REPORT

Member:

Lê Huỳnh Minh Trí- 18125144

Nguyễn Hoàng Long- 18125096

Nguyễn Hữu Khang- 18125086

Nguyễn Tiến Anh- 18125002

1. Algorithm Idea:

Our team developed a specific heuristic algorithm for intelligent agent (seeker) using for all 3 levels. We can change between each level depend on the parameter tranferred to function. The idea for this algorithm mostly come from Manhattan distance. Inside seeker, there is a container containing potential targets. Every turn of this seeker, it will calculate each posible move (not move to the wall or outside of the board) to each target from the container of seeker by using Manhattan distance and it will chose the smallest one. If the container is empty, it will move randomly until hider tell seeker where they are. In theory, this method is not optimal because in some map, the seeker can get stuck in 3-side wall and take time to move out there. If the seeker get stuck (20 moves but it only travels less than 5 squares), it will randomly change it target to another conners of the board.

Besides, hiders in level 3 have the view. Once they realize their location is in view of seeker, they will move as far as posible. If not, it random.

In level 4, we use the Qlearning algorithm for hider and hard-coded heuristic for seeker (the detail in below).

2. Assignment Plan:

We construct this project by using OOP (Create class Seeker, Hider, Obstacle and each object contacts to each other if neccessary through function in each class).

There are 4 weeks for us to complete the project, so our plan is:

+ First week: learning pygame, build UI

+ Second week: complete level 1-2-3

+Third and final week: learning Qlearning algorithm and finish level 4

3. Environment:

In this project, you have to install pygame package if you want to run locally on your computer. But don't worry, we install virtual environment for python with all packages required. All icon for graphics game, all packages and code are in one file. You download and run in this folder.

In Code, we just use pygame package for graphics and the other is standard library of Python. Besides that, folder will contain maps which you can adjust to create whatever you want. Actually, you can change any obstacles, hiders' location and seeker's location in a fixed map 13x12.

4. Completion:

We have completed 3 levels: level 1, level 2 and level 3 and 50% level 4 with satisfying all requirements in each level. We can change between each level by adjusting map in text file and parameters tranferred to functions. Actually, we don't use any game point to evaluate agent's performance as well as data for algorithm, so, we don't show game point.

No.	Specifications	Completion
1	Finish level 1.	100%
2	Finish level 2.	100%
3	Finish level 3.	100%
4	Level 4	50%
5	Graphical demonstration	100%
6	Generate 5 maps	100%
7	Report on algorithm, experiments	100%

5. Demo level 1-2-3:

We upload a demo video on Youtube. You can watch it if you don't want to run our project locally on your computer.

Here is the link

https://youtu.be/2wI78KolJfs

II. More on level 4:

We encountered the problem that to use Qlearning we must have a proper RL environment that have method like step, reset, check. Therefore, the previous environment we build for level 1-2-3 is not working. That why our level 4 is completely different from other levels.

There is only 1 class GameMap (the environment).

1. **Our game mechanics for level 4:**

   + Hider has 4 seconds pregame to move (about 40 moves)

   + Seeker has 8 second in game to move (about 80 moves)

   + Hider can only move in 4 directions (because I want to minimize the qtable)

   +Seeker can not move obstacles (I think it is more reasonable because if it can do that, it will be too strong)

   + Hider can push the obstacles.

   + Number of obstacles must equal or greater than number of holes in the wall, or else it's impossible for hider to win

   We don't have input map or any parameters passed in. To run just run the Main.py with no parameters.

   To train: go to Main.py -> change Training to True

   To retrain: go to config.py -> change Init Qlearing to True

2. **Algorithm: Qlearning**
   - We choose Qlearing because it is model-free and very easy to implement which suitable for who don't have DeepLearning knowledge like us.
   - QLearning use a 2D array(table) to represent all the states and action. The algorithm will try and find the best action in each state based on the reward it received.
   - For every 50 episodes, it will see if them better than previous ones and update the table.

   Reward for hider:

   + Jump in wall or out the map: -1000

   + Has been caught: -300

   + Win: +150

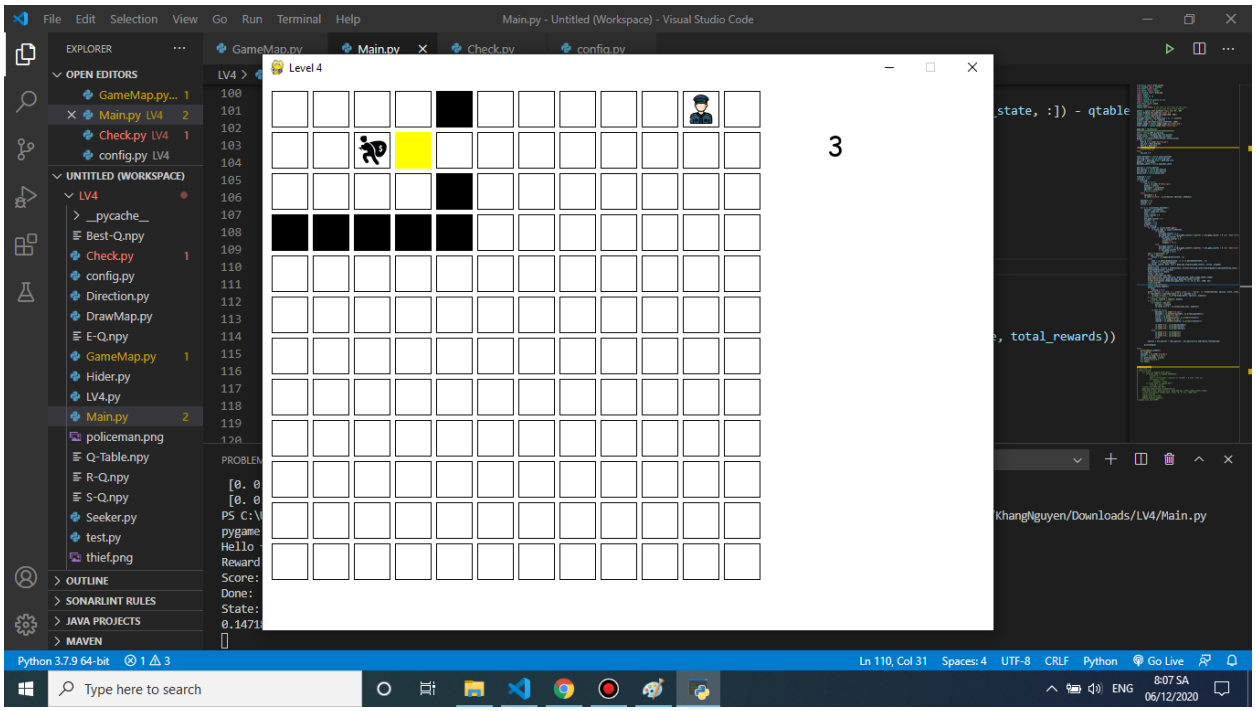
+ Fill 1 hole in the wall: +50

+ Push the obstacle out of the wall: -150

3. Example:

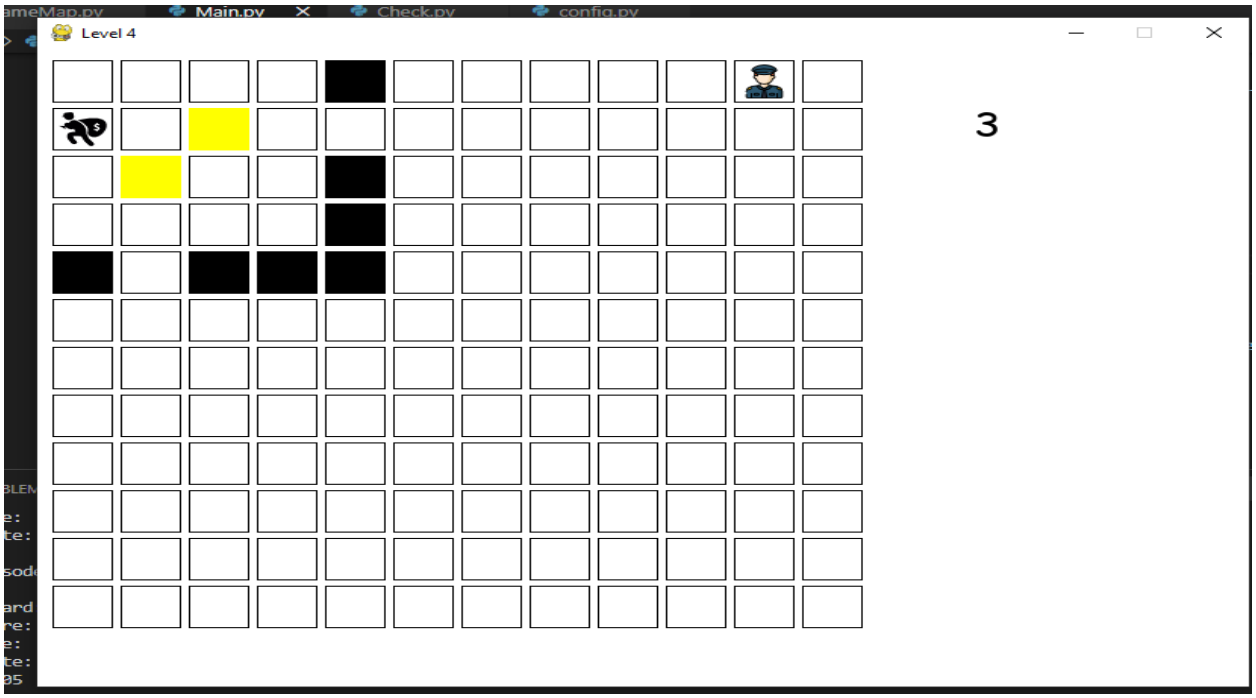
We use the algorithm to solve the map from easiest to hard.

3.1 **1 obstacle:**



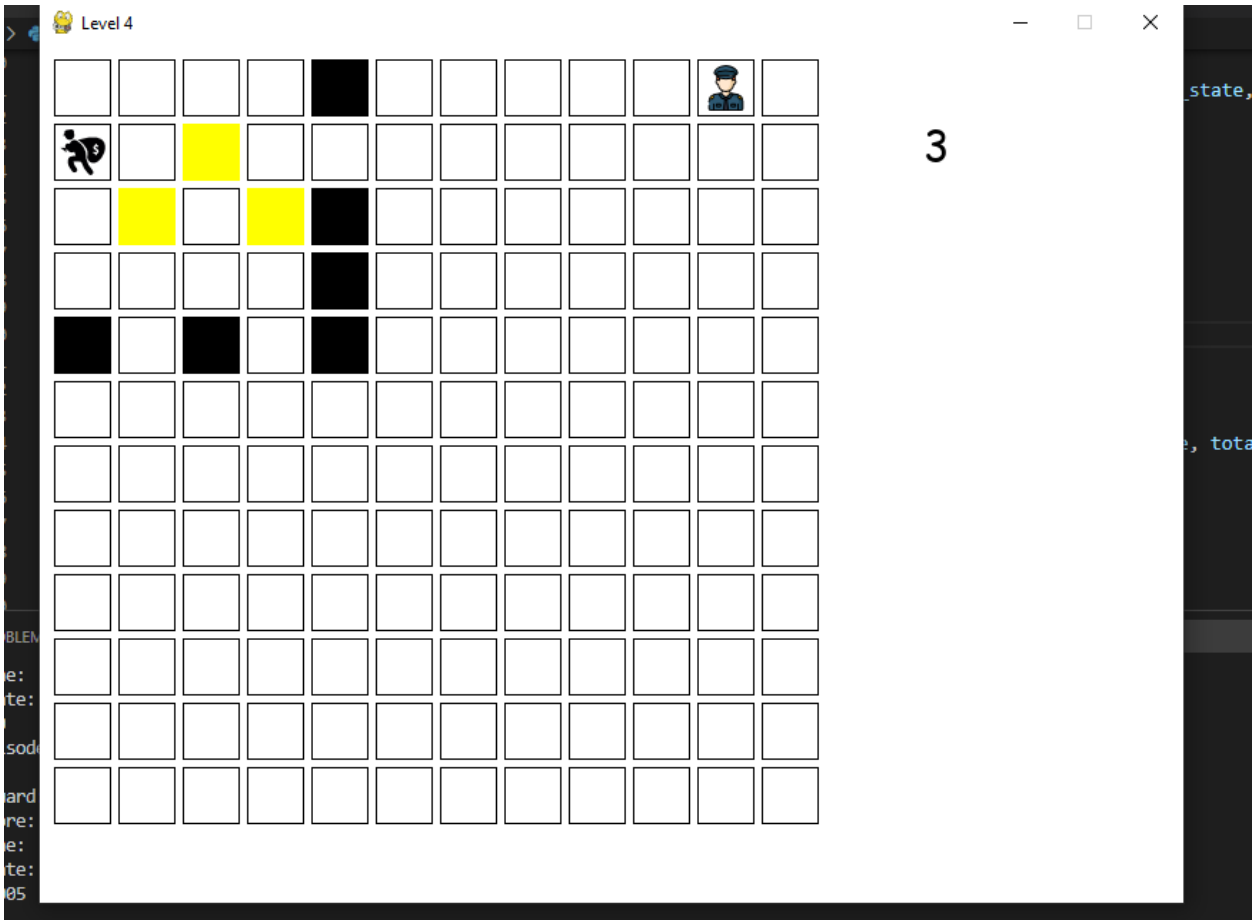
The algorithm work really well, hider can find solution after around 20 episodes.

3.2 2 obstacles:



Use the qtable from (1) we can find solution after about 300 episodes.

3.3 3 obstacles:



We reuse the qtable from (2) to solve this map.

The algorithm shows its weekness:

+ The state space is so large that make training very slow

+ The algorithm gets stuck at local maximal

+ If we train for a long time, it can easily forget the solution.

Hider can fill 1 hole in the walls after around 30 episodes, 2 holes after around 250 episodes. After 1000 episodes, it can not find a complete solution.

Demo on the level 4 video folder.

4. **Conclusion:**

From our experiment, we think that Qlearning can not use to solve this game as the map become more complicated because it can not keep track of such many states.

DeepQ Learning or DoubleQ learning or Neat maybe good methods to try.

References:

- https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c
- https://github.com/jasonwu0731/AI-Pacman/tree/master/Pacman/hw3-reinforcement/reinforcement