MAZE SOLVER

DONE BY :

KENISHA SURANA -RA2211027010078
PRATYUSH KUMAR SINGH-
RA2211027010088
AKSHAT SHARMA - RA2211027010121

CONTENTS

1 Problem Statement & Introduction

2 Algorithm Design Technique

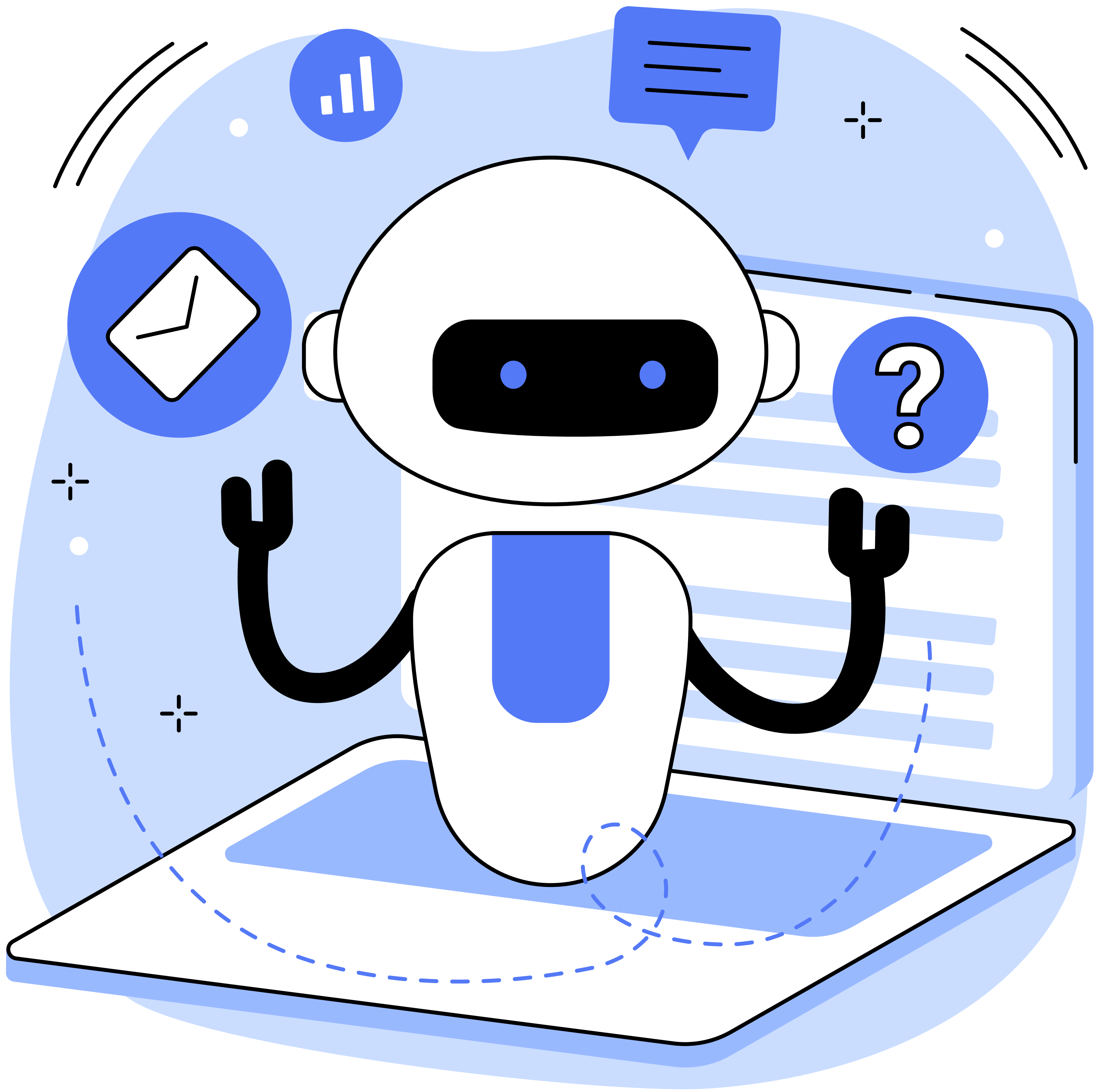
3 Time Complexity

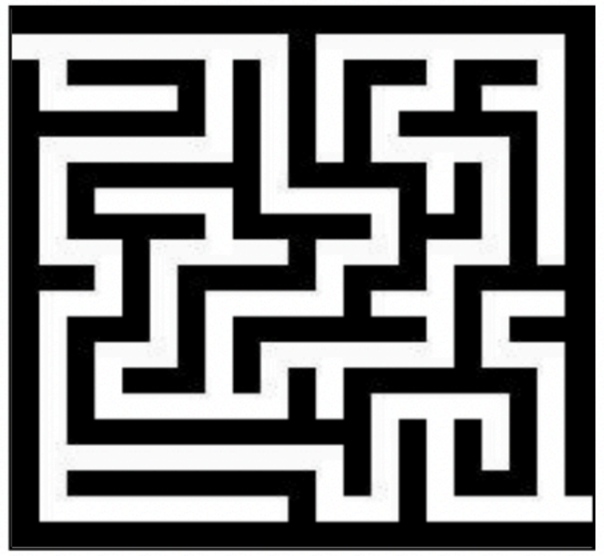
4 Comparative Study of Algorithm

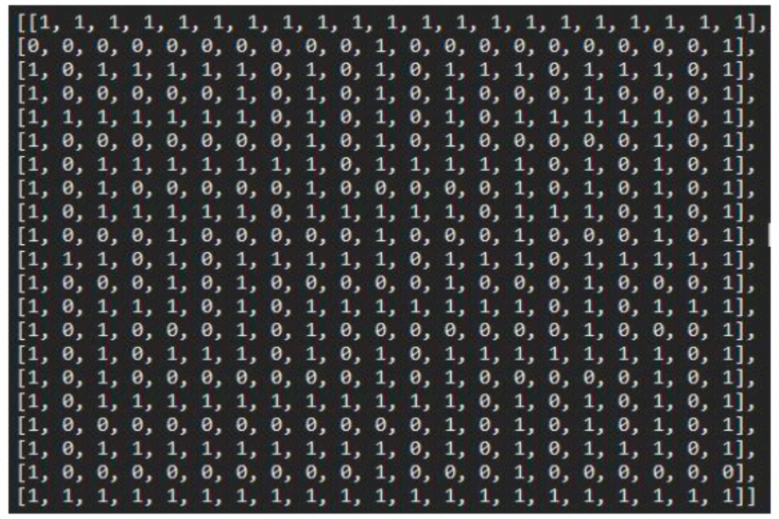
5 Result

PROBLEM STATEMENT

This program takes a gif of a maze as input and outputs a solution to the maze.



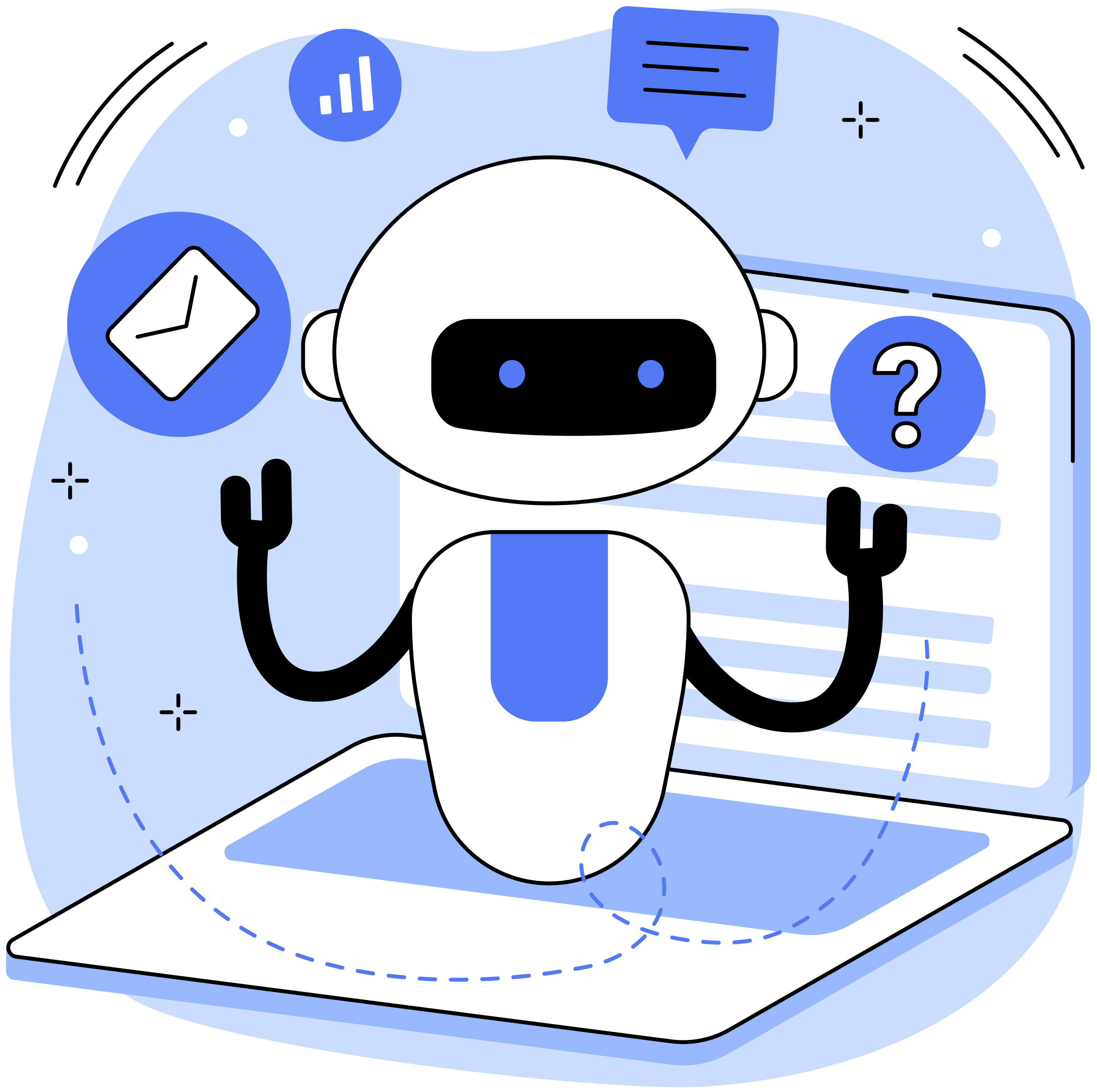




```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
[1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1],
[1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1],
[1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1],
[1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1],
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
[1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1],
[1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1],
[1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1],
[1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
[1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1],
[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

INTRODUCTION TO TOPIC

The program converts the gif into a matrix where each cell represents a 5x5 area of the original gif. It then uses a breadth-first search algorithm to find the shortest path from the starting cell to the ending cell and displays the solution as a gif.



ALGORITHM DESIGN TECHNIQUE

The maze solver using BFS aims to create an algorithm finding the shortest path in a maze. It takes a maze as input, applies BFS to explore it until reaching the endpoint, and outputs the shortest path and steps taken. This has practical applications in robotics, gaming, and navigation, allowing efficient obstacle navigation for robots, guiding characters in gaming levels, and determining the shortest route on maps.

ALGORITHM

STEP-1

Read the input maze and identify the start and end points

STEP-2

Create an empty queue and add the start point to it.

STEP-3

Initialize a visited set to keep track of already visited points and mark the start point as visited.

STEP-4

Loop until the queue is empty:
a. Dequeue the front point from the queue.

STEP-4

b. If the dequeued point is the end point, then the algorithm has found the shortest path.

STEP-4

c. Otherwise, enqueue all adjacent unvisited points to the queue and mark them as visited.

STEP-5

If the end point is reached, trace back the path from the end point to the start point by following the parents of each point.

STEP-6

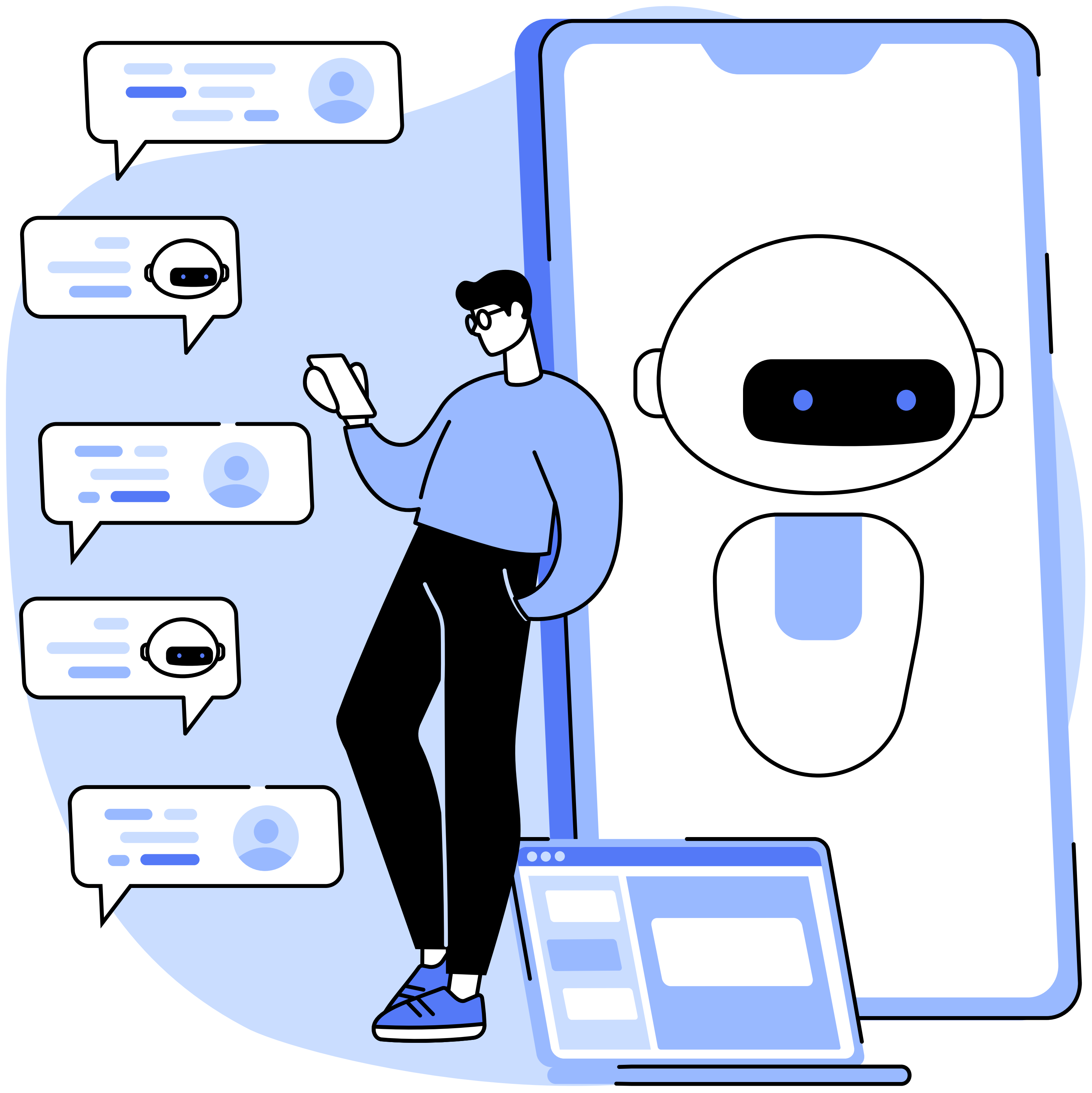
Output the shortest path from the start point to the end point along with the steps taken to reach the end point.

TIME COMPLEXITY

Breadth-First Search

Time Complexity: $O(V + E)$

BFS guarantees the shortest path in unweighted graphs, making it a good choice for maze solving if the maze is unweighted and finding the shortest path is crucial.



COMPARISON WITH OTHER ALGORITHM

Recursive Division

Time Complexity:
O(n^2)

Recursive division is typically used for maze generation rather than solving, but it can be adapted to solving by recursively dividing until the solution path is found.

Depth-First Search

Time Complexity:
O(V + E)

DFS may not find the shortest path and can get stuck in infinite loops if not implemented with care. However, it requires less memory compared to BFS due to its depth-first nature

Dijkstra's Algorithm

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

Dijkstra's algorithm guarantees the shortest path even in weighted graphs, making it suitable for maze solving with weighted paths. However, it's slower than BFS for unweighted graphs.

THANK YOU !!