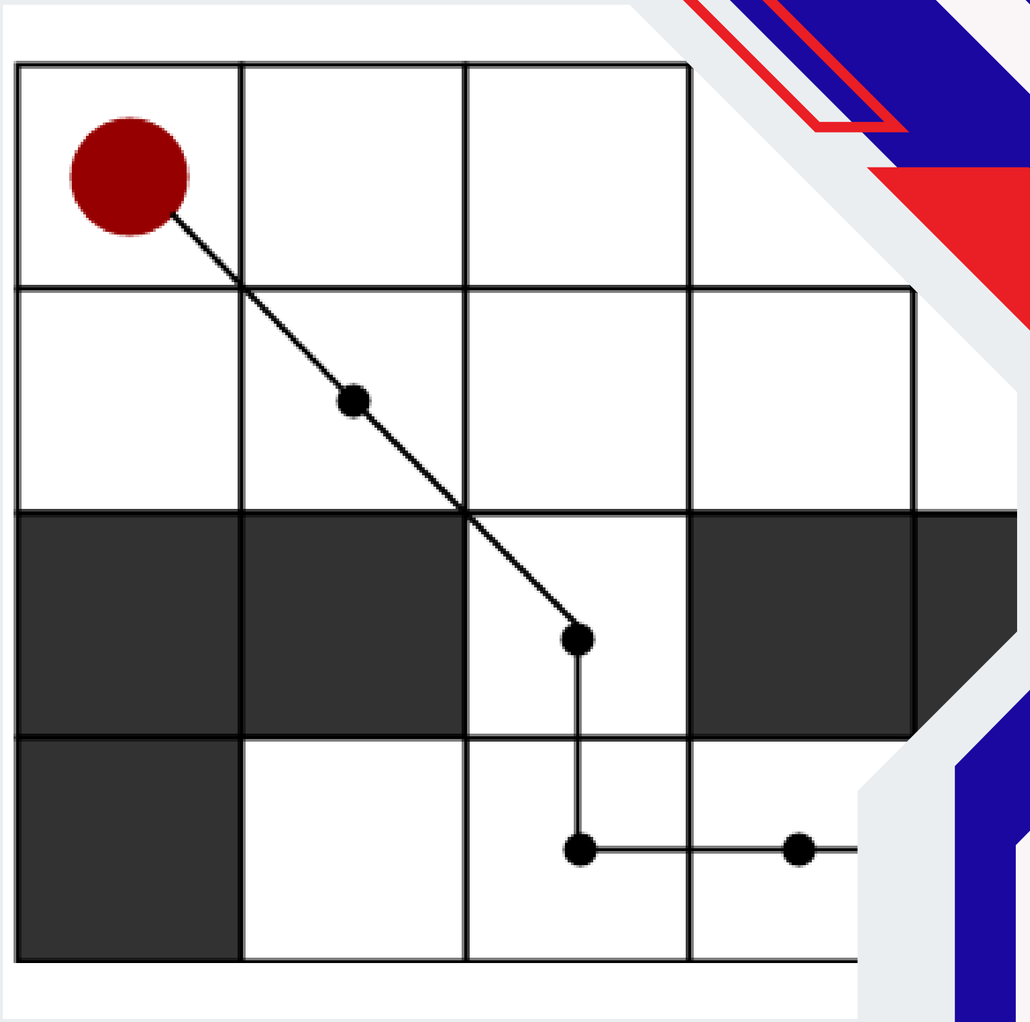
A* (A STAR) SEARCH ALGORITHM



INTRODUCTION TO AI | AI6310

JOHN PAUL M. MANUEL | ACLC COLLEGE OF TAYTAY

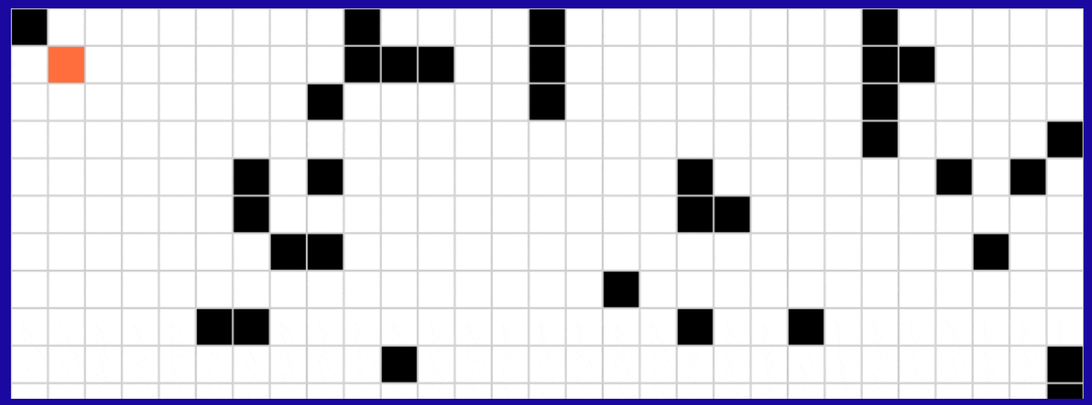
ACLC COLLEGE OF TAYTAY
(A Member of AMA Education System)

VERSION 01-250211-2425

WHAT IS A* ALGORITHM?

The A* algorithm is an informed search algorithm, meaning it leverages a heuristic function to guide its search towards the goal. This heuristic function estimates the cost of reaching the goal from a given node, allowing the algorithm to prioritize promising paths and avoid exploring unnecessary ones.

The A* algorithm is a powerful and widely used graph traversal and path finding algorithm. It finds the shortest path between a starting node and a goal node in a weighted graph.



A* ALGORITHM

HOW DOES THE A* ALGORITHM WORK?

The A* algorithm combines the best aspects of two other algorithms:

1. **Dijkstra's Algorithm:** This algorithm finds the shortest path to all nodes from a single source node.
2. **Greedy Best-First Search:** This algorithm explores the node that appears to be closest to the goal, based on a heuristic function.

Imagine you're trying to find the shortest route between two cities on a map. While Dijkstra's algorithm would explore in all directions and Best-First Search might head straight toward the destination (potentially missing shortcuts), A* does something cleverer. It considers both:

- The distance already traveled from the start
- A smart estimate of the remaining distance to the goal

A* ALGORITHM

KEY COMPONENTS OF A* ALGORITHM

To understand A* algorithm, you need to be familiar with these fundamental concepts:

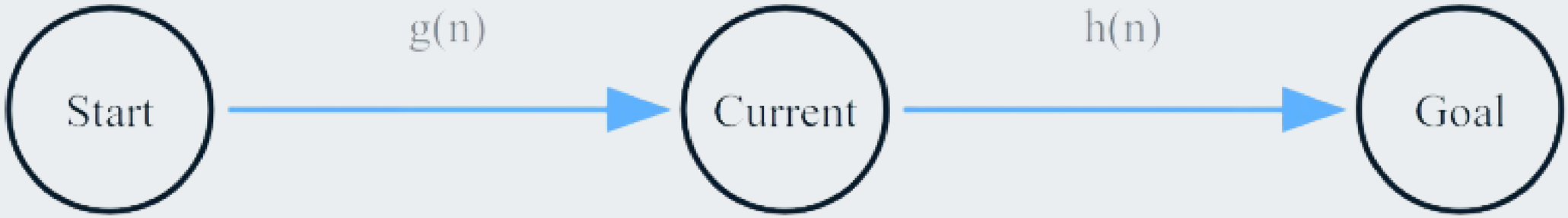
- **Nodes**: Points in your graph (like intersections on a map)
- **Edges**: Connections between nodes (like roads connecting intersections)
- **Path Cost**: The actual cost of moving from one node to another
- **Heuristic**: An estimated cost from any node to the goal
- **Search Space**: The collection of all possible paths to explore

A* ALGORITHM

KEY CONCEPTS IN A* SEARCH

The A* algorithm's efficiency comes from its smart evaluation of paths using three key components: g(n), h(n), and f(n). These components work together to guide the search process toward the most promising paths.

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



A* ALGORITHM

KEY CONCEPTS IN A* SEARCH

Path cost g(n)

The path cost function g(n) represents the exact, known distance from the initial starting node to the current position in our search. Unlike estimated values, this cost is precise and calculated by adding up all the individual edge weights that have been traversed along our chosen path.

Mathematically, for a path through nodes n0(start node) to nk (current node), we can express g(n) as:

$$g(n_k) = \sum_{i=0}^{k-1} w(n_i, n_{i+1})$$

w(ni,ni+1) represents the weight of the edge connecting node ni to node ni+1.

A* ALGORITHM

KEY CONCEPTS IN A* SEARCH

Heuristic function h(n)

The heuristic function h(n) provides an estimated cost from the current node to the goal node, acting as the algorithm's "informed guess" about the remaining path. Mathematically, for any given node n, the heuristic estimate must satisfy the condition $h(n) \leq h^*(n)$, where $h^*(n)$ is the actual cost to the goal, making it admissible by never overestimating the true cost.

In grid-based or map-like problems, common heuristic functions include the Manhattan distance and Euclidean distance. For coordinates (x1,y1) of the current node and (x2,y2) of the goal node, these distances are calculated as:

Manhattan distance

$$h(n) = |x_1 - x_2| + |y_1 - y_2|$$

Euclidean distance

$$h(n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

A* ALGORITHM

KEY CONCEPTS IN A* SEARCH

Total estimated cost f(n)

The total estimated cost f(n) is the cornerstone of A* algorithm's decision-making process, combining both the actual path cost and the heuristic estimate to evaluate each node's potential. For any node n, this cost is calculated as:

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

Where:

- g(n) represents the actual cost from the start to the current node,
- h(n) represents the estimated cost from the current node to the goal.

The algorithm uses this combined value to strategically choose which node to explore next, always selecting the node with the lowest f(n) value from the open list, thus ensuring an optimal balance between known costs and estimated remaining distances.

A* ALGORITHM

MANAGING NODE LISTS

The A* algorithm maintains two essential lists

Open list:

- Contains nodes that need to be evaluated
- Sorted by f(n) value (lowest first)
- New nodes are added as they're discovered

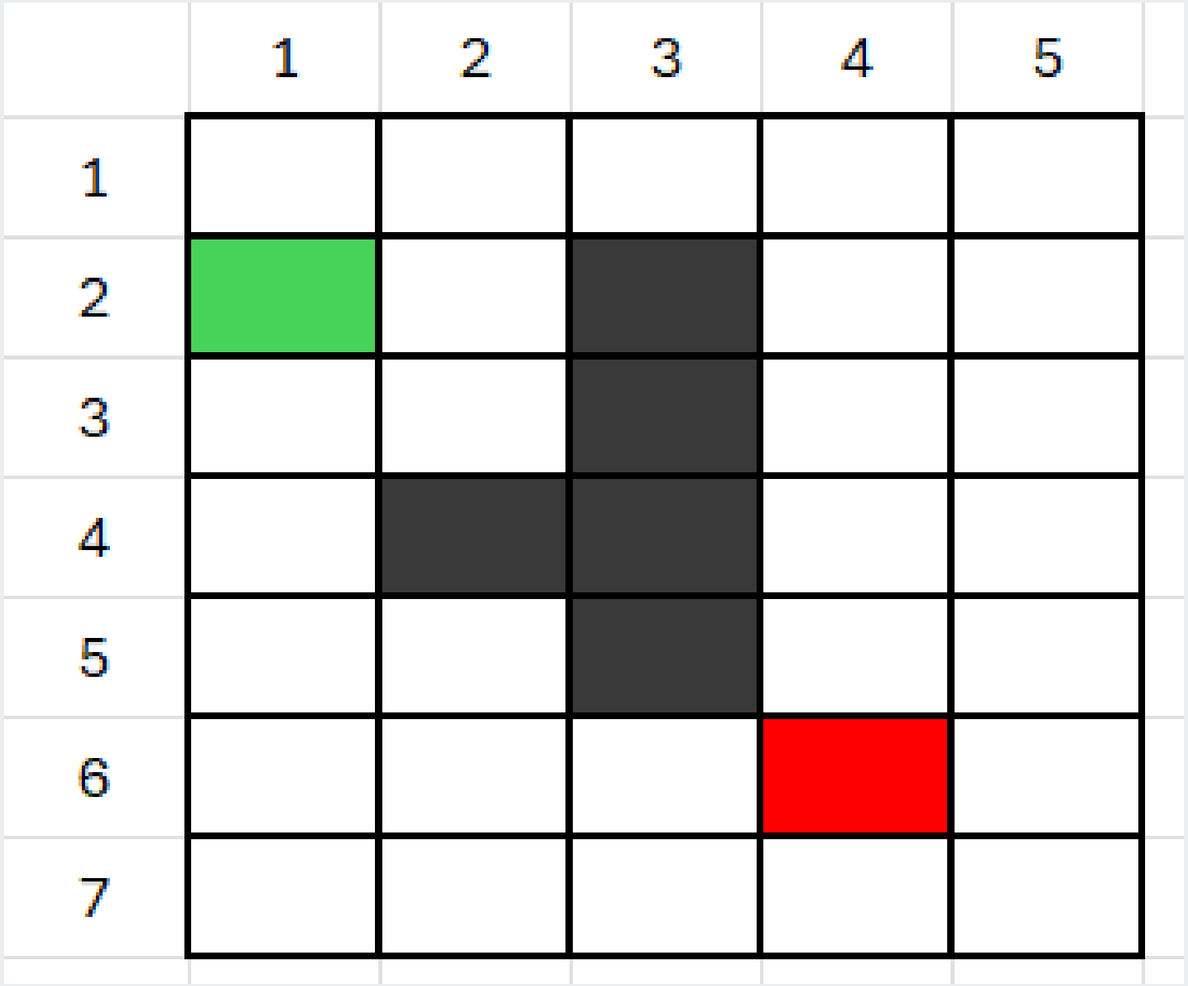
Closed list:

- Contains already evaluated nodes
- Helps avoid re-evaluating nodes
- Used to reconstruct the final path

The algorithm continually selects the node with the lowest f(n) value from the open list, evaluates it, and moves it to the closed list until it reaches the goal node or determines no path exists.

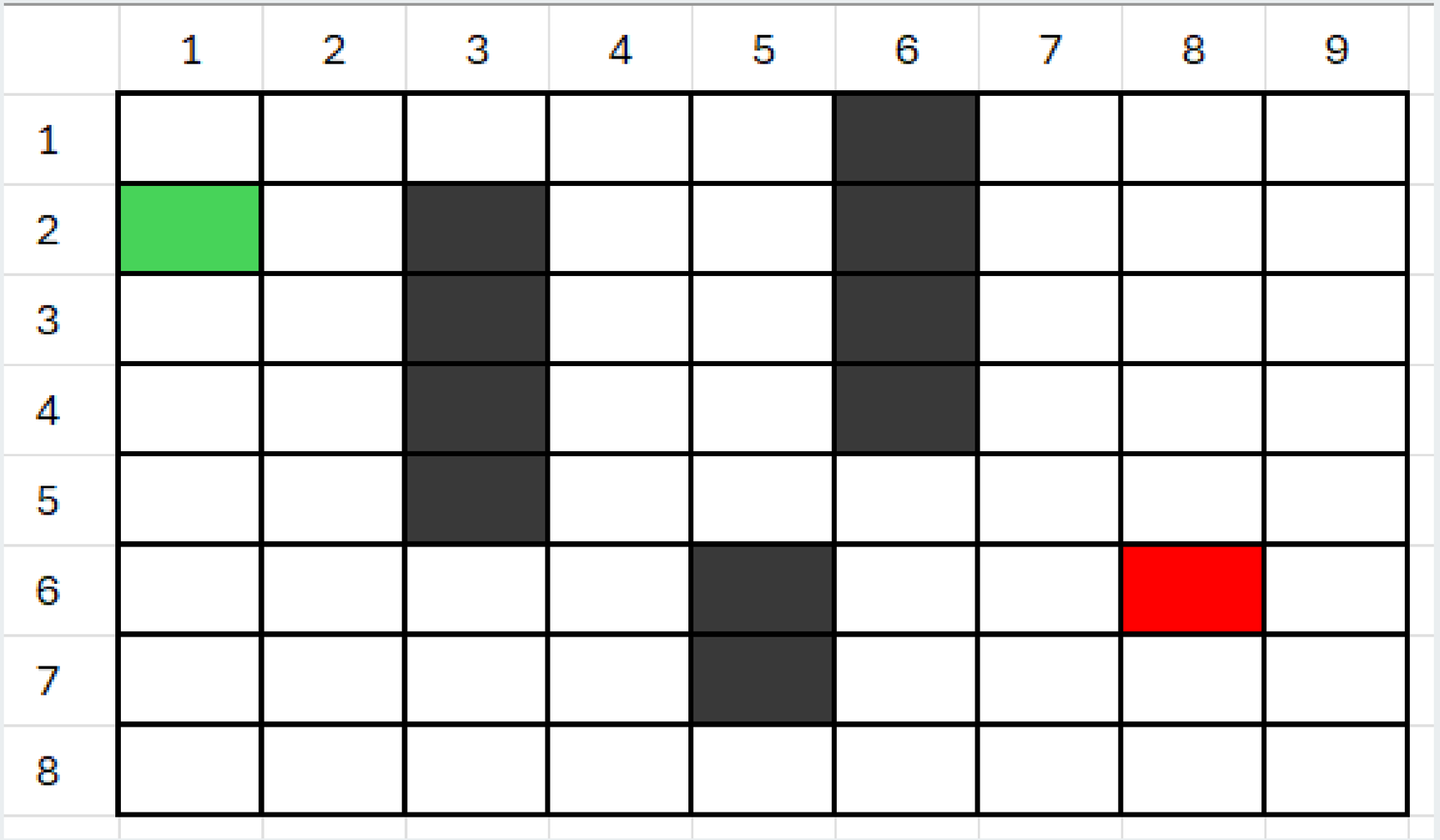
A* ALGORITHM

EXAMPLE 1



A* ALGORITHM

EXAMPLE 2

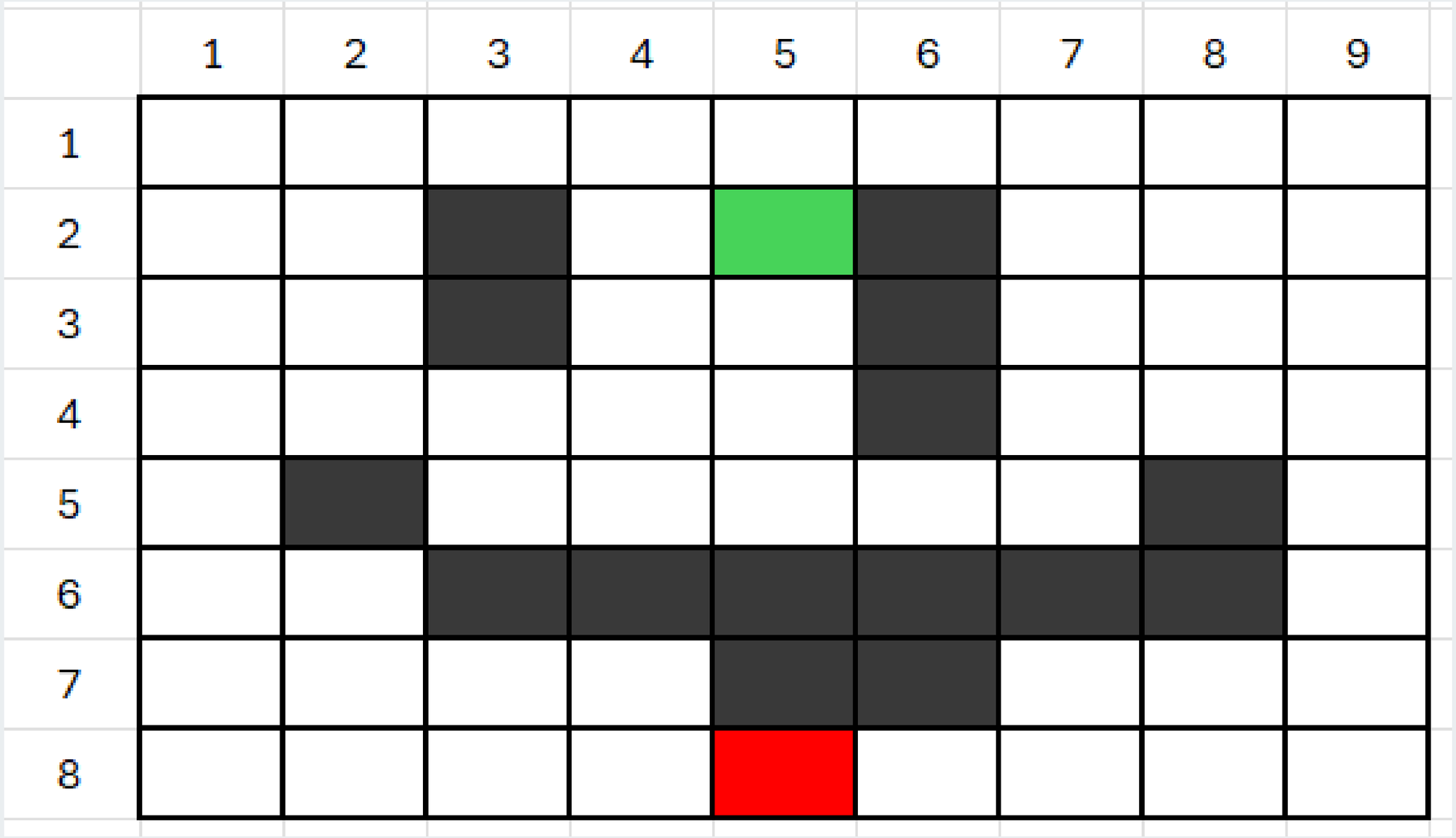


READY FOR ACTIVITY?

A OR B

A* ALGORITHM

ACTIVITY



A* ALGORITHM

ACTIVITY