Project 01
Hide and Seek

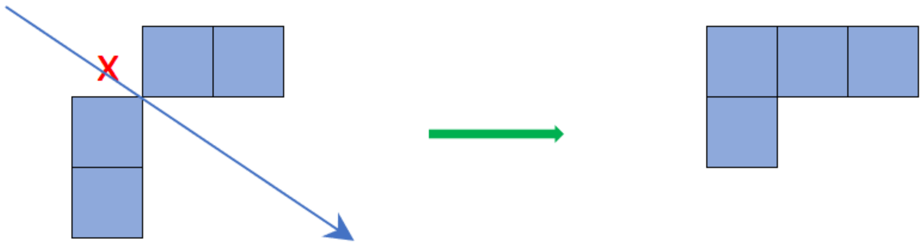
1. Description

You are given a file describing a game that I think you and I played it when we were boys and girls (the time we were all angels, but now beast is a name you could call me 😊). The game is Hide and Seek. In this game, you can play as seeker or hider.

The game is held on a 2D rectangle/square board with limited width and height in the cell unit, and its edges are parallel with screen's. Inside the game board, there are walls, some movable obstacles (shape: rectangle, square), a seeker, some hiders. You are required to make this game with intelligence agents.

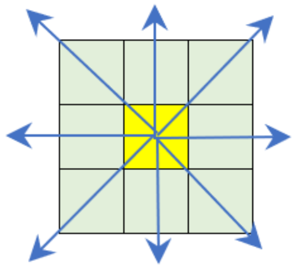
There are some rules in the game you need to know carefully:

- The board is covered by wall.
- The walls and obstacles are placed perpendicular to the board, not oblique. We should not create a gap between two walls by placing two blocks together only at one vertex.

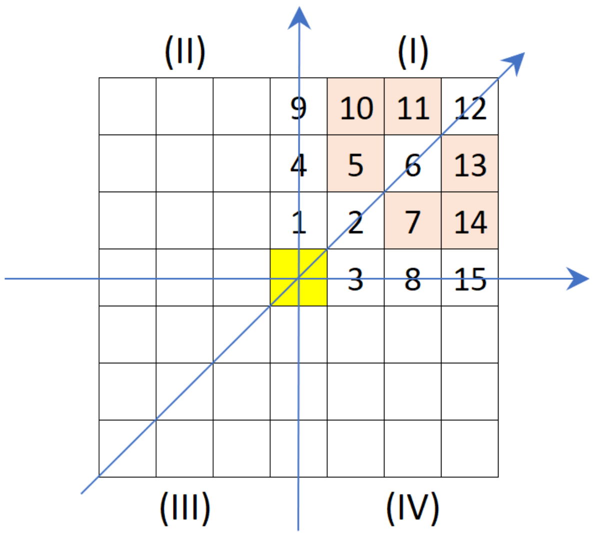


- Each agent takes up one cell in space.

- Agents can move in 8 directions (left, right, up, down, …) but not move over or through objects.



- Each agent only moves to the next one cell at a time and it takes them one unit in distance.
- In the beginning, agents know the map clearly such as the map size, their position on the map, wall position, obstacles.
- The game is over if seeker catch all hiders (win) or time out (lose). The seeker must touch the hiders if they want to catch them.
- The agent can observe within a certain range (for simple, in a square with a radius is 3 units). When there are walls/obstacles in this area, they will hide some cells from agents. For simple, I describe some situations which hide cells.



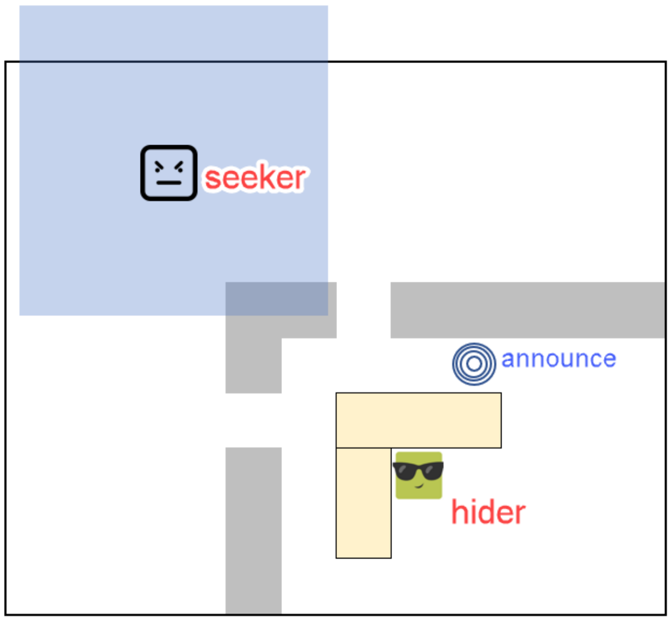
We consider the first quarter, and the remaining quarters are the same.

- Any cell in the blue line will block the following cells if it is placed with wall/obstacles. For example, if c1 is blocked, c4 and c9 are hidden.
- c5 hides c10, c11
- c7 hides c13, c14

- c2 hides c11, c13 more.
- c1 hides c10 more
- c3 hides c14 more.

While moving, you need to mark observable cells.

After about 5-10 units of time (steps), hiders announce their location to the seeker when they are in turn. However, hiders just announce one random cell around their location in 3 unit range.



To solve it from simple to advance, I build up the 4 different levels:

- Level 1: There is only one hider and remain in place all time. Don't limit time (maximum steps).
- Level 2: There are many hiders and remain in place all time. The seeker knows the number of hiders. Don't limit time.
- Level 3: Hiders can be movable but they only observe in 2 units range. Each agent moves in turn. Don't limit time.
- Level 4: Hiders can move obstacles to other places before the game start (seeker do his task) and hiders cannot change them during the game is happening. Time is limited. The seeker can move obstacles. All obstacles are repositioned in valid operation.

Game points are calculated as the following rules:

- Each moving step, your point will be decreased by 1.

- Each hider the seeker catch, 20 points will be given.

You need to run the algorithm(s) on many different maps to make a comprehensive comparison of the performance. You should generate some difficult maps.

You can view demonstration in this link:
https://www.youtube.com/watch?v=kopoLzvh5jY&list=RDCMUCXZCJLdBC09xxGZ6gcdrc6A&index=1&ab_channel=OpenAI

2. Specifications

- **Input:** the given graph is represented by its adjacency matrix, which is stored in the input file, for example, **map1.txt**. The input file format is described as follows:
  - The first line contains two integers N x M, which is the size of map.
  - N next lines represent the N × M map matrix. Each line contains M integers. The number at [i, j] (row i, column j) determines whether wall, hiders or seeker is set. If there is wall at this position, we will have value 1. If there is hider, we will have value 2. If there is seeker, we will have 3. Otherwise (empty path), we will have 0.
  - The last lines store 4 pair numbers indicate top, left, bottom, right of each obstacle.
- **Output**:
  - If you don't use graphic, you can display a result map in text file with pathfinding for seeker, path length, game point such as **result1.txt**. You can display each step of moving or display all steps in one map. However, in case that hiders can move, you must separate steps clearly.
  - I recommend you should use some graphic library for display results.

3. Requirements

No.	**Specifications**	**Scores**
1	Finish level 1 successfully.	15%
2	Finish level 2 successfully.	15%
3	Finish level 3 successfully.	15%
4	Finish level 4 successfully.	10%

5	Graphical demonstration of each step of the running process. You can demo in console screen or use any other graphical library.	10%
6	Generate at least 5 maps with difference in number and structure of walls, hiders, seeker, and obstacles.	5%
7	Report your algorithm, experiment with some reflection or comments.	30%
Total		100%

4. Notice

- This is a **GROUP** assignment. Each group has a maximum of 4 members.
- Duration: about 3 or 4 weeks.
- Your group can use any programming language to do.
- Beside above requirements, report must also give the following information:
    - Your detail information (Student Id, Full Name)
    - Assignment Plan
    - Environment to compile and run your program.
    - Estimating the degree of completion level for each requirement.
    - References (if any)
- Any plagiarism, any tricks, or any lie will have 0 point for course grade.