PATH FINDER AND VISUALIZER

BY

M V PRANAV 18BCE1057

C KOUSHIK 18BCE1227

HARSHAAN MURALI 18BCE1289

A project report submitted to

DR. R. RAJALAKSHMI
SCSE
as Project Component of the course titled

CSE2003- DATA STRUCTURES AND ALGORITHMS

B.Tech. COMPUTER SCIENCE ENGINEERING

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

NOVEMBER 2019

BONAFIDE CERTIFICATE

Certified that this project report entitled **“PATH FINDER AND VISUALIZER”** is a bonafide work of **M V PRANAV (18BCE1057), KOUSHIK C (18BCE1227)** and **HARSHAAN M (18BCE1289)** who carried out the project work under my supervision and guidance.

Dr. R. RAJALAKSHMI

School of Computer Science Engineering

(SCSE)

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

An effective method to find the shortest distance between two points (starting point and destination) is necessary. Our project is aimed at doing this, considering roads to be laid in a grid wise arrangement. The code also involves the automatic generation of obstacles around which the nodes are traversed.

The user is allowed to enter the source and destination from a matrix to which the routing is done and the shortest path between the two points is graphically represented in a grid. The path is also highlighted with a different colour for easy viewing.

This programme can be applied in real life for route optimizations.

ACKNOWLEDGEMENT

We are extremely grateful to **Prof. JAGADEESH KANNAN R** Dean of the School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr. R Rajalakshmi, School of Computer Science and Engineering, for her consistent encouragement and valuable guidance.

We also take this opportunity to thank all the faculty of the School for their support and wisdom they imparted to us throughout the course.
We thank our parents, family, and friends for the opportunity they provided us in undertaking this course in such a prestigious institution.

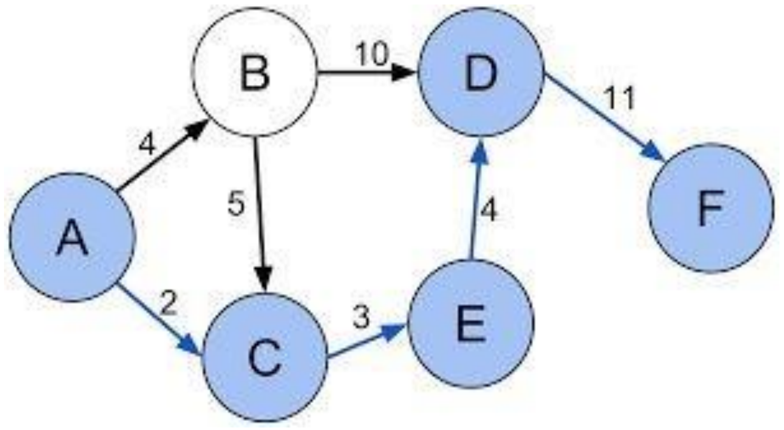
M V PRANAV
KOUSHIK C
HARSHAAN M

TABLE OF CONTENTS

INTRODUCTION

Routing

Routing is the process of finding the best path between two or more locations with a fixed order in a road or rail network. The criterion according to which a path is the best can vary. You may be looking for the shortest path (by distance), the fastest (by travel time), but also the most scenic or the safest path. Anything is possible as long as it can be specified by assigning some quantity, a generalized cost , to each road segment, and looking for the least cost path.



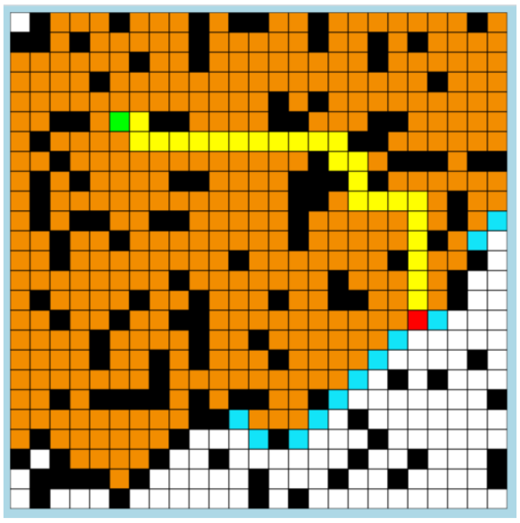
"**Path Finder and visualizer**" is the process of displaying the shortest path from the source vertex and destination vertex. In this project, an n*n matrix is formed, and the shortest path between the source and destination is calculated and visualized using the implemented algorithms.

INPUT (as code) : Vectors/position of start and end points Probability of occurrence of obstacles/walls.

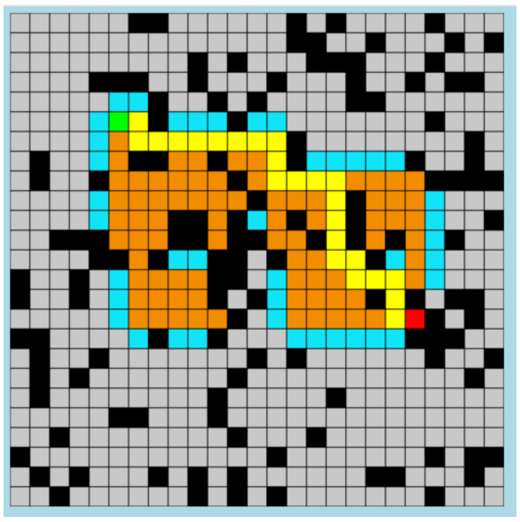
OUTPUT: Shortest path between source and destination visualized in a grid.

OUTPUT SCREENSHOTS

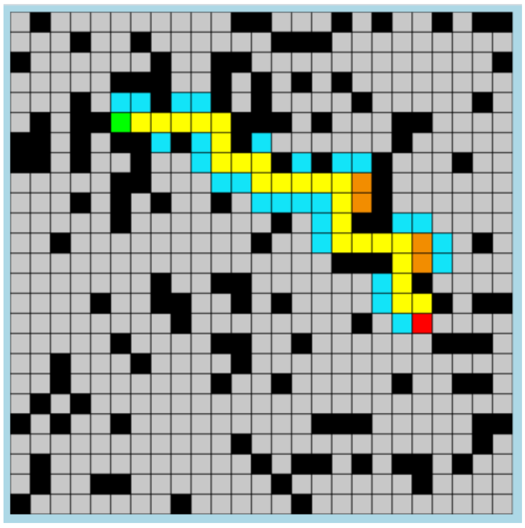
Dijkstra- Output



Astar(A*)-Output



Greedy- Output



WORKING

The program finds the shortest path between the source and destination by implementing one of three algorithms -

DIJKSTRA between two vertices of a graph where all edges have nonnegative weight. It is based on Dijkstra's Algorithm is an algorithm for computing the shortest path

repeatedly expanding the closest vertex which has not yet been reached. Since it traverses through all nodes of the graph, it is considered less efficient.

GREEDY ALGO - Greedy algorithm selects the next best node whose distance to destination is minimal. Unlike Dijikstra, it only traverses towards the direction of the end vertex. Hence it is more efficient than Dijkstra, but may not provide the most efficient path.

A-STAR -A* algorithm picks the node according to a value-‘f’ which is a parameter equal

to the sum of two other parameters – ‘g’ and ‘h’. At each step it picks the node/cell having the lowest ‘f’, and process that node/cell. We define ‘g’ and ‘h’ as simply as possible below g = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there. h = the estimated movement cost to move from that given square on the grid to the final destination.

The specified paths are visualized using JavaScript p5 Library.

APPLICATION

Shortest path algorithms are applied to automatically find directions between physical locations, such as driving directions on web mapping websites like MapQuest or Google Maps. For this application fast specialized algorithms are available[

If one represents a nondeterministic abstract machine as a graph where vertices describe states and edges describe possible transitions, shortest path algorithms can be used to find an optimal sequence of choices to reach a certain goal state, or to establish lower bounds on the time needed to reach a given state. For example, if the vertices represent the states of a puzzle like a Rubik's Cube and each directed edge corresponds to a single move or turn, shortest path algorithms can be used to find a solution that uses the minimum possible number of moves.

CONCLUSION, FUTURE SCOPE AND RESULT

CONCLUSION

The Path Finder and Visualizer programme responds correctly to all sources and destinations and provides the shortest path between the former and latter. The highlighted route helps distinguish itself from the grid and makes it easier for the user.

FUTURE SCOPE

1. We can increase the working of the programme to any size of matrix.

2. Multiple destination points can be added for a single traversal.

RESULT

Path Finder and Visualizer code is executed and the shortest path is found between the specified start and end points. Hence, the result is obtained.

REFERENCES

1. https://stackoverflow.com/questions/31695202/the-greedy-algorithm-and-implementation
2. https://www.geeksforgeeks.org/a-search-algorithm/
3. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
4. https://www.w3schools.com/jsref/jsref_reference.asp