# AlgoSim--

**Submitted by**

Md. Asim Alam Chowdhury
BSSE-1611
Institute of Information Technology (IIT)
University of Dhaka

**Supervised by**

Dr. Md. Nurul Ahad Tawhid
Associate Professor
Institute of Information Technology (IIT)
University of Dhaka

# Project Overview

## The Mission

To develop a **modular and extensible** 2D grid-based algorithm simulation framework.

Designed to visually demonstrate the working principles of diverse computational algorithms.

## The Impact

Transforms **abstract algorithmic processes** into intuitive, observable behaviors.

Covers a broad spectrum from classical pathfinding to adaptive AI-driven optimization.

# Core Concept: The 2D Grid World

### Dynamic Environment

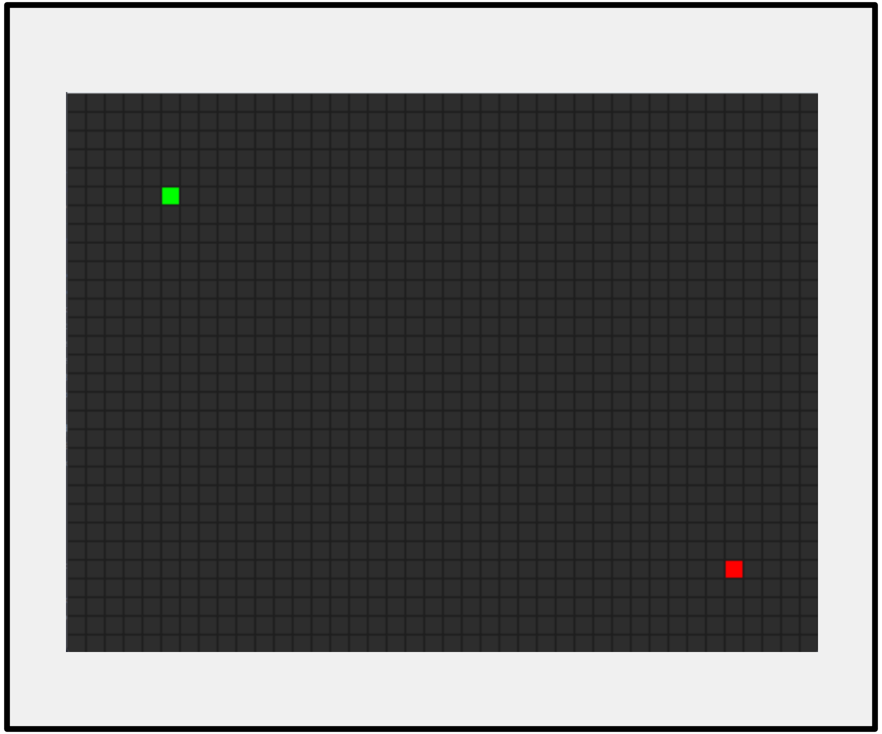
Algorithms operate in a real-time 2D grid where each cell functions as a discrete computational unit.

### State Visualization

Tracks intermediate states: exploration, frontier expansion, learning, and convergence.

### Interactive Elements

Cells represent environmental elements like obstacles, weights, or targets.



# Key Features of AlgoSim

## 01

### Modularity

A plug-and-play architecture that allows for the seamless addition or modification of algorithms within the framework.

## 02

### Extensibility

Designed to support a wide range of computational models, from simple pathfinding to complex AI-driven optimizations.

## 03

### Step-by-Step Visualization

Observe the algorithm's decision-making process at every stage, providing transparency into abstract computational logic.

## 04

### Interactive Environment

A dynamic grid system that supports real-time environmental changes, such as adding obstacles or moving targets during execution.

# Pathfinding & Search Algorithms

Focusing on foundational and heuristic search algorithms to visualize graph traversal and optimal path computation within the 2D grid environment.

01

### A* (A-Star) Search

Heuristic Optimal

02

### Dijkstra’s Algorithm

Shortest Path

03

### Dynamic A* (D*)

Incremental Search

04

### Bellman–Ford

Negative Weights

05

### Greedy Best-First

Heuristic Fast

MID-TERM

### Core Implementation

5 Algorithms

# Dijkstra’s Algorithm

**Initialization**

Sets all node distances to infinity except the start node, which is zero.

**Priority Queue**

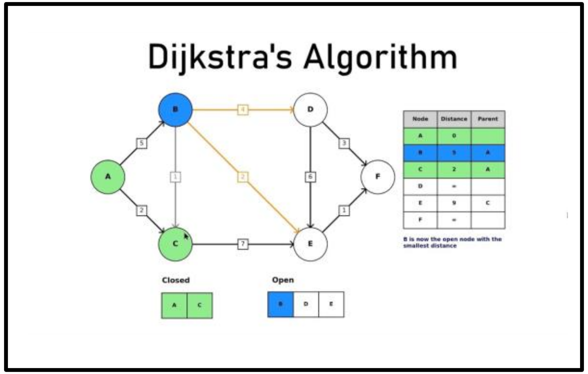
Always expands the node with the smallest known distance from the source.

**Relaxation**

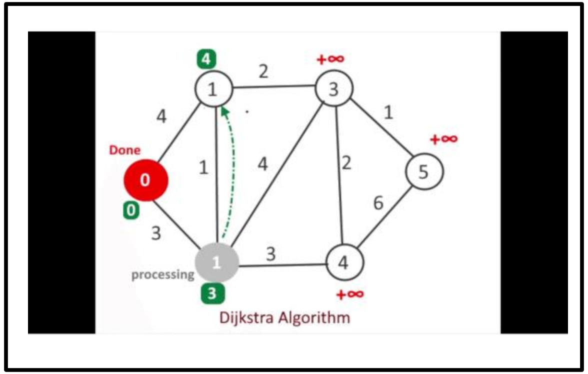
Updates neighbor distances if a shorter path is discovered through the current node.

**Convergence**

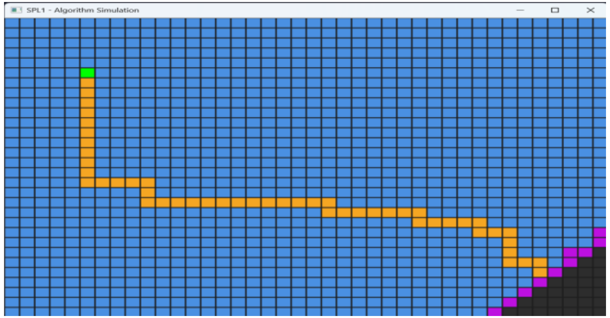
Repeats until all reachable nodes are visited or the target is reached.




Graph Representation




Node Processing State



# A* (A-Star) Search

**The Heuristic**

$$f(n) = g(n) + h(n)$$

**g(n):** Actual cost from start to node *n*.

**h(n):** Estimated cost from *n* to goal.

**Algorithm Process**

### Initialization

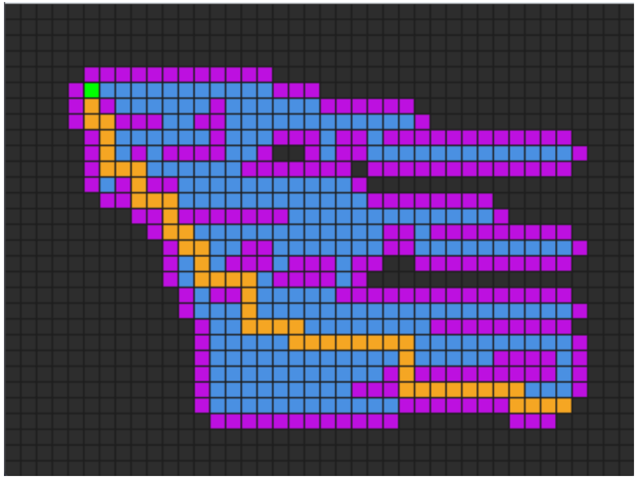
Assigns all nodes to infinity and identifies the start node.

### Frontier Expansion

Checks neighbors and pushes them into a priority queue based on f(n).

### Iteration

Pops the node with the lowest f(n) and repeats until the goal is reached.



# Dynamic A* (D*)

### Incremental Search

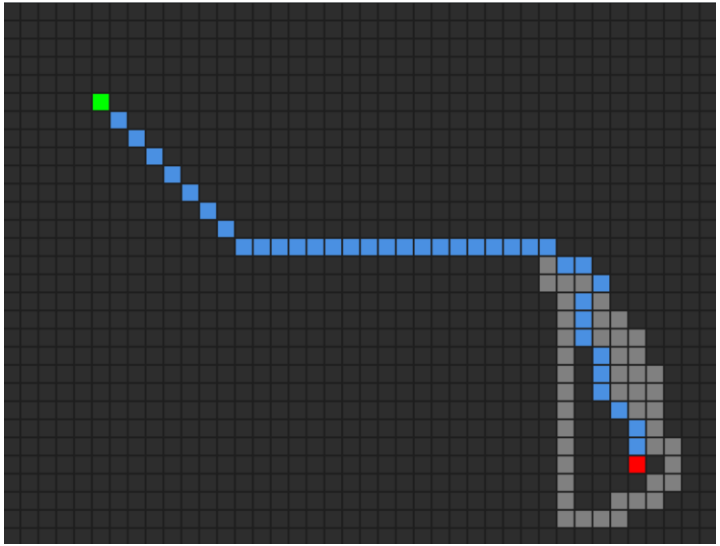
Designed for dynamic environments where map information changes or is partially known.

### Local Path Repair

When an obstacle is detected, D* repairs the path locally rather than recalculating from scratch.

### Efficiency

Only updates the affected parts of the grid, making it ideal for real-time robotic navigation.



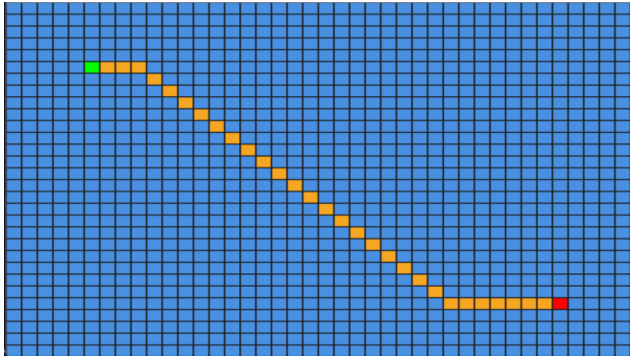
Grid-based Path Planning & Distance Calculation

# Bellman–Ford Algorithm

**Core Strength**

Unlike Dijkstra, Bellman-Ford can handle graphs with negative edge weights and detect negative cycles.

Computes shortest paths from a single source to all other vertices in a weighted digraph.



## Edge Relaxation

Repeatedly relaxes all edges |V|-1 times, where |V| is the number of vertices in the graph.

## Cycle Detection

Performs a final check: if any edge can still be relaxed, a negative weight cycle exists.

## Robustness

Slower than Dijkstra but essential for networks where negative costs or rewards are present.

**Time Complexity: O(V * E)**

# Greedy Best-First Search

### Heuristic Mechanism

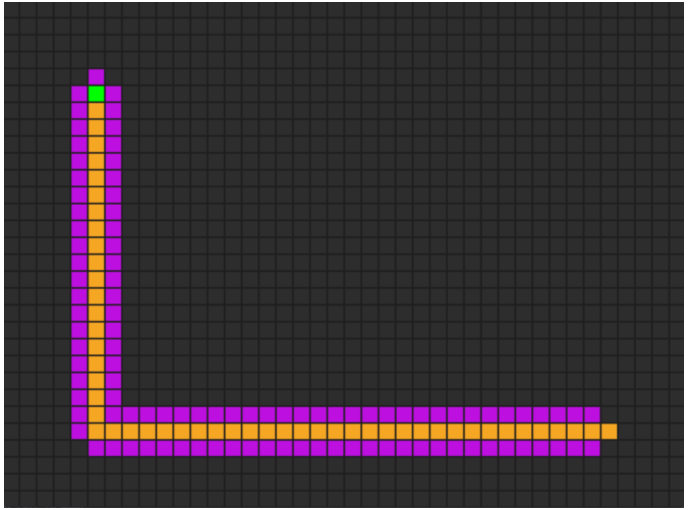
Prioritizes nodes that appear "closest" to the goal based solely on the heuristic function *h(n)*.

### Search Process

Neighbors are pushed into a priority queue; the most promising node is popped and expanded next.

### Efficiency

Extremely fast in simple environments as it aggressively moves toward the target.



**The Trade-off**

Fast execution but NOT guaranteed to find the shortest path as it ignores the cost to reach the node.

# Implementation Progress

## Framework Setup

Completed

2D Grid environment and basic UI components for algorithm control.

## Algorithm Integration

Completed

Core logic for A*, Dijkstra, D*, Bellman-Ford, and Greedy Search.

## Visualization Engine

In Progress

Real-time state updates and color-coded cell transitions.

# Thank You

**Project Impact**

AlgoSim provides a **powerful framework** for understanding complex algorithms through real-time visualization, enhancing both learning and debugging.

**Future Directions**

Expanding the **algorithm library** and optimizing performance for large-scale grids and complex AI-driven optimization models.