PATH VISUALIZER

A PROJECT REPORT

SUBMITTED BY:
SUSHANT.A.A (2K19/IT/129)
TANMAY ARORA (2K19/IT/131)

UNDER THE SUPERVISION OF:
MRS. SWATI SHARDA

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi- 110042
NOVEMBER, 2020

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

<u>CANDIDATE'S DECLARATION</u>

We Sushant A.A., Roll No – 2K19/IT/129 & Tanmay Arora , Roll No –2K19/IT/131 , students of B.Tech. (INFORMATION TECHNOLOGY), hereby declare that the project Dissertation titled "Path visualizer" which is submitted by us to the Department of INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the 3rd semester of the Bachelor of Technology, is made by us. This work has not previously formed the basis for the award of any Degree, Diploma Associate, Fellowship or other similar title or recognition.

Place: Delhi

Date: 18-11-2020

Sushant A.A. (2K19/IT/129)

Tanmay Arora (2K19/IT/131)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “Path visualizer” which is submitted by Sushant.A.A, Roll no - (2K19/IT/129) and Tanmay Arora, Roll no - (2K19/IT/131) INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date: 18-11-2020

MRS. SWATI SHARDA
SUPERVISOR

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude to our professor, Mrs. Swati Sharda for giving us this wonderful opportunity to do our project with the objective of "**Path visualizer"**

This project has definitely helped us enhance our horizon of knowledge and has taught us a lot about the multivariable linear regression and other algorithms that can be used to predict the desired output and how they can be used to solve real life issues.
We would also like to take this opportunity to thank our friends and family members for creating an environment and for providing us with the materials required to complete this project.
Lastly, we thank you for your constant guidance and support.

INDEX

Objective of this project:

Path visualization of different graph algorithms and different user inputs like wall, weights, etc. to make it interactive.

INTRODUCTION

The graph data structure is very popular and is used in various fields like Google Maps, handwriting recognition, social media, World Wide Web, operating systems, machine learning, etc. this data structure has immense power but students face many difficulties in learning it at the first time. our project tries to resolve this problem but displaying various algorithms in a 2D grid. it is user-friendly as it has many features like wall creation, weight addition, traps, etc. which makes our project friendly.

DATA STRUCTURE USED

Graphs

A graph data structure consists of a finite (and possibly mutable) set of vertices (also called nodes or points), together with a set of unordered pairs of these vertices for an undirected graph or a set of ordered pairs for a directed graph. These pairs are known as edges (also called links or lines), and for a directed graph are also known as arrows. The vertices may be part of the graph structure, or may be external entities represented by integer indices or references used for making the pathway the connections of nodes on which algorithms will work upon.

Hashmap

The Map object holds key-value pairs and remembers the original insertion order of the keys. Any value (both objects and primitive values) may be used as either a key or a value. It is used in dijkstras , astar , greedy bfs in the project

Arrays

An Array data structure, or simply an array, is a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key. An array is stored such that the position of each element can be computed from its index tuple by a mathematical formula. The simplest type of data structure is a linear array, also called one-dimensional array. 3d array is used to represent the 2d matrix .and different algorithms in the project.

Stack

Stack is an abstract data type that serves as a collection of elements, with two main principal operations push and pop. The order in which elements come off a stack gives rise to its alternative name, LIFO (last in, first out). Additionally, a peek operation may give access to the top without modifying the stack. It is used in dfs algorithm and animation in the project.

Queue

A Queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence. By convention, the end of the sequence at which elements are added is called the back, tail, or rear of the queue, and the end at which elements are removed is called the head or front of the queue, analogously to the words used when people line up to wait for goods or services , used in bfs algorithm and animation in the project

Strings

A string is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation). A string is generally considered as a data type and is often implemented as an array data structure of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. *String* may also denote more general arrays or other sequence (or list) data types and structures It is used for creating ids for the blocks in html dom manipulation in the project.

Algorithms Displayed

DFS

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

BFS

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'[1]), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level

DIJKSTRA’S Algorithm

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm) is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks.

A*

A* (pronounced "A-star") is a graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency. Thus, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance, as well as memory-bounded approaches; however, A* is still the best solution in many cases.

BFS PATH

BFS PATH shows the path traversed by the bfs algorithm to reach the end node form the starting node.

FEATURES

Wall

Using wall user can break the connecting weights between the nodes

Start

User can change the start point

End

User can change the endpoint

Weight 5

User can increases the weight of the edge connecting its neighbor to 5 times

Weight 3

User can increases the weight of the edge connecting its neighbor to 3 times

Traps

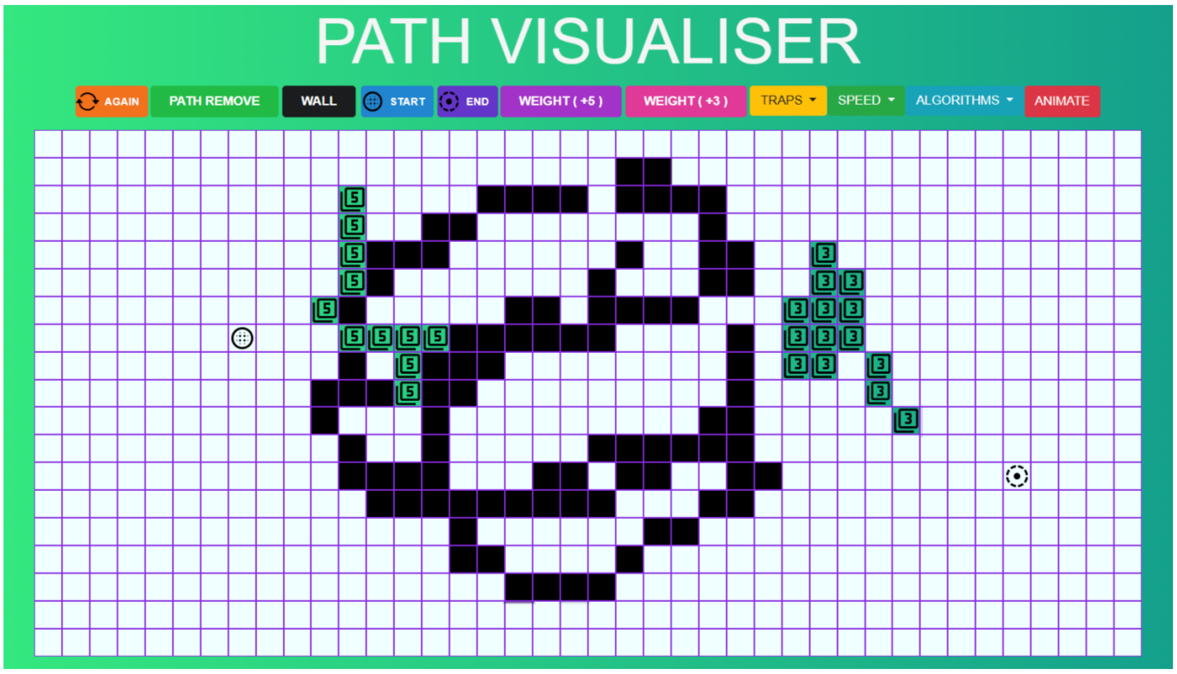
> **Basic random** – Places walls and weights at different random locations in the grid
>
> **Random wall** – Places walls at different random locations in the grid
>
> **Random weight-**Places only weights at a different random location in the grid
>
> **Custom trap** – A self-designed trap

SPEED

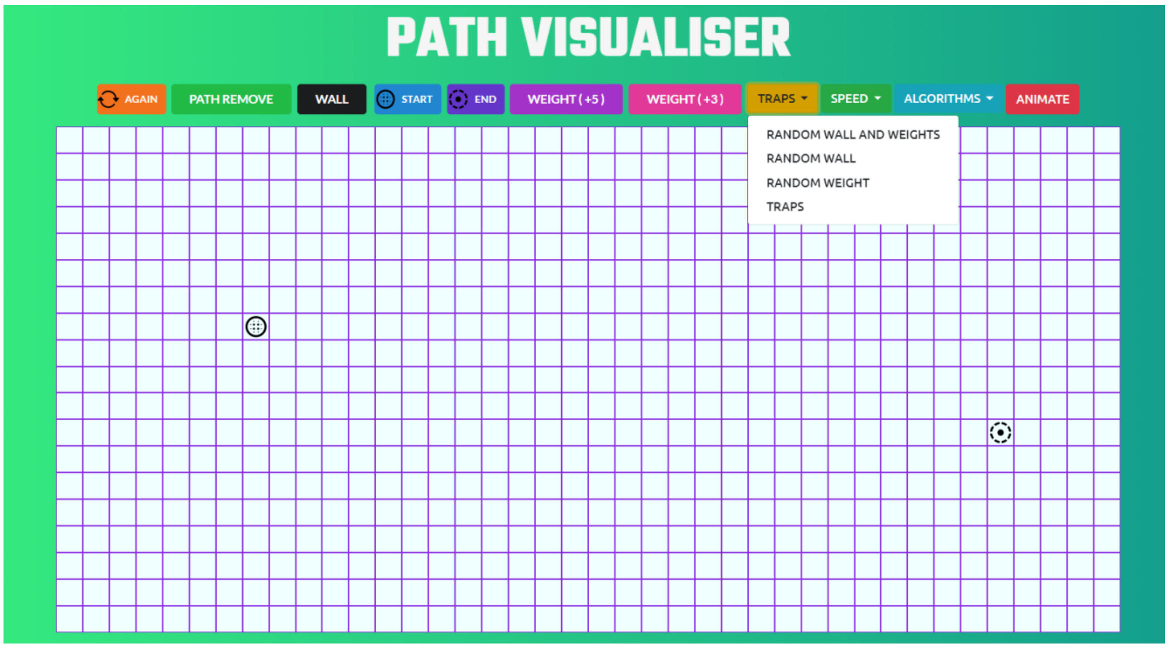
Users can change speed as fast, medium, and slow.

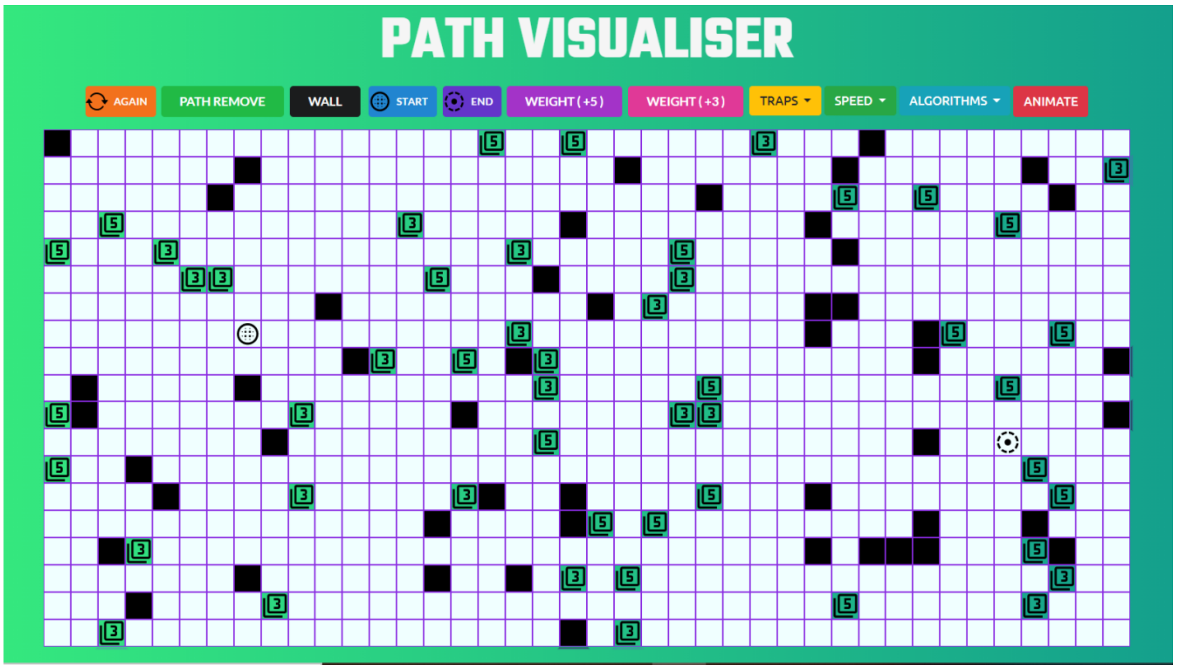
HOW TO USE

ADD RANDOM WALLS AND WEIGHTS

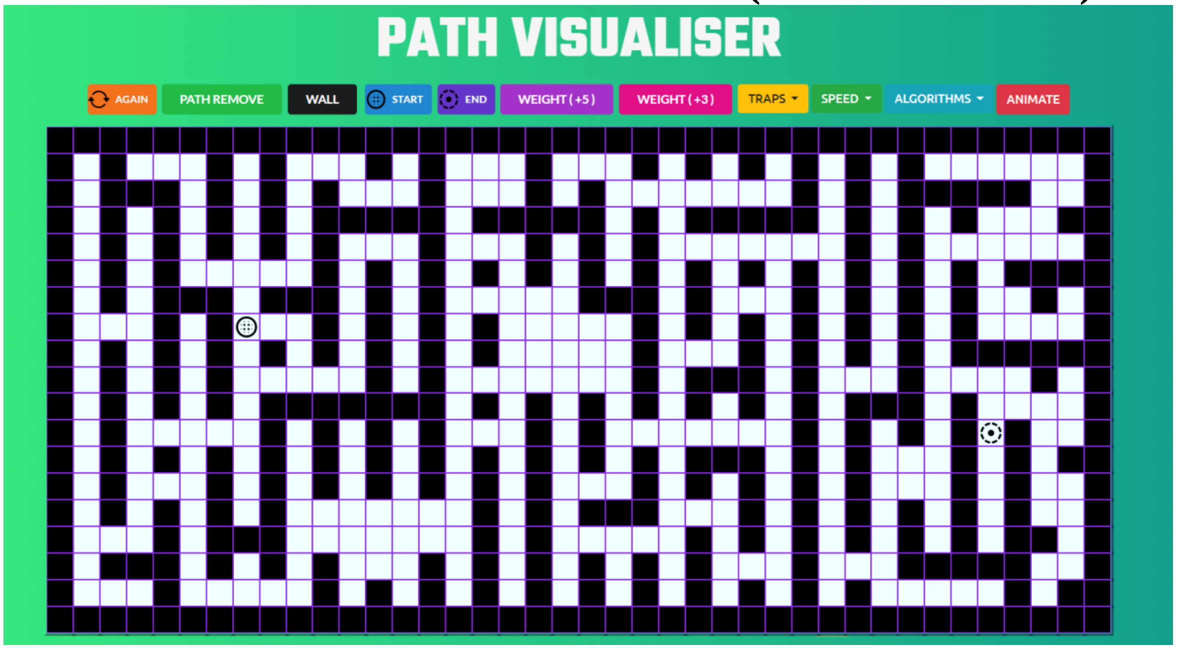


OR ADD RANDOM TRAPS

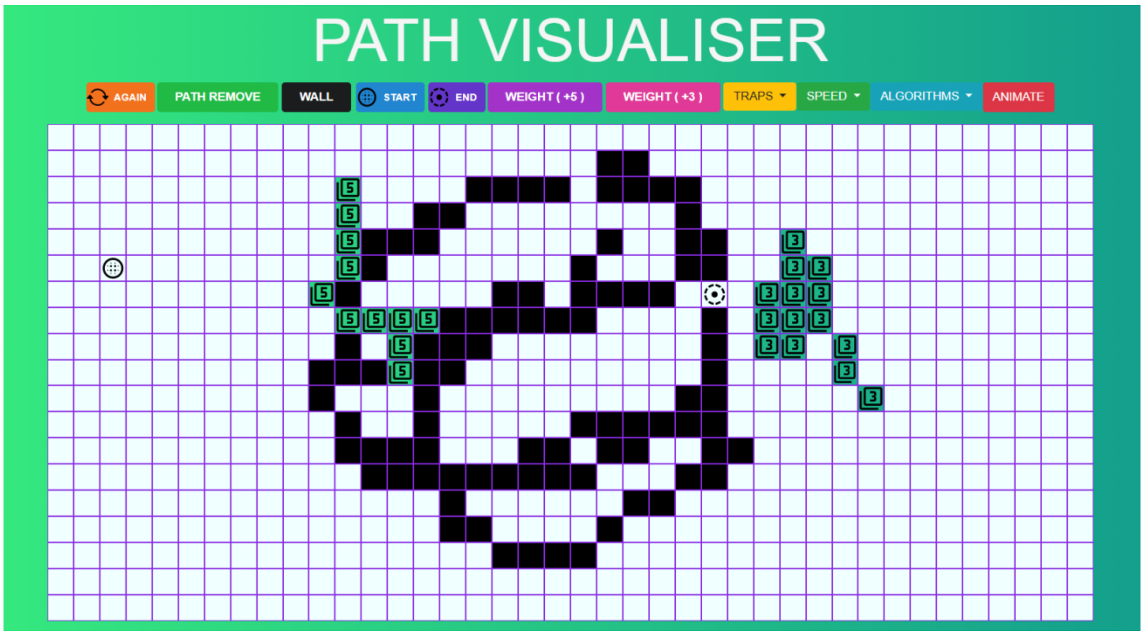




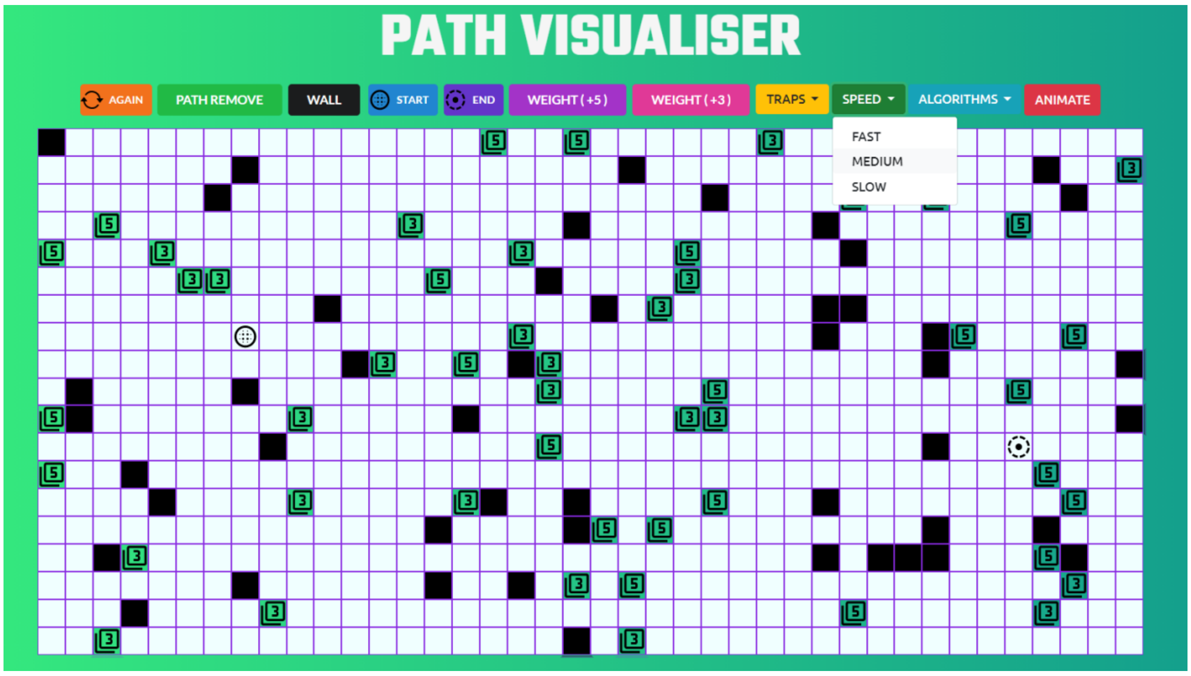
OR ADD OUR SELF CREATED TRAP (TRAPS OPTION)



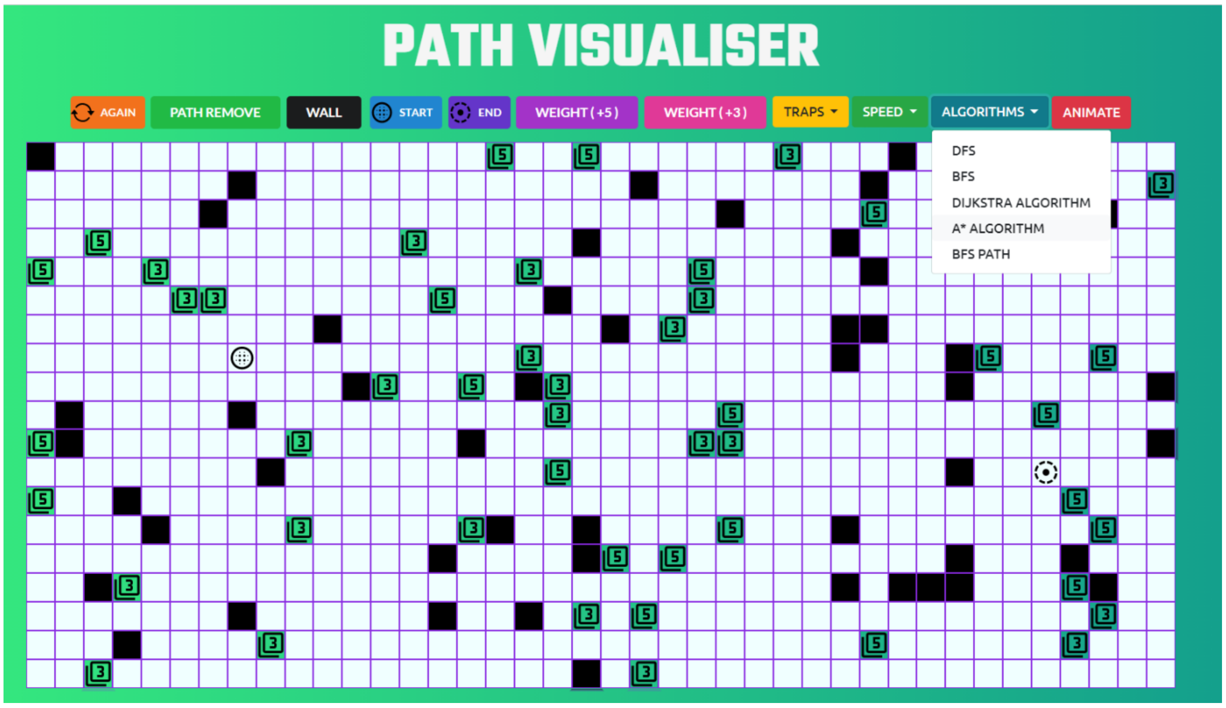
SHIFT STARTING AND ENDING POINT USING START END BUTTONS



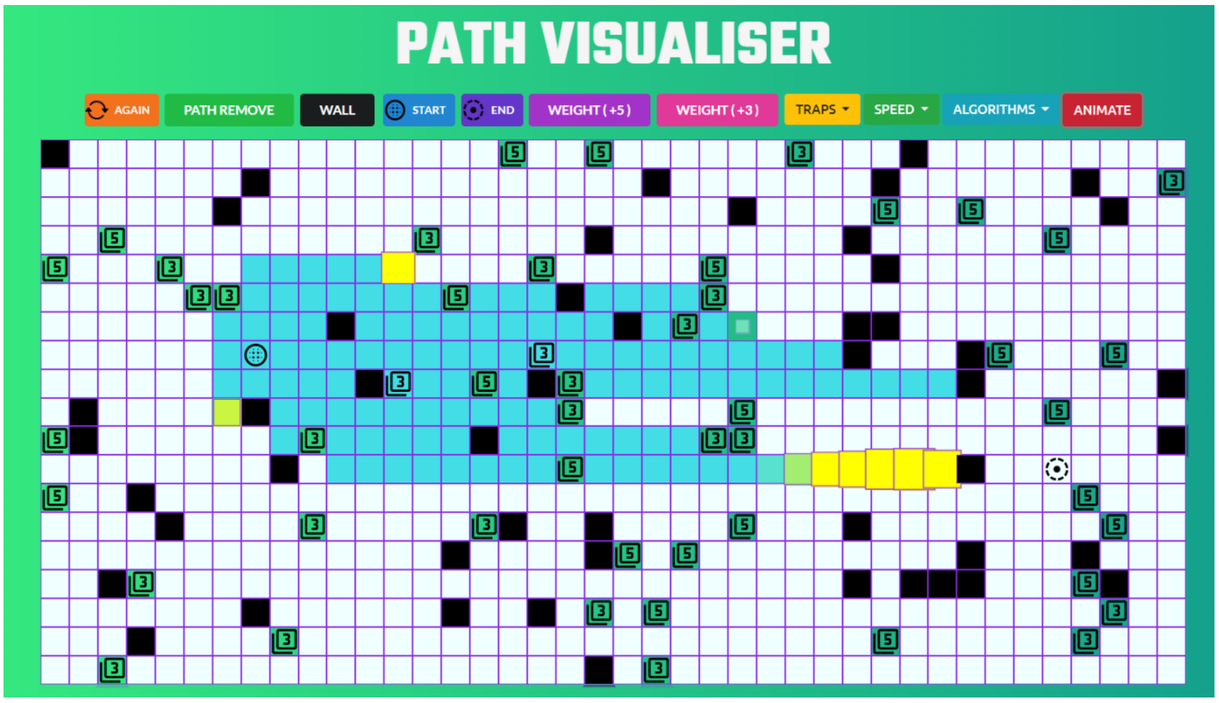
MODIFY SPEED



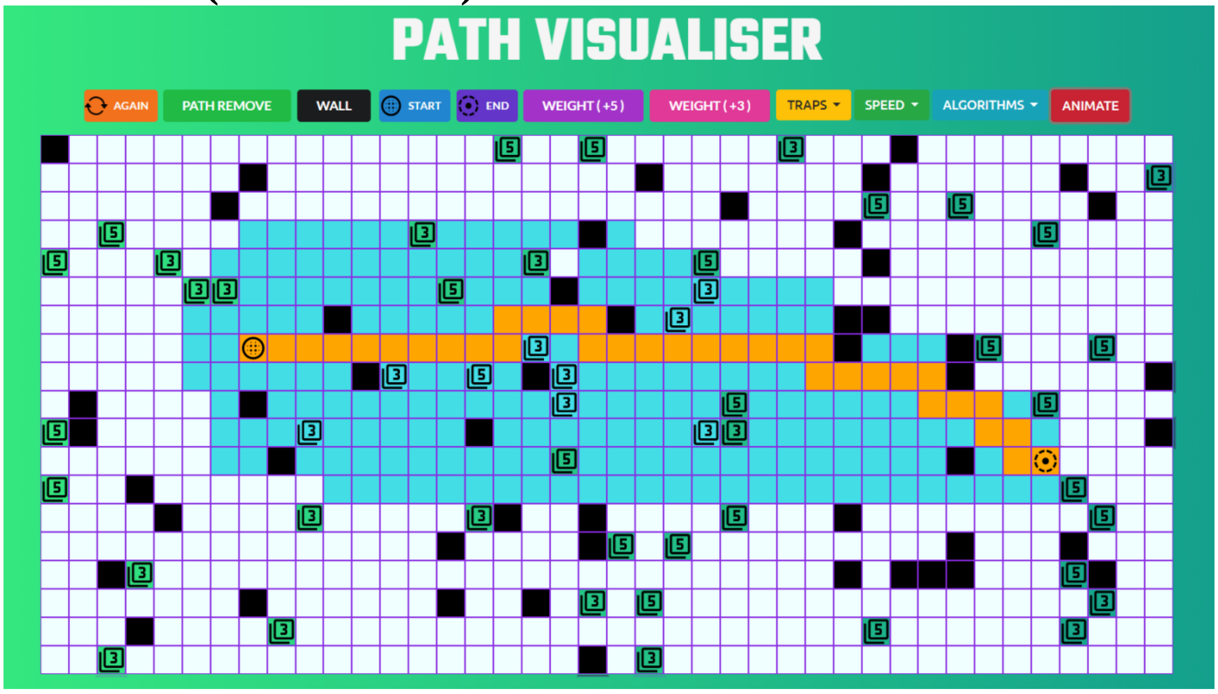
CHOOSE ALGORITHM TO VISUALISE AND CLICK ON ANIMATE



AND THEN VISUALISE

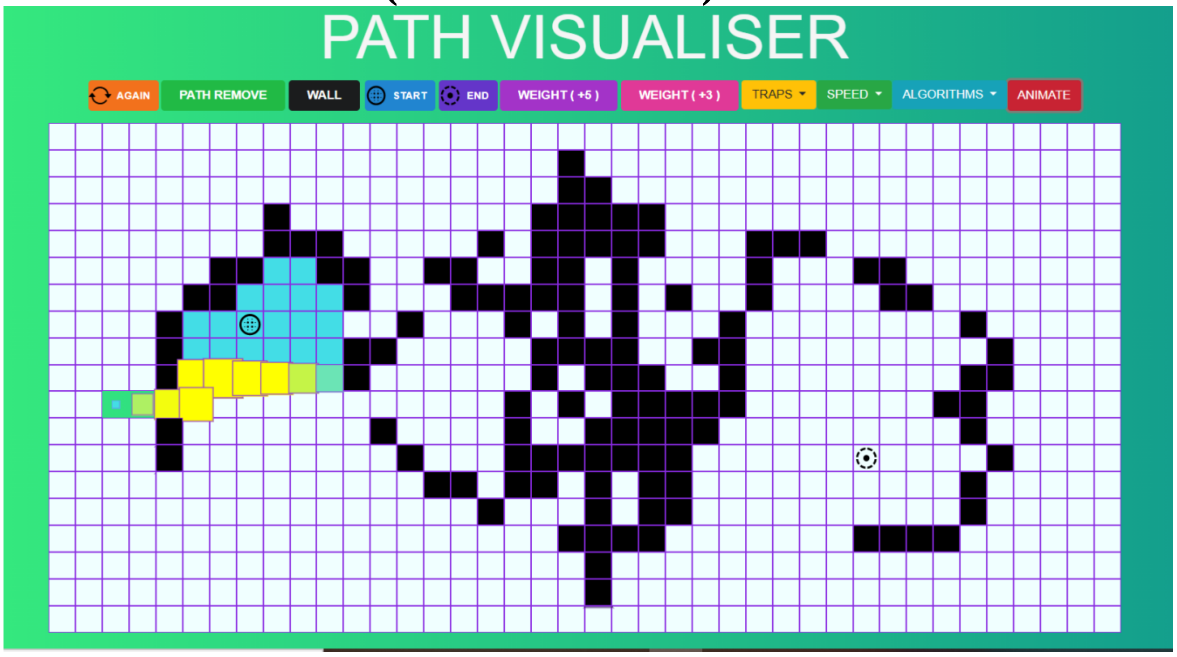


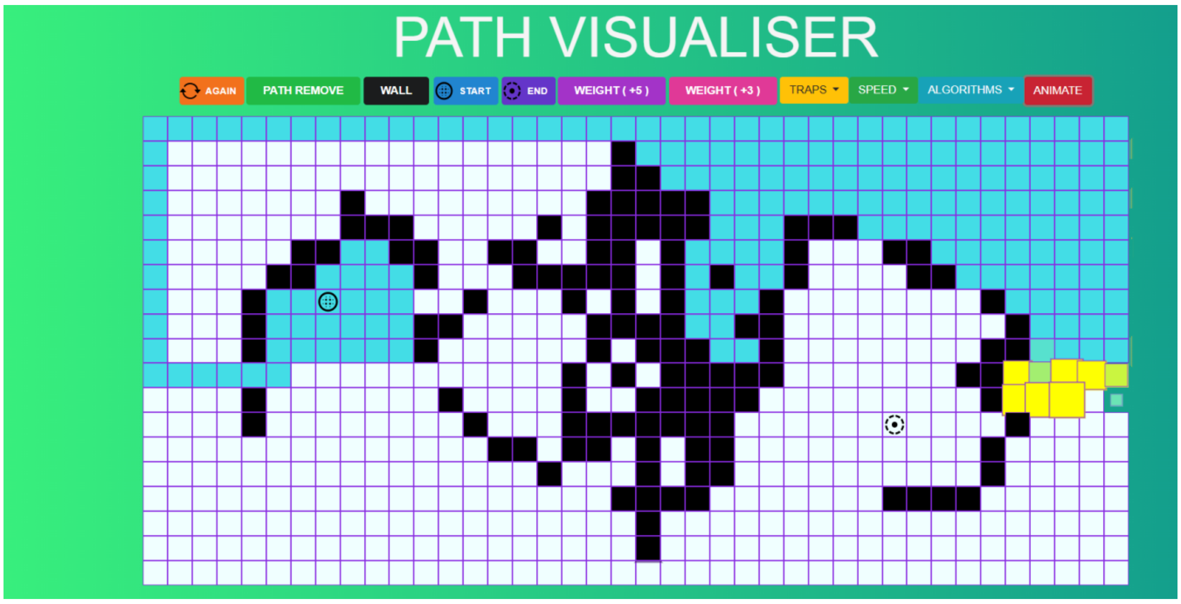
A * HERE (WEIGHTED)

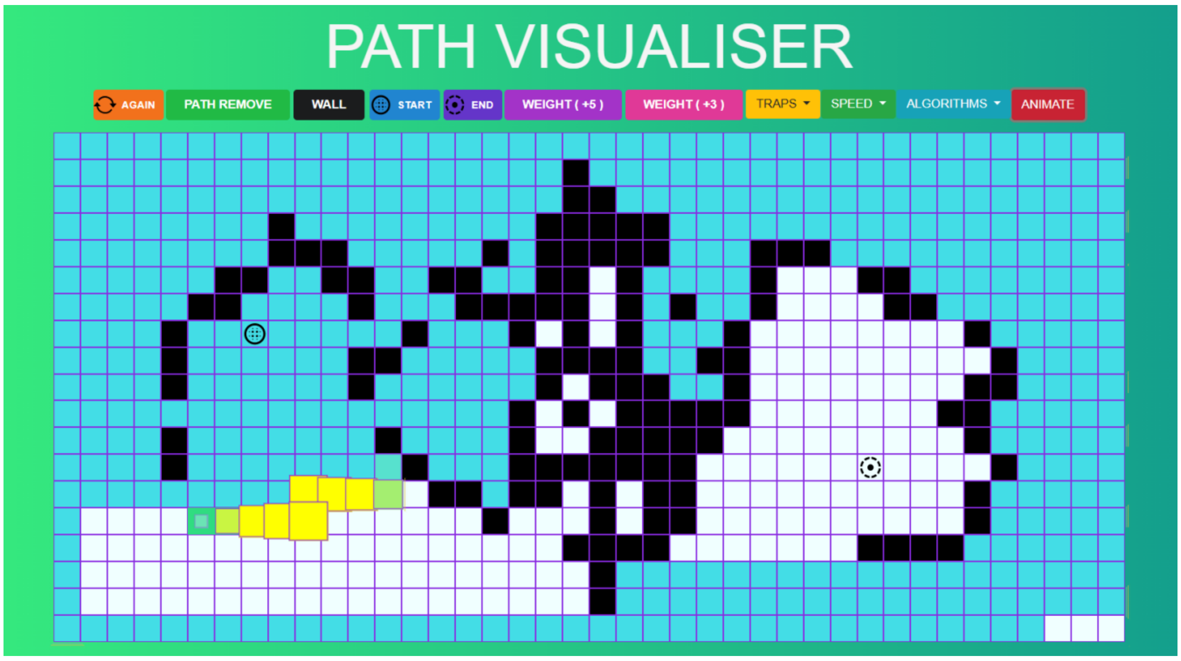


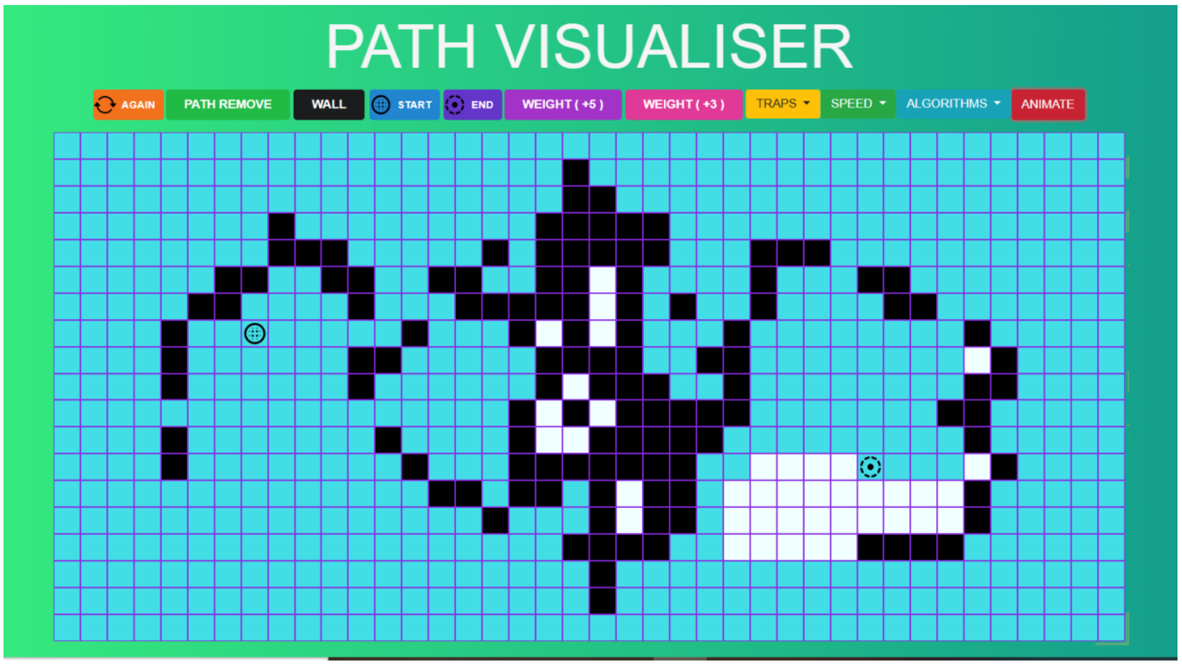
OUTPUTS

VISUALISING DFS (UNWEIGHTED)

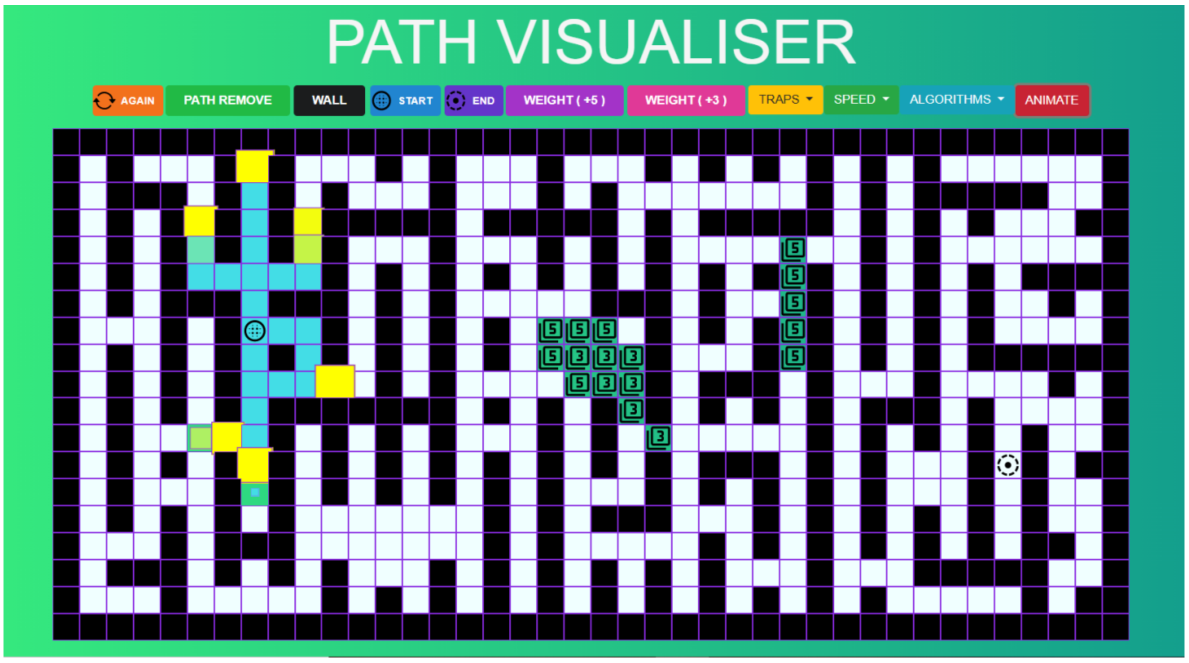


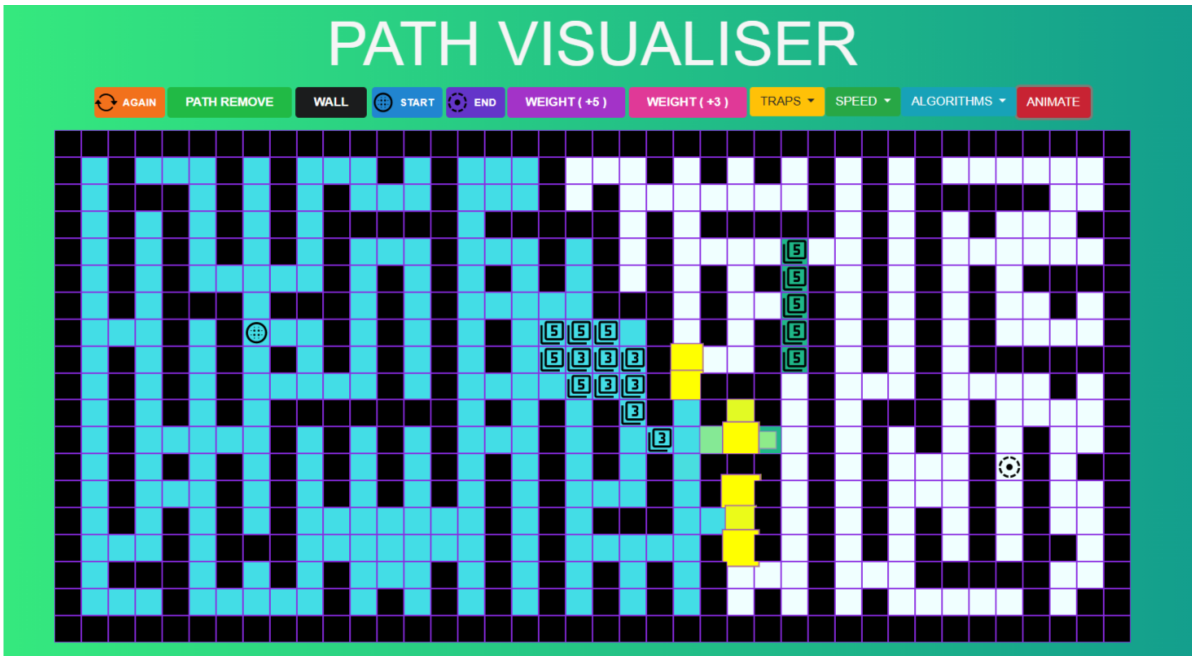

PATH VISUALISER
AGAIN
PATH REMOVE
WALL
START
END
WEIGHT (+5)
WEIGHT (+3)
TRAPS
SPEED
ALGORITHMS
ANIMATE


PATH VISUALISER
AGAIN
PATH REMOVE
WALL
START
END
WEIGHT (+5)
WEIGHT (+3)
TRAPS
SPEED
ALGORITHMS
ANIMATE

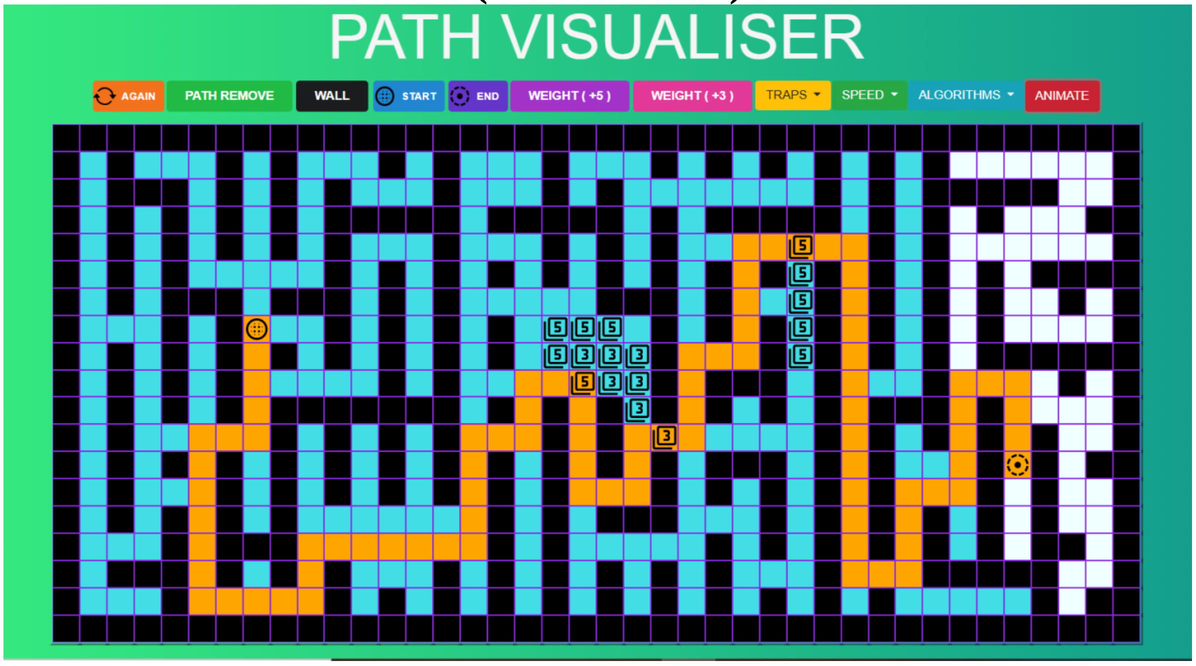


VISUALISING DIJKSTRA

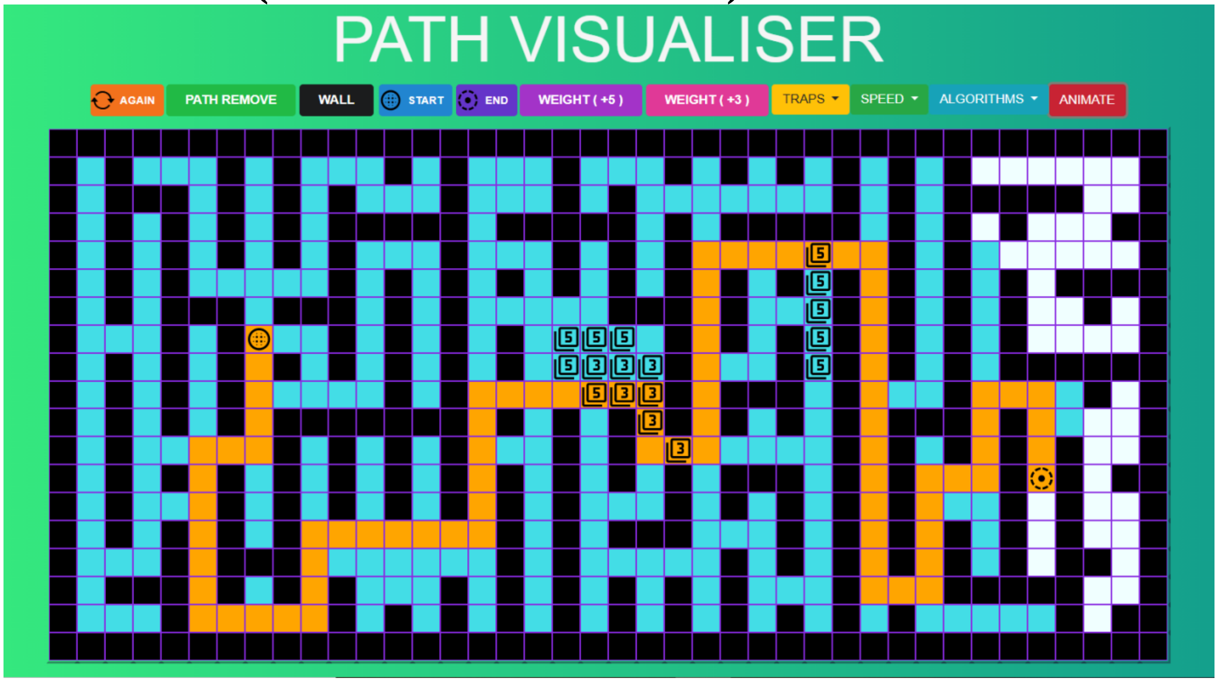




OUTPUT OF DIJKSTRA(WEIGHTED)



VS FOR BFS (PATH - UNWEIGHTED)



CODE

Github repository - https://github.com/SushantAA/Path-visualizer
Website link - https://sushantaa.github.io/Path-visualizer/index.html

BIBLIOGRAPHY

https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
https://www.w3schools.com/js/
https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
https://fomantic-ui.com/introduction/getting-started.html
https://api.jquery.com/