Beihang University
School of General Engineering

Beijing, May 18th, 2021
Lin Wei and Yida Ding

Data Management and Artificial Intelligence
Lab Class 11

Task 1 ***Maze*** *(70 minutes)*

As shown in Figure 20, given a square board with many grids, a worm wants to start from the origin cell (green) and go to the destination cell (orange). However, there are walls in some cells (grey) that cannot be passed by the worm. Please help the worm to get to the destination cell by solving the following tasks:

1. Download the python file 'DMAI_Lab11_file.py' and try to understand the given code. *MazeGenerator* is used to set up the maze and return the data (2-D nested list) for initial state and goal state. Note that 0 represents empty cell, 1 represents the wall, and 2 represents the current position of the worm. *MazeVisualizer* is used to visualize a state given the data for a state. Then you need to implement the *State* class in the following.

2. Implement functions for goal state judgement, legal action identification, state transition and etc. Keep in mind that your code should work for any sizes of square maze.

3. Implement two heuristic functions: Manhatten distance and Euclidean distance. Which one is better for solving this problem?

4. Implement three uninformed searching methods: BFS, DFS and UCS (Uniform-Cost Search) to solve the initial configuration obtained by *MazeGenerator*.

5. Implement two informed searching methods: GBFS (Greedy best first search), and A^* to solve the initial configuration obtained by *MazeGenerator*.

6. Implement a member function in class *MazeVisualizer* to visualize the path of A^*. A possible outcome is visualized in Figure 21.

7. Try to compare the number of expanded nodes and runtime for different sizes of maze and different searching methods. How efficient is A^* compared to the other searching methods?

Task 2 ***8-Queen*** *(Once you have finished Task 1)*

Try to solve the 8-queen problem from the lecture, where you need to place eight queens on a chess board, such that neither pair of queens are attacking each other. You can model it as a game with eight initial queens on the board at random locations and then an action in the game corresponds to moving one queen to a new location, while minimizing the number of attacking pairs.

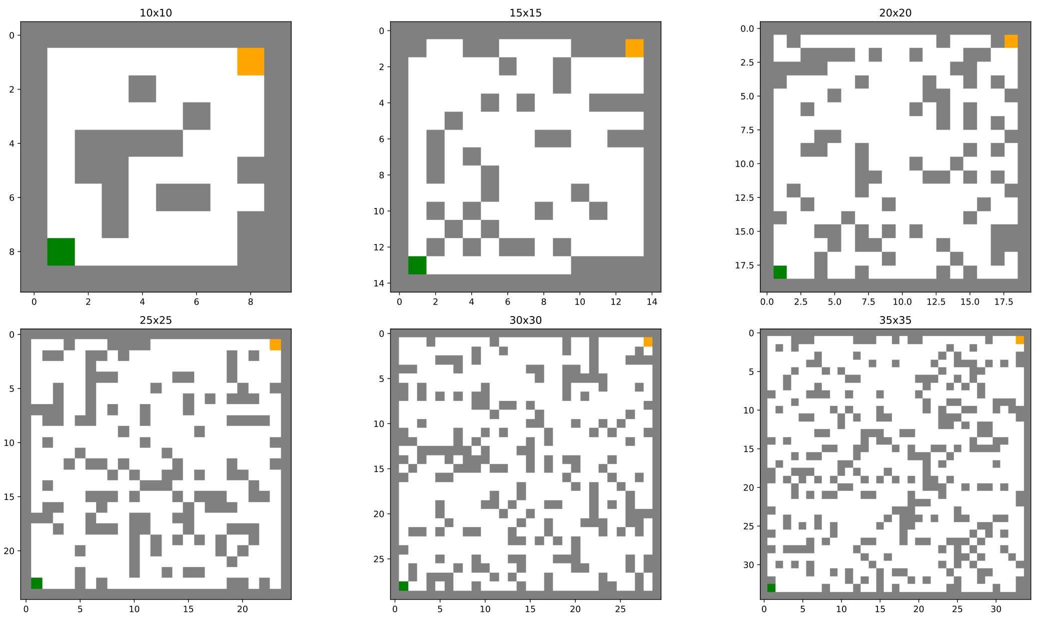

Figure 20: Origin cell and destination cell in six mazes

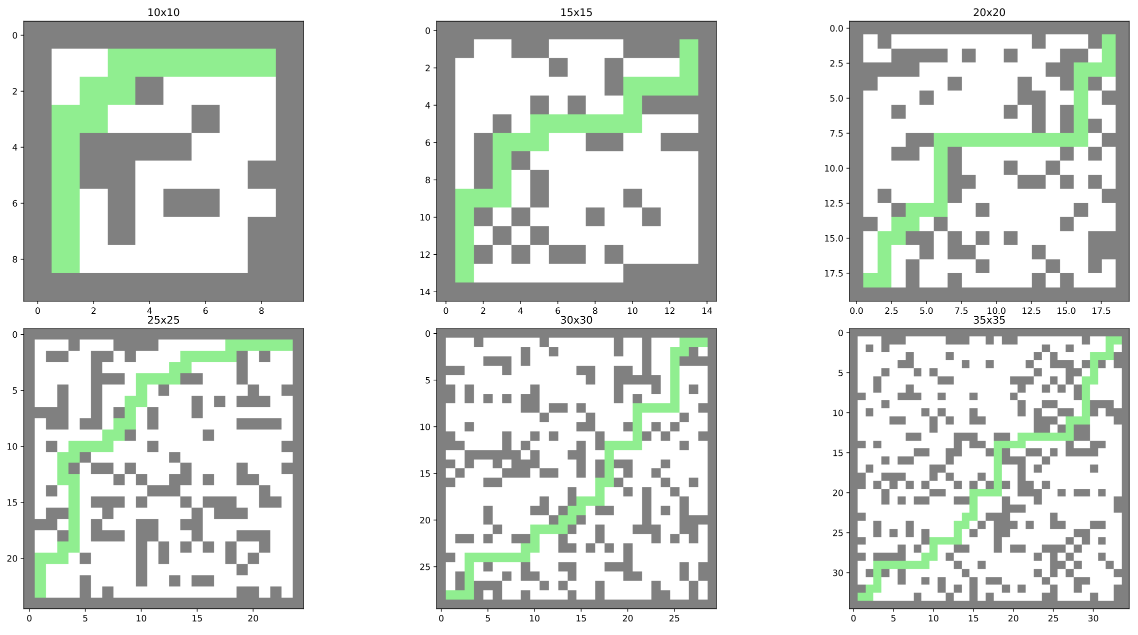

Figure 21: The paths obtained by A^* algorithm