** | **Status** |
| 1 | TC\_1 | The array must be visible after the module opens | view on  array container | N/A | The downward graph represents an array | The downward graph is representing the array | Pass |
| 2 | TC\_2 | Verify the number of sorting algorithms | View the menu bar | N/A | Required sort algorithms: Bubble, Insertion, Selection, Merge, Heap, and Quick sort | Actual sort algorithms: Bubble, Insertion, Selection, Merge, Heap, and Quick sort | Pass |
| 3 | TC\_3 | Verify level of speed for visualization | Adjusting the “Speed of algorithm ” | N/A | There must be 5 levels of speed | There are 5 levels of speed | Pass |
| 4 | TC\_4 | Verify the values associated with the bar | View the value at the top of each bar | N/A | should be the height of that bar | Displaying the height of the bar | Pass |
| 5 | TC\_5 | Verify the “Generate New Array” button | Click on the Button | Button | Every time a new array should generated | Every time a new array is generated | Pass |
| 6 | TC\_6 | Verify the difference between the speed of algorithm | 1. Sort the array on high speed 2. Sort the array on low speed | Array,sort | The difference should be visible | The difference is visible | Pass |
| 7 | TC\_7 | Verify the Bubble sort | 1. generate new array 2. Click the button of bubble sort | Array,sort | The bar should be arranged in sorted order corresponding with its value | The bars are arranged in a sorted order corresponding with its value | Pass |
| 8 | TC\_8 | Verify the Insertion sort | 1generate new array  2Click the button of Insertion sort | Array,sort | The bar should be arranged in sorted order corresponding with its value | The bars are arranged in a sorted order corresponding with their value | Pass |
| 9 | TC\_9 | Verify the Selection sort | 1.generate new array  2.Click the button of Selection sort | Array,sort | The bar should be arranged in sorted order corresponding with its value | The bars are arranged in a sorted order corresponding with their value | Pass |
| 10 | TC\_10 | Verify the Merge sort | 1. generate new array   2Click the button of Merge sort | Array,sort | The bar should be arranged in sorted order corresponding with its value | The bars are arranged in a sorted order corresponding with their value | Pass |
| 11 | TC\_11 | Verify the Quick sort | 1.generate new array  2.Click the button of Quick sort | Array,sort | The bar should be arranged in sorted order corresponding with value | The bars are arranged in a sorted order corresponding with their value | Pass |
| 12 | TC\_12 | Verify the Heap sort | 1. generate new array 2. Click the button of Heap sort | Array,sort | The bar should be arranged in sorted order corresponding with its value | The bars are arranged in a sorted order corresponding with their value | Pass |
| 13 | TC\_13 | Verify the complexity of algorithm | 1. Select the algorithm 2. View complexity | array | The complexity should be calculated for each array dynamically | The complexity is displayed for each algorithm statically | Fail |
| 14 | TC\_14 | Veritfy the user-defined array functionality | 1. Click on the “+” button at the bottom right corner 2. Add the value to the text bar separated by comma 3. Click on “OK” 4. View the array container | Array item | A downward bar graph corresponding with the inputted value should be created | A downward bar graph corresponding with the inputted value is created | Pass |
| 15 | TC\_15 | Verify the link to jump back to the home page | Click on the logo (at the top right corner) | N/A | Should Navigate Home page | Is Navigating to the Home page | Pass |

* **Test Cases for Path Finder:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Testcase**  **\_ID** | **Test case  Objective** | **Step** | **Input** | **Expected  Result** | **Actual Result** | **Status** |
| 1 | TC\_1 | Verify the start and end point is movable or not | Drag the start and end points | N/A | Should be movable | Is movable | Pass |
| 2 | TC\_2 | Verify the Algorithms | Click on the drop-down arrow at the bottom-left corner of the menu bar | N/A | Required algorithms: Dijkstra’a, A\*, BFS, DFS algorithm | Actual algorithms: Dijkstra’a, A\*, BFS, DFS algorithm | Pass |
| 3 | TC\_3 | Verify the Visualize button | Click on the visualize button | button | Start the visualization according to the selected algorithm | Starting the visualization according to the selected algorithm | Pass |
| 4 | TC\_4 | Verify the create Grid functionality | Click and drag the mouse where you want to create obstacles | N/A | Obstacles must not block endpoints. | Obstacles do not block points. | Pass |
| 5 | TC\_5 | Verify the clear grid button | Click on the clear grid button | Button | Must make the grid clear like it was before performing any activity | Make the grid clear | Pass |
| 6 | TC\_6 | Verify the link to jump back to the home page | Click on the logo (at the top right corner) | N/A | Should Navigate Home page | Is Navigating to the Home page | Pass |

* **Test Cases for Binary Search Tree:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Testcase**  **\_ID** | **Test case  Objective** | **Step** | **Input** | **Expected  Result** | **Actual Result** | **Status** |
| 1 | TC\_1 | Verify the insert functionality | 1. Enter the desire value in the insert element text field 2. Click on the insert button | Node value | If the add to tree at the position according to its value | Adding to the tree | Pass |
| 2 | TC\_2 | Verify the delete functionality | 1. Enter the desire value in the delete element text field 2. Click on the delete button | Node value | Remove the node and replace it with the next node | Removing the node and replacing with the next node | Pass |
| 3 | TC\_3 | Verify if user enters the node which is already present in the tree to insert | 1. Enter the existing value 2. Click insert | Existing value | Error “Value already exists in tree” | Error “Value already exists in tree” | Pass |
| 4 | TC\_4 | Verify if the user deletes the node which is not present in the tree | 1. Enter the non-existing value 2. Click delete | Non-existing value | First track and then Error “Value is not present in tree” | Error “Value is not present in tree” | Pass |
| 5 | TC\_5 | Verify which node is replacing node after deleting | Delete the node which had two children nodes | Node value | The node with the least value on the right hand side (greater)  Should replace the deleting node | The node with the least value on the right hand side (greater) replacing the deleting node | Pass |
| 6 | TC\_6 | Verify the link to jump back to the home page | Click on the logo (at the top right corner) | N/A | Should Navigate Home page | Is Navigating to the Home page | Pass |

**Chapter 8**

**Advantages and Disadvantages**

**8.1 Advantages:**

1. **Improved Understanding:**

Algorithm visualizer helps to understand the algorithmic process by providing visual representation of the execution process. This makes it easier to understand how the algorithm works and identify any potential issues. Algorithm visualizer makes it easier to compare and contrast two different algorithms by providing a visual representation of their execution process side-by-side.

1. **Increased Engagement:**

Algorithm visualizer can be an engaging way to learn or teach algorithms, as it allows users to interact with the algorithm and observe its execution process in real-time.

1. **Better Debugging:**

Algorithm visualizer can be a powerful debugging tool, as it helps to identify any issues or errors in the algorithm and provides a visual representation of the execution process. This can make it easier to understand and resolve issues.

1. **Time-Saving**:

Algorithm visualizer can help save time in understanding and debugging algorithms, as it provides a quick and easy way to visualize the algorithm execution process. This can lead to faster development and debugging times.

1. **Experimentation**:

Algorithm visualizer allows users to experiment with different Example and observe the results in real-time. This can help to identify the most efficient algorithm for a particular problem or task.

**8.2 Disadvantages:**

1. **Limited Scope:**

Algorithm visualizer is a tool that is limited to the visualization of algorithmic execution. It does not take into account the broader context of the program, such as system interactions or user input.

1. **Simplification:**

Algorithm visualizer often simplifies the code by removing certain details or variables, which can result in a less accurate representation of the algorithm.

1. **Limited Customization:**

Algorithm visualizer often has limited customization options, which can limit its effectiveness for specific use cases or preferences.

1. **Resource Intensive:**

Algorithm visualizer requires significant computing resources to provide real-time visualization of the algorithmic execution, which can impact the performance of the system.

1. **Learning Curve:**

Algorithm visualizer requires some level of technical proficiency to use effectively, which can be a barrier to entry for some users.

**Chapter 9**

**Conclusion**

We can conclude that:

* The project provides a real-time solution for students who struggle to comprehend data structure algorithms.
* In this project, students are empowered to visualize algorithms and achieve a crystal-clear understanding of how they work.
* We can also include additional features in the application:

1. such as the code for each algorithm in multiple programming languages.
2. Students can compare the advantages and disadvantages of two algorithms, allowing them to gain a better understanding of the differences between them.

**Chapter 10**

**Future Scope**

To further enhance the functionality of this system, there are several features that can be integrated.The below list shows the future scope to be considered:

* Expand the range of algorithms that are available for visualization, including more advanced data structures and algorithms.
* Add more customization options for the visualization, such as the ability to adjust color scheme.
* The system could be integrated with machine learning algorithms to provide even more powerful analysis and insights.
* It could be expanded to include interactive coding challenges or quizzes to test student understanding.

By continuing to evolve and improve the algorithm visualizer, it could become an even more valuable tool for students, educators, and professionals in the field of computer science.

**Chapter 11**

**Reference**

* In 2008, paper “AlCoLab: Architecture of Algorithm Visualization System” concerns the style of script supported algorithm visualization systems for educational purposes, focusing on the support and the improvement that those systems provide in the process of teaching of an conceptual subject such as algorithms.
* 2019, paper “Towards Developing an Effective Algorithm Visualization Tool For Online Learning” reports a work-in-progress research project at Athabasca University on developing an effective algorithm visualization tool for online learning.
* In 2019, paper “Open Interactive Algorithm Visualization ” presents a working in-progress project form developing an open interactive algorithm visualization website.

* <https://visualgo.net/en>
* <https://youtu.be/OrnUuS0SUa8>