CSE2003 - Data Structures and Algorithms

Analysis Of Shortest Path Algorithms

J - Component Presentation

By -

Andrew John - 18BEC1278

Arjun Sharma - 18BEC1117

Presented to Dr. Mirza Galib Anwarul Husain Baig

School: SCOPE

Abstract

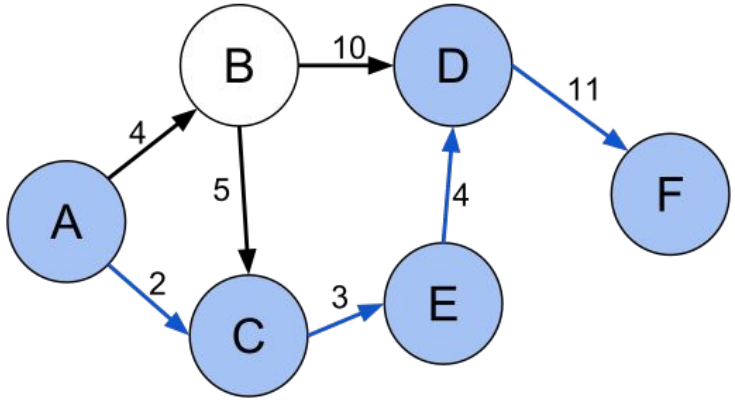
In this project we have proposed an effective method to find the shortest distance between two points (starting point and stop or destination point) that's necessary and the project is aimed at doing this, considering roads to be laid in a grid-wise arrangement. The algorithm also involves the automatic generation of obstacles around which the nodes are traversed and thus plots various different shortest paths. The user is allowed to enter the source point and destination point from a matrix table through which the routing is done and the shortest path between the two points is graphically represented in a grid. The shortest path plotted is highlighted with a different color for ease of visualization. The proposed program can be applied in real life for route optimizations.

Introduction and Objective

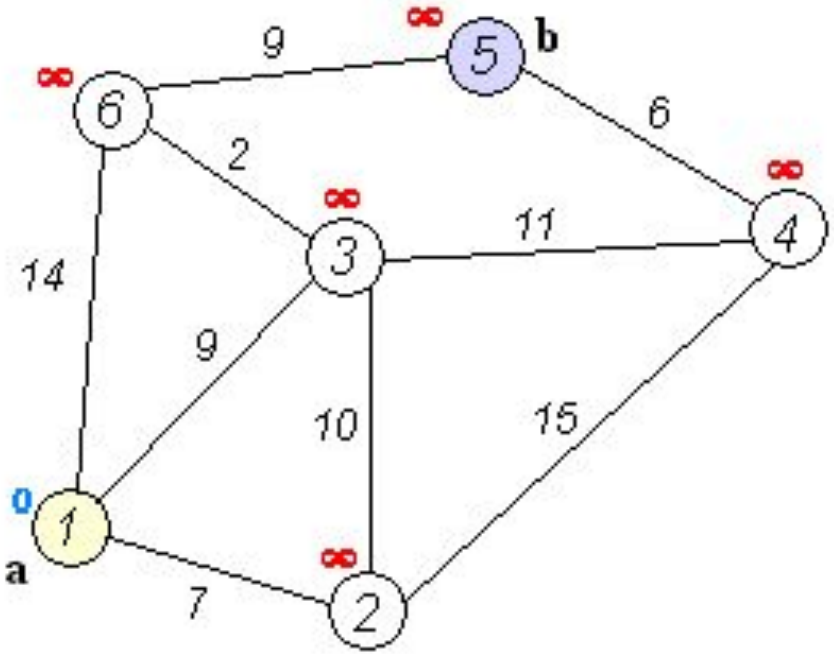
- Routing is the process of finding the best path between two or more locations with a fixed order in a path or network given. The criterion according to which a path is the best can vary. The user may be looking for the shortest path (by distance), the fastest (by travel time), but also the most scenic or the safest path.
- The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs, the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

Introduction and Objective

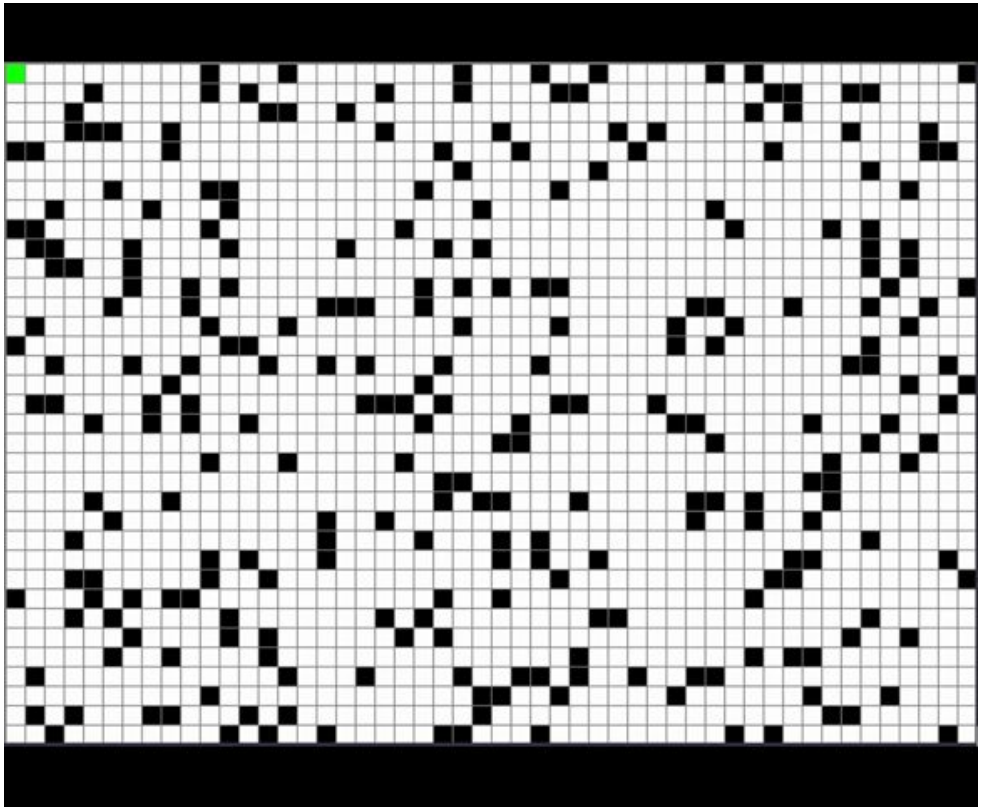
- In this project, we analyze Dijkstra’s algorithm, A* Star Algorithm, and Greedy Algorithm. Dijkstra's algorithm solves the single-source shortest path problem with non-negative edge weight. A* search algorithm solves for single-pair shortest path using heuristics to try to speed up the search. The greedy algorithm is an algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.



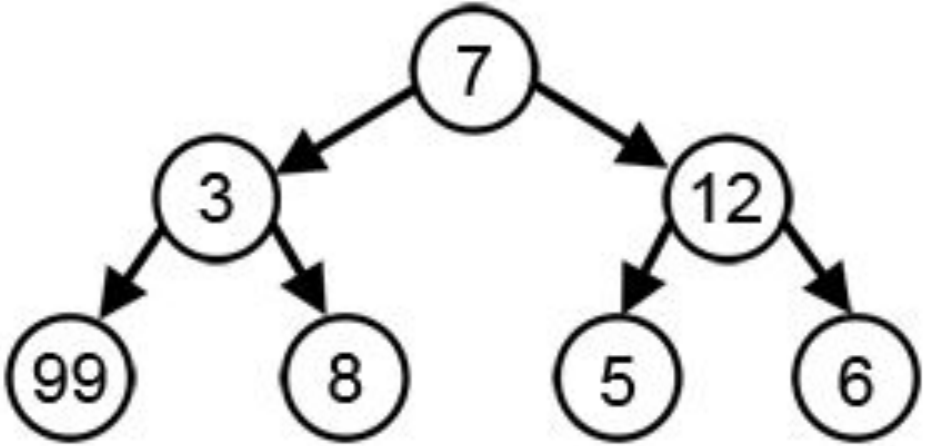
Dijkstra Algorithm Visualization



A Algorithm Visualization*



Greedy Algorithm Visualization



Problem Statement

- In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. The problem of finding the shortest path between two intersections on a road map may be modeled as a special case of the shortest path problem in graphs, where the vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of the segment.
- So, here we wanted to simplify major aspects of solving the problems in the shortest path algorithms with our novel approach and thus it can be implemented in various applications.

Approach Taken / Our Contribution

- The programming language used in this project is JavaScript and the visualizer is visualized in the markup language, HTML and JavaScript p5 Library
- In this project, we are performing a pathfinder for the Shortest path algorithm, and a visualizer for visualizing the route in which the process of displaying the shortest path from the source vertex and destination vertex. An n*n matrix is formed, and the shortest path between the source and destination is calculated and visualized using the implemented algorithms that are Dijkstra, A* and Greedy 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 of n*n matrix.

Approach Taken / Our Contribution

FUNCTIONS USED:

function setup()

function removeElement()

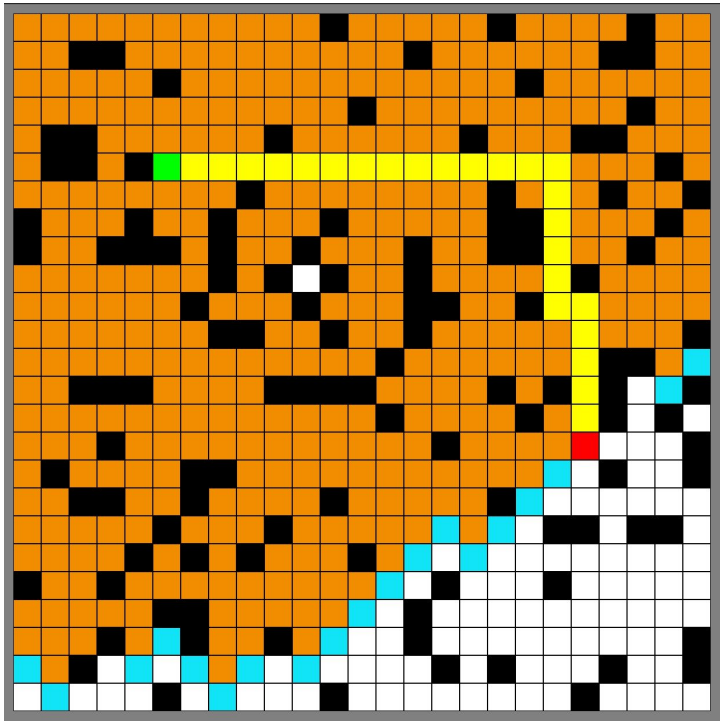
function shortestPath()

function heuristic()

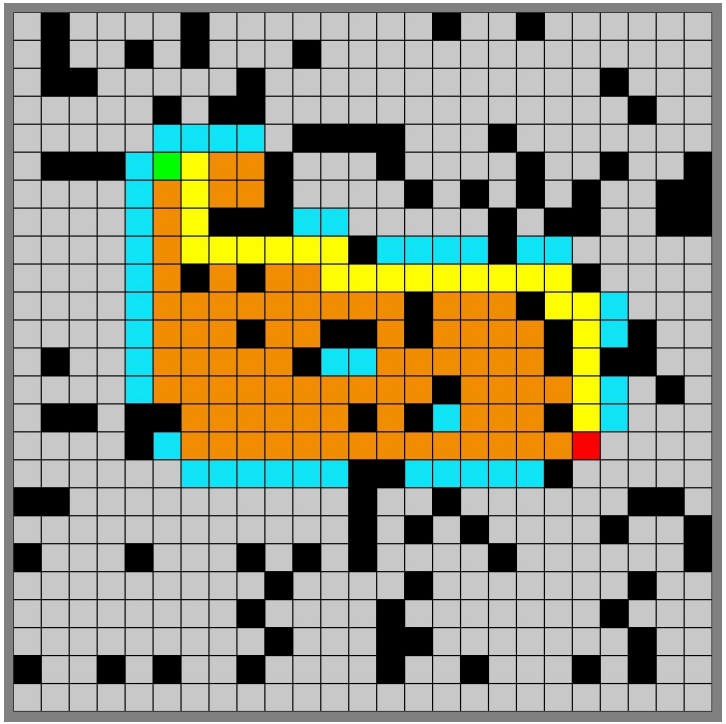
function draw()

function drawfinal()

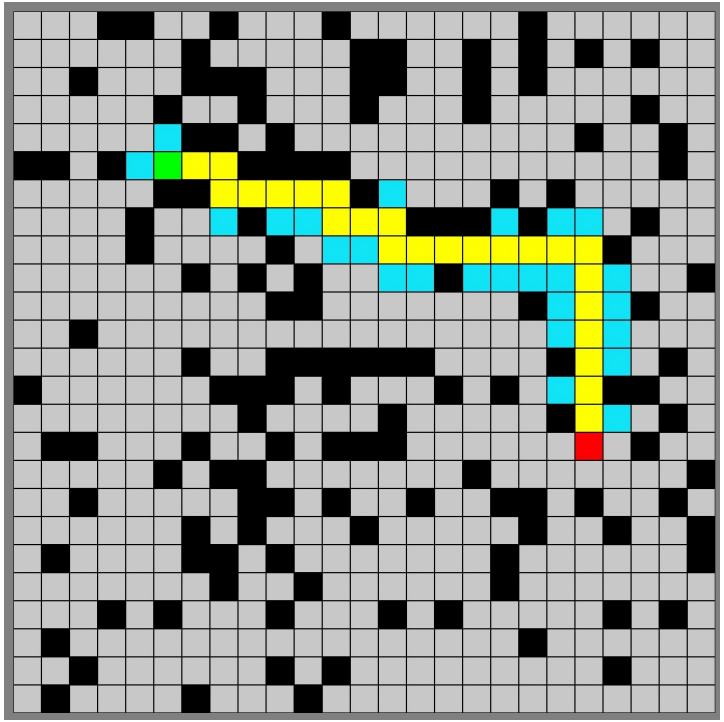
Results Obtained - Dijkstra Algorithm



Results Obtained - A Algorithm*



Results Obtained - Greedy Algorithm



Applications of Shortest Path Algorithms

- 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.

Real-Life Applications of the Discussed Algorithms

Dijkstra's Algorithm

1. Digital Mapping Services in Google Maps
2. Social Networking Applications
3. Telephone Network
4. IP routing to find Open shortest Path First
5. Flighting Agenda
6. Designate file server
7. Robotic Path

A* Algorithm

1. Originally designed as a general graph traversal algorithm.
2. Pathfinding in Video Games.
3. Using stochastic grammars in NLP.
4. Informational search with online learning.

Greedy Algorithm

1. Finding an optimal solution.
2. Finding close to the optimal solution for NP-Hard problems like TSP.

Conclusion and Future Scope

- The path finder and visualizer programs for *Analysis of Shortest Path Algorithms* responds correctly to all sources and destinations as per their definitions 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 to comprehend.

1. Increasing the working of the programme to any size of matrix is the first future scope and extends analysis for the same if achieved.

2. Multiple destination points can be added for a single traversal and is the second future scope.

Thank You!