# Rat in maze solving problem

## Introduction:

The Rat Maze Solver program is designed to find a path from the upper left corner to the lower right corner of a maze represented as a binary matrix. The maze-solving algorithm is implemented using multithreading and employs a GUI for user interaction.

## Problem Description

#### Maze Representation

* The maze is represented as an 𝑁 × 𝑁 binary matrix.
* 0 indicates a dead end, and 1 indicates a traversable path.
* The rat starts at the upper-left corner (0,0) and must reach the lower-right corner (N-1,N-1).
* The rat can only move forward or down.

#### Multithreading

* The program uses multithreading to explore multiple paths simultaneously.
* Each possible direction creates a new thread.
* The number of threads is limited based on the number of processors.

#### Thread Management

The program uses a semaphore to limit the number of threads based on the number of processors. This is done to ensure correct usage of threads and synchronization mechanisms

GUI

The program uses a graphical user interface (GUI) to interact with the user. The GUI includes a text field to enter the dimensions of the maze, a button to create the maze, a button to solve the maze, and a grid to display the maze. The grid cells are buttons that the user can click to specify dead blocks

## Program Structure

#### i.RatMazeGUI Class

* Attributes:
  + N: Maze dimension.
  + cells: 2D array of buttons representing the maze cells.
  + maze: 2D array representing the maze.
  + semaphore: Semaphore for limiting the number of threads.
* Methods:
  + RatMazeGUI(): Constructor initializing the GUI components.
  + CellActionListener: ActionListener for updating the maze on cell clicks.
  + main: Entry point of the program.

#### ii.RatMazeSolver Class

* Attributes:
  + maze: 2D array representing the maze.
  + N: Maze dimension.
  + cells: 2D array of buttons representing the maze cells.
  + pathFound: Boolean indicating whether a path is found.
  + semaphore: Semaphore for limiting the number of threads.
* Methods:
  + RatMazeSolver: Constructor initializing the solver with maze details.
  + run: Overrides the run method for thread execution.
  + solveMaze: Recursive method to explore possible paths.
  + updateCellColor: Updates the color of a cell in the GUI.

## Program Execution

* User enters maze dimensions through the GUI.
* Clicking "Create Maze" generates a grid of buttons representing the maze cells.
* User specifies dead blocks by clicking on cells.
* Clicking "Solve" initiates the maze-solving process using multithreading.
* The algorithm explores paths concurrently and updates the GUI in real-time.
* Threads are limited based on the number of processors.

## Conclusion

he Rat Maze Solver program provides an interactive and multithreaded solution to finding a path through a maze. It effectively utilizes synchronization mechanisms, multithreading, and a graphical user interface for an engaging user experience. The program adheres to the specified problem requirements and provides real-time updates during maze exploration.

#### Thank you