Hacktrick 2023

Hackathon Description

Welcome to Hacktrick!

In this hackathon, you will be tasked with designing and implementing an AI agent that can navigate through a maze to rescue trapped children and safely exit the maze while carrying them.

The agent should be able to understand the maze's layout, identify the location of the trapped children, plan the optimal path to rescue them. Once the agent has rescued the children, it must also be able to carry them to safety in the shortest time while avoiding obstacles.

There's a catch! Rescuing the children is not simply reaching the location and picking them up. Your agent will need to solve a security riddle for each child in order to rescue them.

We will be evaluating your agent based on the number of children rescued, number of actions taken in the allotted time and efficiency when solving the riddles. More technical in-depth technical details are provided in the upcoming sections.

Finally, it is worth noting that there are no constraints on how you implement your agent. We will be providing you with assistance and some tips & tricks, but by no means do we require you to use a specific method.

Maze Environment

The environment consists of two main components, which are the action space and observation space.

Action Space

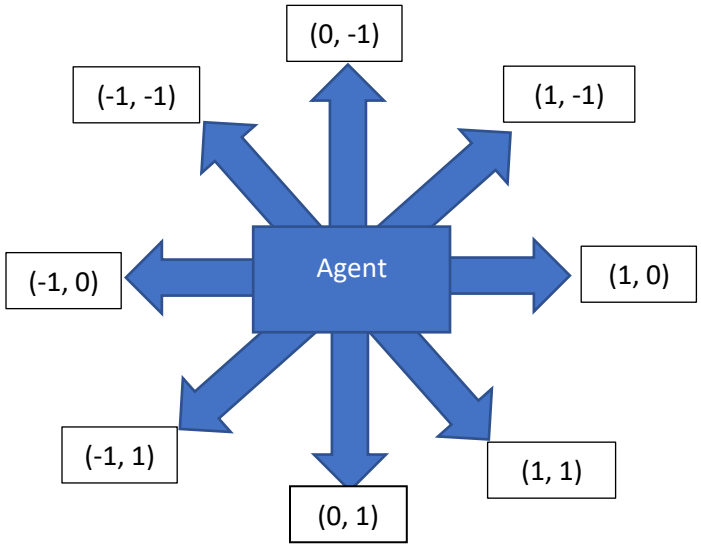
The action space allows the agent to move in only four directions which are up, down, right and left ("N", "S", "E", "W"). Note that these directions are global and are not relative to the direction that the agent is facing.

Observation Space

The observation space is a representation of the agent's state for each timestep in the environment. It is defined by the following three attributes:

- (X, Y): Represents the coordinate/position of the agent. The initial point is top left (0,0)
- [D1, D2, D3 …. Dn]: This is an array representing the relative Manhattan distance from the agent's location to all rescue items in the maze.
- [(X1, Y1), (X2, Y2), (X3, Y3) …. (Xn, Yn)]: This is an array of tuples representing the relative directions from the agent's location to all rescue items in the maze. There are 3 possible values for X and Y [-1, 0 ,1]. The following figure represents the rescue items directions relative to the

agent's position. Note that if the direction is (0,0) this means that the agent is on top of the rescue item.



Important Notes:

- Since the distance is represented using the Manhattan distance, it will always be a positive number.
- If a riddle has been attempted, the associated rescue item will have a distance of -1 and direction (0,0) regardless of the agent's position. If the attempted solution is wrong, you will not be able to solve it again.

Rescue Items and Riddles State

The state will also contain information regarding the state of the riddles and rescue items. In each step, the environment will send the current number of rescued items. If the agent reaches a rescue item, the environment will also send the associated riddle type and question. Riddle type can be one of the following four types: ["cipher", "pcap", "server", "captcha"]. For more information about the riddle types, please refer to the riddles documentation.

Solving a maze

The maze is considered solved when the agent reaches the exit which is located at (9,9). This needs to be done before the timeout set by the environment or reaching the maximum number of steps. More details regarding the timeout conditions can be found in the server/API documentation.

Solving a riddle

In order to solve a riddle, the agent must first parse the type of riddle from the received state. Then, the agent should call the dedicated function that you have implemented to solve this type of riddle and respond with the solution.

If your agent's solution matches the reference solution used for evaluation, the riddle will be marked as solved and the number of rescue items will be incremented.

Maze Generation

There are two methods where you can generate your own maze. The first method is manually building the maze using NumPy arrays and passing this array to the init_maze() function which can be found in the maze_manager.py file.

The second method is using the random maze generator. The random maze generator can be found in the maze_env.py file. These generators allow you to build random mazes of sizes 3x3, 5x5, 10x10, 20x20, 30x30, 100x100.

More details about the maze representation can be found in the Maze Representation section.

Phases

Initial Phase

- Each team will get 5 attempts to submit a solution to the first phase of the competition. You can develop and test your solution locally on the provided repository with no limits. However, once you start interacting with the HackTrick Server, each attempt will be tracked and scored according to your solution's performance.
- To develop and test your solution, you are provided with sample mazes and sample riddle questions. You can use those to build your solution, although the mazes and riddles during submission **will** be different.
- You are required to submit your solution before the initial phase deadline, any submissions sent to the server after the deadline will be blocked.
- The top 10 teams from the initial phase will be informed that they are progressing to the final phase.

Final Phase

- Each team will get 1 attempt to submit a solution to the final phase of the competition. You can develop and test your solution locally on the provided repository with no limits. However, once you start interacting with the HackTrick Server, your attempt will be tracked and scored according to your solution's performance.
- Within the deadline of 17th of March, you will be required to submit a 10x10 maze generated by you to enter the final phase. Failure to do so within the deadline will result in disqualification of your

team. More documentation on the maze submission process and requirements are provided in the Maze Submission Documentation.

- You can build on top of your solution from the initial phase if you are satisfied with it, or you can make modifications.
- The final phase will be run in a “league” format, where your solution will be run against all other teams’ generated mazes.
- The top 4 teams from the final “league” phase will be qualified to deliver the business pitch to a panel.