# ARTIFICIAL INTELLIGENCE LABORATORY

MINI PROJECT REPORT

|  |  |
| --- | --- |
| REGISTER NUMBER | 2117240070061 |
| NAME | DHARISANA R |
| PROJECT TITLE | MAZE ESCAPE GAME USING BFS |
| DATE OF SUBMISSION | 30/10/2025 |
| FACULTY IN-CHARGE | Mrs. M. Divya |

Signature of Faculty In-charge

# Maze Escape Game using BFS

## INTRODUCTION

Artificial Intelligence enables computers to solve problems that usually require human intelligence. In this project, we apply the Breadth-First Search (BFS) algorithm to help a player escape a maze. The algorithm systematically explores all possible paths from the start position until it reaches the exit, ensuring the shortest path is found efficiently.

## PROBLEM STATEMENT

To develop a maze-solving game where an AI agent uses the Breadth-First Search algorithm to find the shortest escape path from the maze.

## GOAL

To demonstrate the use of BFS in solving maze navigation problems and visualize the shortest path from the start to the goal node.

## THEORETICAL BACKGROUND

The Breadth-First Search (BFS) algorithm is a graph traversal technique that explores nodes level by level. It uses a queue data structure to explore all neighboring nodes before moving to the next level. This guarantees the shortest path in an unweighted graph like a maze.

## ALGORITHM EXPLANATION WITH EXAMPLE

Step 1: Represent the maze as a 2D grid where walls and paths are defined.  
Step 2: Initialize a queue with the start position.  
Step 3: Explore all neighboring cells (up, down, left, right) and mark visited ones.  
Step 4: Continue exploration until the goal (exit) is reached.  
Step 5: Backtrack to reconstruct the shortest path.

## IMPLEMENTATION AND CODE

# Maze Escape Game using BFS with Visualization

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from collections import deque

# Define the BFS function

def bfs\_with\_visualization(maze, start, goal):

    rows, cols = len(maze), len(maze[0])

    visited = [[False]\*cols for \_ in range(rows)]

    parent = {}

    queue = deque([start])

    visited[start[0]][start[1]] = True

    directions = [(-1,0), (1,0), (0,-1), (0,1)]

    frames = []  # store exploration steps for animation

    while queue:

        current = queue.popleft()

        frames.append([row[:] for row in visited])  # snapshot

        if current == goal:

            break

        for d in directions:

            new\_row, new\_col = current[0] + d[0], current[1] + d[1]

            if (0 <= new\_row < rows and 0 <= new\_col < cols and

                maze[new\_row][new\_col] == 0 and not visited[new\_row][new\_col]):

                queue.append((new\_row, new\_col))

                visited[new\_row][new\_col] = True

                parent[(new\_row, new\_col)] = current

    # Reconstruct shortest path

    path = []

    node = goal

    while node != start:

        path.append(node)

        node = parent.get(node)

        if node is None:

            print("No path found!")

            return frames, []

    path.append(start)

    path.reverse()

    return frames, path

# Draw the maze and animate BFS exploration

def visualize\_bfs(maze, frames, path):

    fig, ax = plt.subplots()

    maze\_display = [[1 if cell == 1 else 0 for cell in row] for row in maze]

    img = ax.imshow(maze\_display, cmap='gray')

    def update(frame):

        display = [[1 if cell == 1 else 0 for cell in row] for row in maze]

        for r in range(len(frame)):

            for c in range(len(frame[0])):

                if frame[r][c]:

                    display[r][c] = 0.6  # explored area

        img.set\_data(display)

        return [img]

    ani = animation.FuncAnimation(fig, update, frames=frames, repeat=False, interval=200)

    plt.title("BFS Maze Exploration")

    plt.show()

    # Show final path

    for r, c in path:

        maze\_display[r][c] = 0.3

    plt.imshow(maze\_display, cmap='gray')

    plt.title("Shortest Path Found by BFS")

    plt.show()

# Example Maze (0 = path, 1 = wall)

maze = [

    [0, 0, 1, 0, 0],

    [0, 0, 1, 0, 1],

    [1, 0, 0, 0, 0],

    [0, 1, 0, 1, 0],

    [0, 0, 0, 1, 0]

]

start = (0, 0)

goal = (4, 4)

# Run BFS and visualize

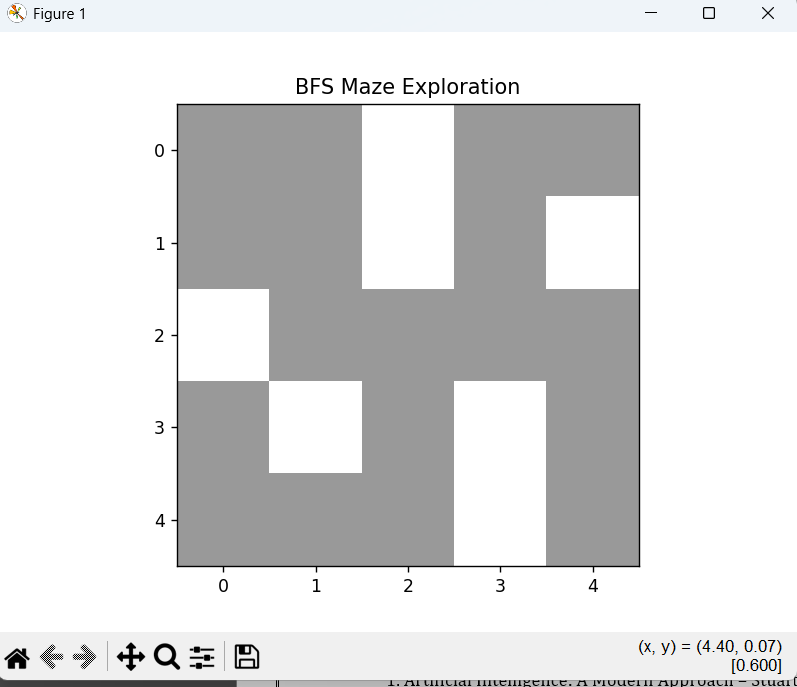
frames, path = bfs\_with\_visualization(maze, start, goal)

print("Shortest Path:", path)

visualize\_bfs(maze, frames, path)

## OUTPUT

Shortest Path: [(0, 0), (1, 0), (1, 1), (2, 1), (2, 2), (2, 3), (2, 4), (3, 4), (4, 4)]

.

## RESULTS AND FUTURE ENHANCEMENT

The BFS algorithm successfully finds the shortest path through the maze. Future improvements can include adding user input for maze generation, visual animation of pathfinding, or comparison with DFS and A\* algorithms.

GitHub Link of the project and report: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## REFERENCES

* “Optimization of Breadth-First Search Algorithm for Path Solutions in Maze Games”

— Bonifacius Vicky Indriyono & Widyatmoko.

Link: [https://www.researchgate.net/publication/362385104\_Optimization\_of\_Breadth-First\_Search\_Algorithm\_for\_Path\_Solutions\_in\_Mazyin\_Games](https://www.researchgate.net/publication/362385104_Optimization_of_Breadth-First_Search_Algorithm_for_Path_Solutions_in_Mazyin_Games?utm_source=chatgpt.com) [ResearchGate](https://www.researchgate.net/publication/362385104_Optimization_of_Breadth-First_Search_Algorithm_for_Path_Solutions_in_Mazyin_Games?utm_source=chatgpt.com)

* “A Comparative Study of DFS, BFS, and A\* Search Algorithms” — Badr Elkari et al., 2024.
* Link: [https://www.iapress.org/index.php/soic/article/view/1939](https://www.iapress.org/index.php/soic/article/view/1939?utm_source=chatgpt.com) [iapress.org](https://www.iapress.org/index.php/soic/article/view/1939?utm_source=chatgpt.com)
* “Introduction to Pathfinding with Mazes and Breadth-first Search”

— Tom Cant (blog post).

Link: [https://tomcant.dev/posts/2021/09/introduction-to-pathfinding-with-mazes-and-breadth-first-search/](https://tomcant.dev/posts/2021/09/introduction-to-pathfinding-with-mazes-and-breadth-first-search/?utm_source=chatgpt.com) [tomcant.dev](https://tomcant.dev/posts/2021/09/introduction-to-pathfinding-with-mazes-and-breadth-first-search/?utm_source=chatgpt.com)

* “Solve Maze using Breadth-First Search (BFS) Algorithm” —

Luthfis Auqi (Medium article).

Link: [https://medium.com/%40luthfisauqi17\_68455/artificial-intelligence-search-problem-solve-maze-using-breadth-first-search-bfs-algorithm-255139c6e1a3](https://medium.com/%40luthfisauqi17_68455/artificial-intelligence-search-problem-solve-maze-using-breadth-first-search-bfs-algorithm-255139c6e1a3?utm_source=chatgpt.com) [Medium](https://medium.com/%40luthfisauqi17_68455/artificial-intelligence-search-problem-solve-maze-using-breadth-first-search-bfs-algorithm-255139c6e1a3?utm_source=chatgpt.com)

* “Pathfinding in Strategy Games and Maze Solving Using A\*” — SCIRP article

Link: [https://www.scirp.org/journal/paperinformation?paperid=70460](https://www.scirp.org/journal/paperinformation?paperid=70460&utm_source=chatgpt.com) [scirp.org](https://www.scirp.org/journal/paperinformation?paperid=70460&utm_source=chatgpt.com)