Grid-Navigator

A Graph-Theoretic Pathfinding Simulator

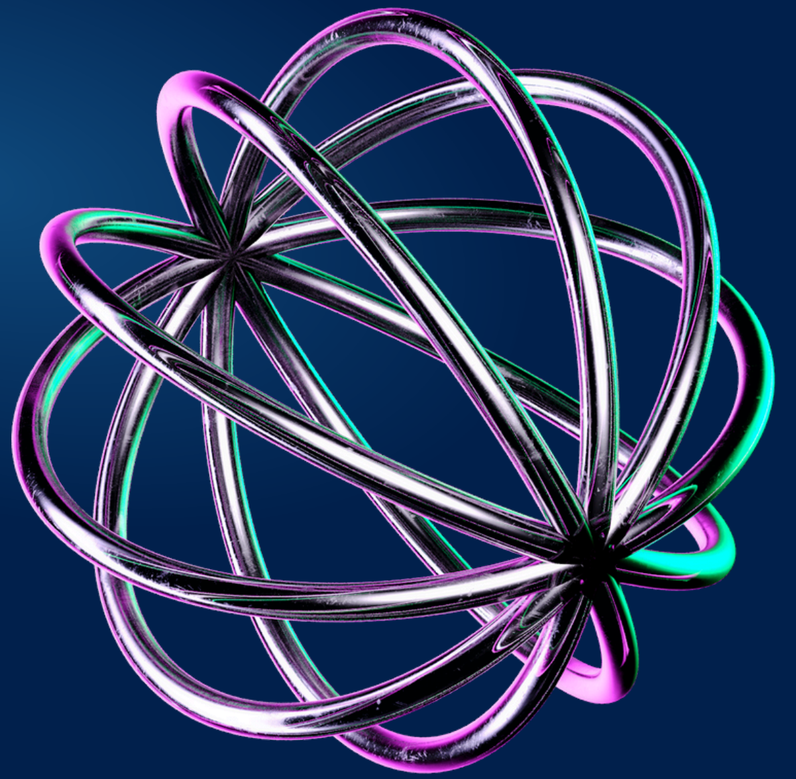
Instructor - Dr. Suchetna Chakroborty

Mentor TA - Arnav Sharma

Problem Statement

Problem -

- Finding the most efficient path between two points in a grid, which maps to many real-world problems (GPS navigation, robotics, gaming AI).



Relevancy -

- From navigation systems like Google Maps and self-driving cars to AI behaviors in games, drone route optimization, and robotics, efficient pathfinding is crucial.

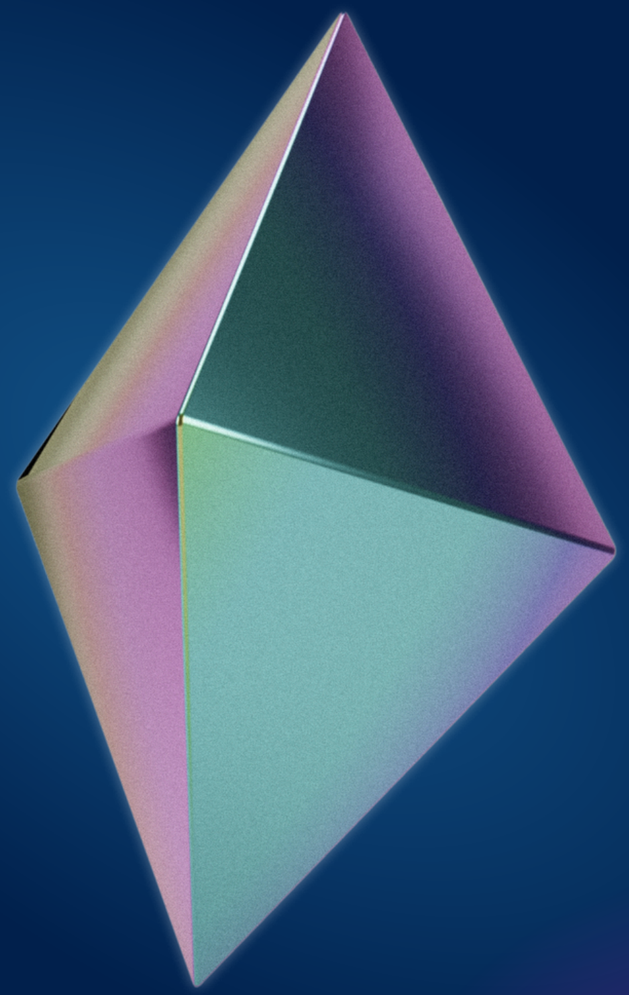
Challenges in Pathfinding

Algorithmic Trade-offs

- Pathfinding isn't a one-size-fits-all solution. Every algorithm balances speed (time complexity), memory consumption (space complexity), and path optimality.

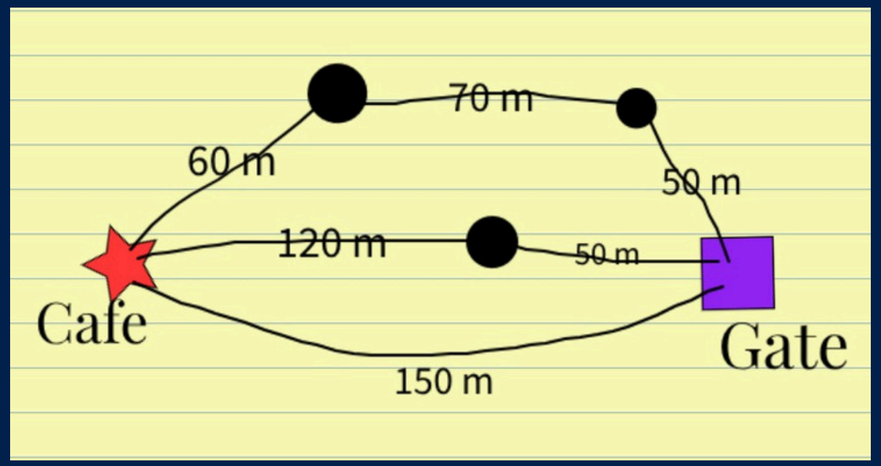
Visualization Complexity

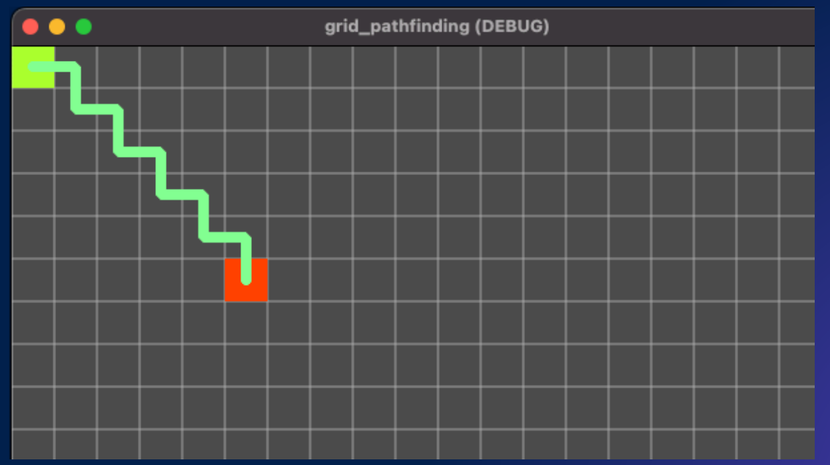
- Graph algorithms are abstract and operate over node-link structures.
- Debugging logic errors in algorithms like A* becomes easier when you can see which nodes are being visited and in what order.



Why choose a Data-Driven Solution?

- Pathfinding algorithms are *inherently data-driven* : they respond dynamically to the grid structure, wall placement, start/end positions, and weighting (if any).
- Visual feedback allows for instant verification of correctness, understanding of performance bottlenecks, and algorithmic insight:
  - *Users can see how many nodes are explored.*
  - *They can compare path lengths and computation times visually.*
  - *It helps debug (e.g., is the algorithm unnecessarily revisiting nodes?).*

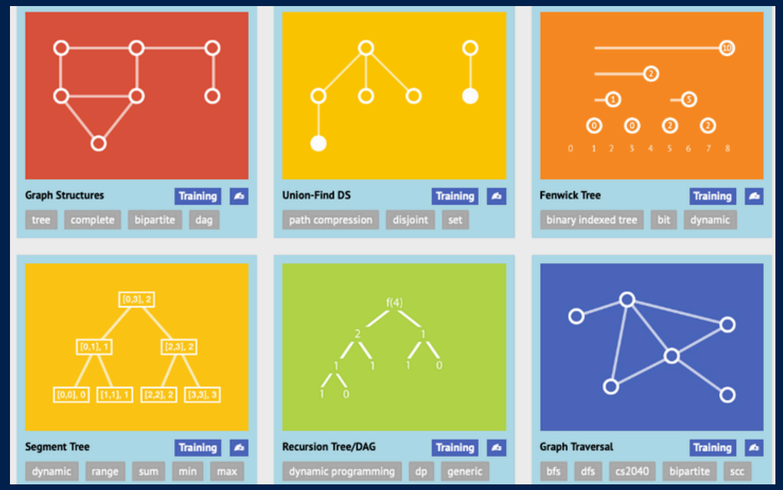




Existing Tools / Visualizers

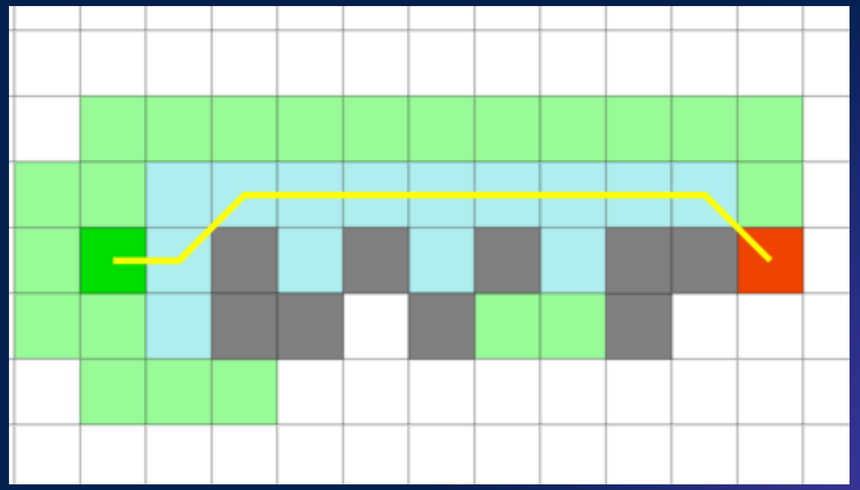
1. *VisuAlgo – DSA Visualizer*

- ***Website:*** VisuAlgo
- ***What it Offers:***
  - Academic-grade animations for BFS, DFS, Dijkstra's, etc.
  - Clean UI, step-by-step animation (suitable for teaching)
  - Displays adjacency list/matrix + runtime info
- ***Limitations:***
  - Not interactive for drawing or editing graphs manually
  - No maze generation or freeform grid setup



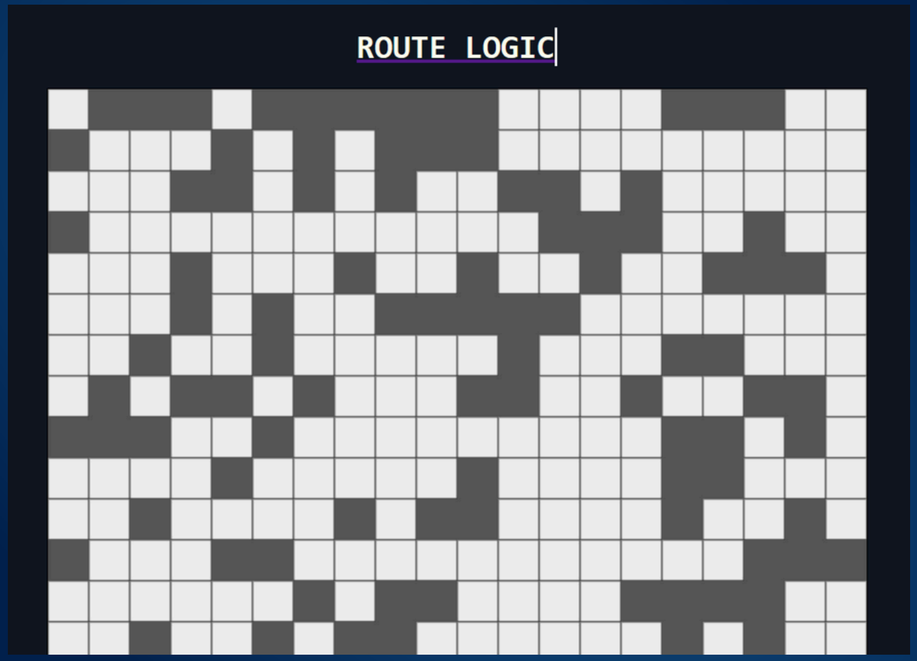
2. *MIT Maze Solver Demo*

- ***Website:*** MITMaze
- ***What it Offers:***
  - Mostly a pre-programmed bot solving a static maze
  - Visualizes robot movement in solving a puzzle
- ***Limitations:***
  - Not customizable — can't change maze structure
  - Focuses on physical constraints, not algorithm comparison
  - No algorithm choice for users



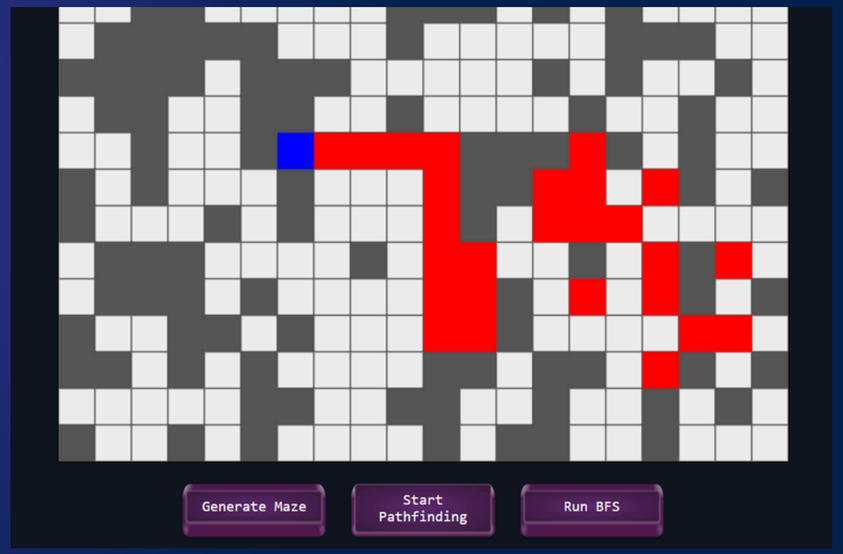
Our Idea

- ***Proposed Solution:***
  - A dynamic, web-based grid simulator where users can draw walls, set start/end points, and observe BFS, DFS, Dijkstra's, and A* in action.
- ***What's Improved?***
  - *Interactive grid editing and Maze generation*
  - *Real-time animation of each step*
  - *Lightweight, responsive UI*
- ***Data Structures Used:***
  - Queue (BFS), Stack (DFS)
  - Min-Priority Queue (Dijkstra)
  - Heuristic Search (A*)
- ***GitHub Repository Link*** - *https://github.com/hecker-200/DSAFINAL.git*

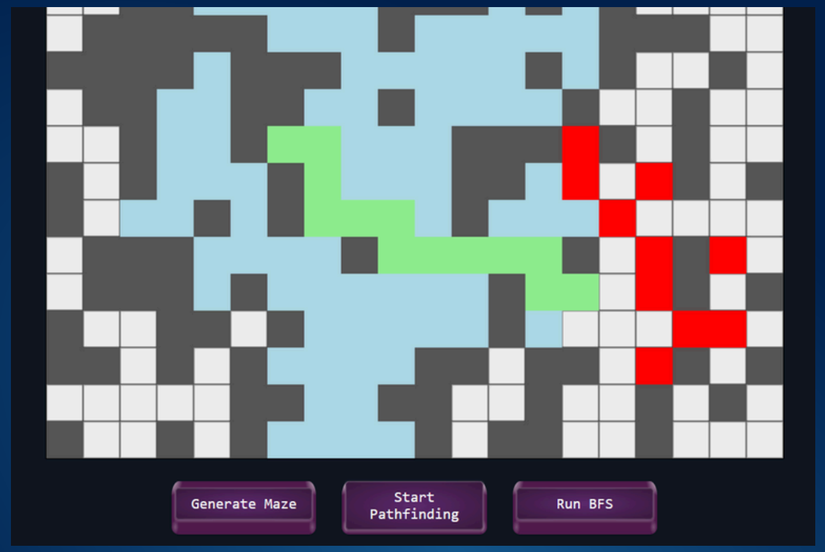


Live Demo : Link

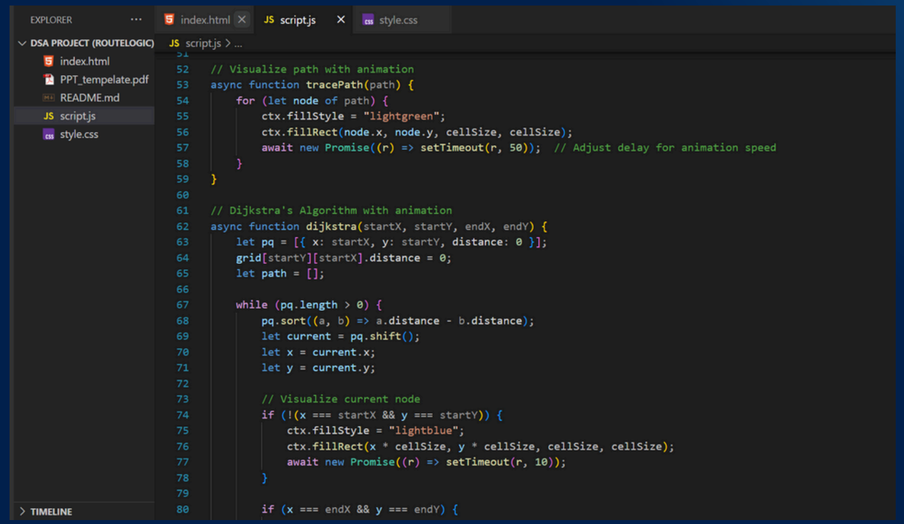
Demonstration



Generate Maze



Start Pathfinding



Backend

Algorithms Used

ALGORITHMS

Dijkstra's Algorithm

A weighted algorithm that guarantees the shortest path by visiting the smallest known distance nodes.

Strengths

Always finds shortest path

Weaknesses

Slower, explores all directions

Breadth-First Search (BFS)

Explores all neighboring nodes at the current depth before moving to the next level using a queue.

Strengths

Shortest path for unweighted graphs

Weaknesses

Memory intensive

Depth-First Search (DFS)

Explores as far as possible along each branch before backtracking, using a stack data structure.

Strengths

Less memory usage

Weaknesses

No guaranteed shortest path

A* Search Algorithm

Strengths

Optimal and complete if heuristic is admissible.

Weaknesses

Can be slow and memory intensive in large graphs.

**This snapshot is from Grid-Navigator's website itself.*

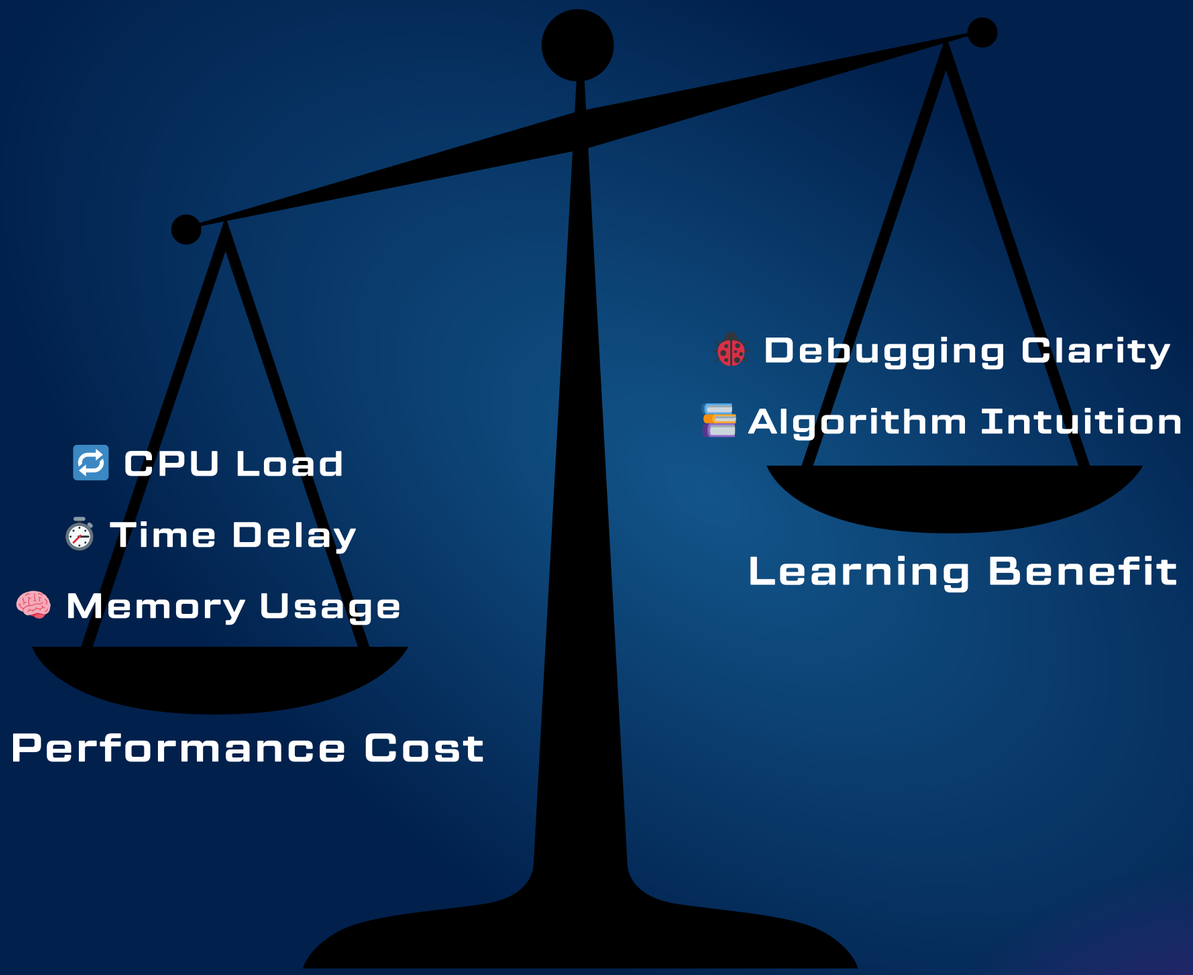
Algorithmic Comparision

Algorithm	Time Complexity	Space Complexity	Optimal Path	Speed	Notes
BFS	O(V + E)	O(V)	✅ Yes	🟨 Medium	Best for unweighted graphs
DFS	O(V)	O(V)	❌ No	🟩 Fast	May get stuck on deep paths
Dijkstra’s	O(E*logV)	O(V)	✅ Yes	🟥 Slow	Always finds shortest path
A*	O(E*logV)	O(V)	✅ Yes	🟩 Fastest	Heuristic-guided; optimal if admissible

Cost-Benefit Trade-off

- ***Languages Used:*** HTML5, CSS3, JavaScript (ES6), C++
- ***Libraries/Tools:*** GSAP for smooth animations
- ***Canvas API:*** For grid rendering

- ***Cost:***
  - Rendering each cell animation costs time
  - Dijkstra/A* slightly slower with full-grid updates
- ***Benefit:***
  - *Clear understanding of algorithm behavior*
  - *Interactive learning tool*
  - *Lightweight – runs on browser, no setup required*



Weighing Scale Visualization

How Grid-Navigator Stands Out ?

Feature	*VisuAlgo*	*Pathfinding.js*	*MIT Maze Solver*	*Grid-Navigator*
Draw/Erase Walls	❌	✅	❌	✅
Set Custom Start/End	❌	✅	❌	✅
Maze Generation	❌	❌	❌	✅
Visual Algorithm Comparison	❌	✅	❌	✅
Mobile Responsive UI	❌	❌	❌	✅
Stylish Modern UI & Animations	❌	❌	❌	✅ (GSAP-based)
Algorithm Insights (Strength/Weak)	❌	❌	❌	✅ (HTML Cards)
Code Extensibility (Open Source)	Limited	Moderate	Closed	✅ Fully Modular

Conlcusion

Key Learnings

- ***Algorithmic Differences in Practice***
  - While all four algorithms aim to find paths, their strategies, performance, and efficiency vary drastically under identical grid setups.
- ***Power of Visualization in Learning***
  - Visualizing algorithm behavior helps:
    - Instantly grasp traversal logic and node expansion.
    - Debug edge cases or inefficiencies.
    - Build a stronger intuition about when and why to use a specific algorithm.

Scope for Future Work

- ***Support Weighted Graphs -*** Simulate real-world maps and logistics.
- ***Heuristic Customization (for A*) -*** Allow users to select or tweak heuristic functions.
- ***Add Terrain & Obstacles with Costs -*** Simulate forests, water, traffic zones, etc.

Acknowledgements

Inspired by:

- *VisuAlgo.net* – for algorithm animation references
- *Wikipedia* – for algorithm theory
- *GeeksForGeeks* - for code references
- *JavaScript.info* - for JS DOM & animation tutorials

Team Contributions -

Adithya Subhash and Garv Sheth -

- Core logic for BFS, DFS, Dijkstra, A* in JS, and Animation control.

Advaith Moholkar and Debjit Ghorai -

- UI/UX design, CSS styling, layout integration, event handling.

Thank You

Thank You

Thank You