Report

Detailed Information

> Student 1
> Name: Bùi Lê Gia Cát
> SID: 20125071

> Student 2
> Name: Lê Phạm Nhật Quỳnh
> SID: 20125110

> Student 3
> Name: Trương Thúy Tường Vy
> SID: 20125125

Self-assessment

No	Criteria	Complete
1.1	Create 3 maps with size: 16 x 16, 32 x 32, 64 x 64	Complete
1.2	Create 2 maps with size: 2 maps larger than 64 x 64	Complete
1.3	Implement a map generator	Complete
2	Implement the game rule	Complete
3	Implement the logical agent	Complete
4	Implement the visualization tools	Complete
5	The agent can handle a complex map with large size, many of regions, prison, mountains	Complete
6	Report your implementation with some reflection or comments	Complete
7	Contest	Complete

Usage

To run the game, first you need to go to `Treasure_Island` folder;

```
cd .\Treasure_Island\
```

The main function is placed in `game.py` and it has to be provided with three arguments: the input folder, the output folder and the visualization flag

For example, by running the following command:

```
game.py input\ output\ False
```

Meaning that `main` function in game.py is executed, the input text files are stored in folder `input`, similarly for the output text files that record game's LOG.

Visualization flag is used to indicate whether you want the game to automatically run on itself or not. If you set the visualization flag to `True`, you have to press enter in order to go to the next step.

Implementation

Maps generator

Algorithm's description

We first initialize a map of customed size filled with value -1 for all positions. Then, with function `create_ocean`, those tiles that are outlying will becomes *Ocean*, and by using function `expand_prob`, *Ocean* tiles can be expanded to inside tiles small probabilities. Then we create regions with `create_region` function; basically, we traverse the whole map and pick the first tile that has not been assigned region value, take that tile as the initial location for the region and expand to adjacent tiles until the desired quantity is reached. Then we allocate mountains, prisons and treasure by `allocate_mountain`, `allocate_prison` and `allocate_trasure` respectively.

Pseudo Code

```
CLASS Map:
    function init (size, num_of_region,num_of_prison,num_of_mountain):
        map <- list
        size <- size
        num_of_region <- num_of_region
        num_of_prison <- num_of_prison
        num_of_mountain <- num_of_mountain
        FOR i from 0 to size
```

```
            add empty list to map
            FOR j from 0 to size
                assign -1 to the position at row ith, column jth in map
            END FOR
        END FOR
        create_ocean()
        create_region()
        allocate_mountain()
        allocate_prison()
        allocate_treasure()
```

Create Ocean

Tiles that are on the edge of map will be assigned to be an OCEAN. Then with certain probability, each ocean tile will be randomly chosen and expanded the OCEAN to adjacent positions.

For maps that are big in size, the OCEAN will be expanded once more time but with lower probability so that the map will not be filled with too many OCEAN tiles.

Create Region

First we divide the region, which means we assign how many tiles for each region. Then for each region, we repeat the process:

1. Pick a location on map that has not been assigned to a region.
2. Assign that location with region.
3. From that location, recursively expand to adjacent tiles until the number of tiles in that region is reached.

Allocating Necessary Objects

Description

We simply allocate mountains, treasure and prisons by randomly selecting tiles on the map.

Hint

Algorithm's description

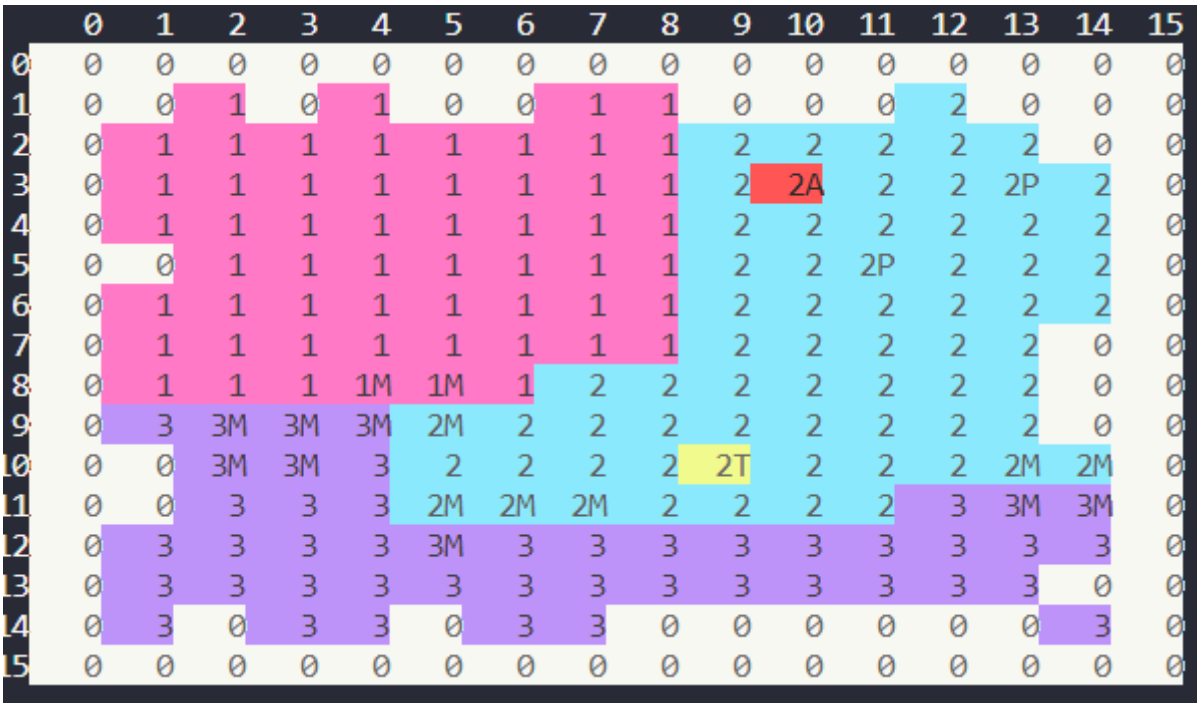
In each turn, the program call *generateHint()* function to randomly generate one of 15 hints. Those hints that are rare will have lower probability of being chosen. The program also prints out the chosen hint's explanation on the LOG; the hint is then stored and verified in the next turn

Hint 01: A list of random tiles that doesn't contain the treasure (1 to 12)

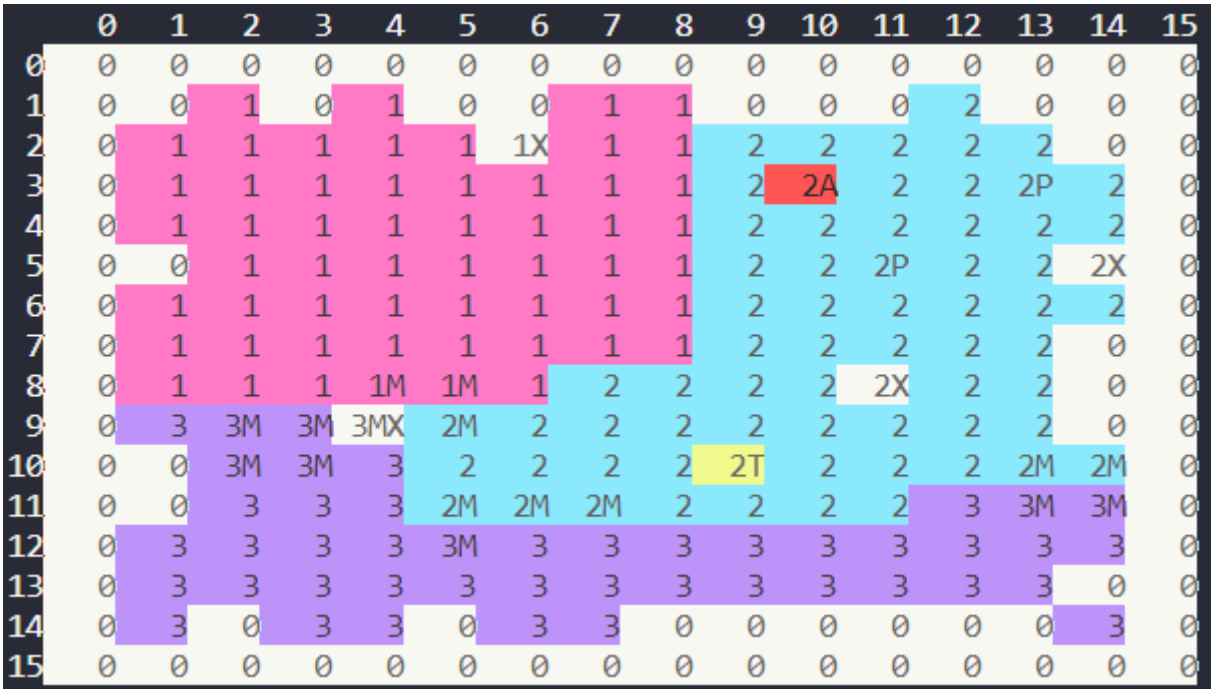
Generate Hint: For a random number of tiles, we will randomly choose 1 coordinate on the map

Verify Hint: Traverse through the list of random tiles, if one of those contain treasure, return false indicating that this hint is inaccurate; else, we return true

Illustration:



> These tile(s) do not contain the treasure: (2, 6), (9, 4), (8, 11), (5, 14)

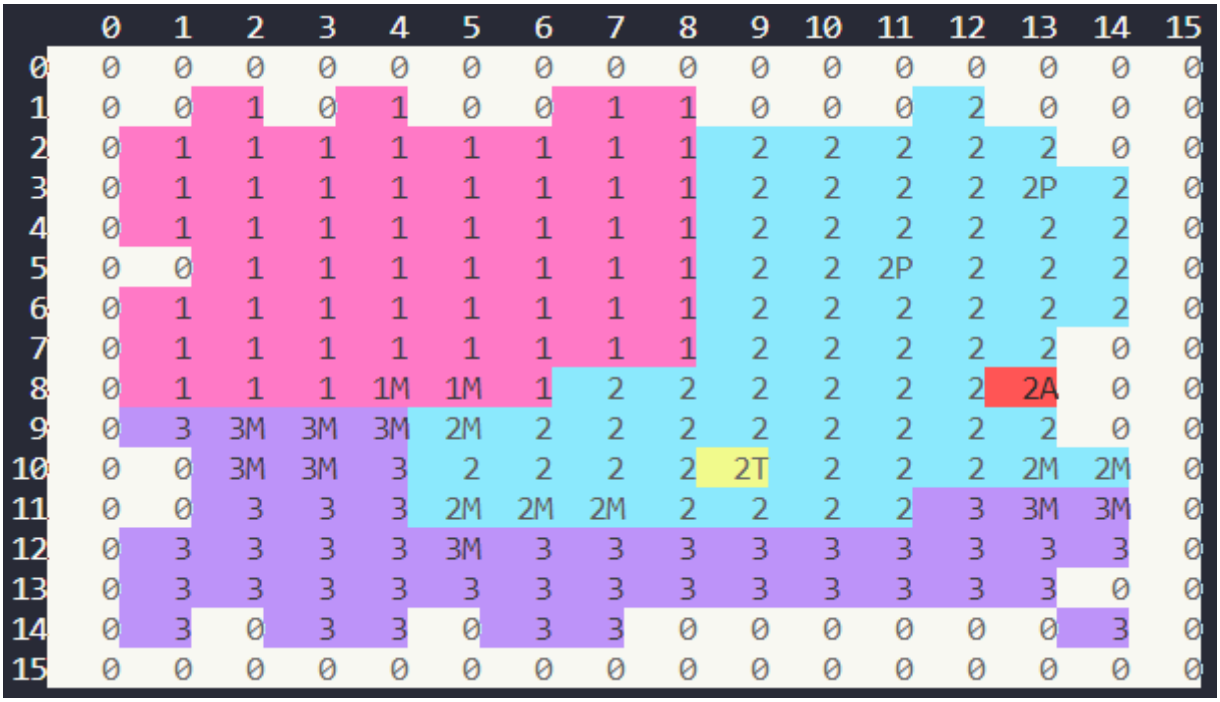


Hint 02: 2-5 regions that 1 of them has the treasure.

Generate Hint: For a random number from 2 to 5, we randomly choose a region from a list of defined regions

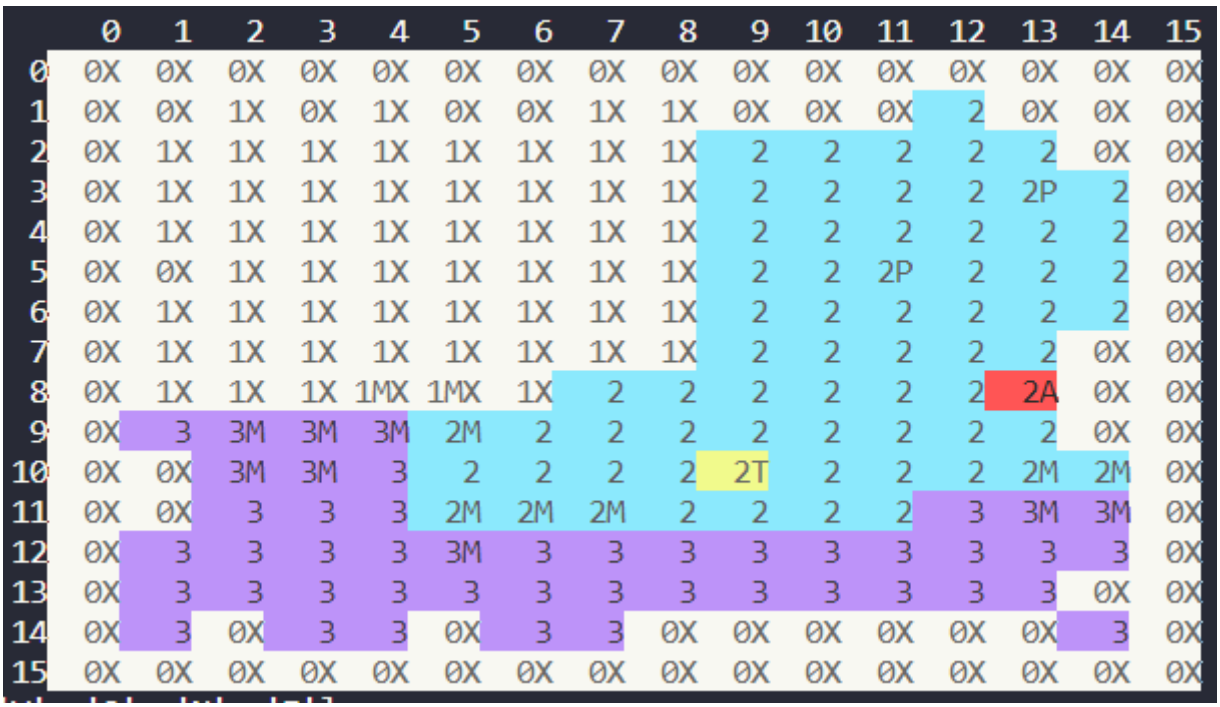
Verify Hint: If none of the regions contain treasure, the hint is inaccurate

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2A	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> Treasure is in one of these region(s): 4, 2, 3



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2	0X	0X	0X
2	0X	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	0X	0X
3	0X	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2P	2	0X
4	0X	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	2	0X
5	0X	0X	1X	1X	1X	1X	1X	1X	1X	2	2	2P	2	2	2	0X
6	0X	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	2	0X
7	0X	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	0X	0X
8	0X	1X	1X	1X	1MX	1MX	1X	2	2	2	2	2	2	2A	0X	0X
9	0X	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0X	0X
10	0X	0X	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0X
11	0X	0X	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0X
12	0X	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0X
13	0X	3	3	3	3	3	3	3	3	3	3	3	3	3	0X	0X
14	0X	3	0X	3	3	0X	3	3	0X	0X	0X	0X	0X	0X	3	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 03: 1-3 regions that do not contain the treasure.

Generate Hint: For a random number from 1 to 3, we randomly choose a region from a list of defined regions

Verify Hint: If one of the regions contain treasure, the hint is inaccurate

Illustration:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1A	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> These region(s) do not contain the treasure: 1, 4

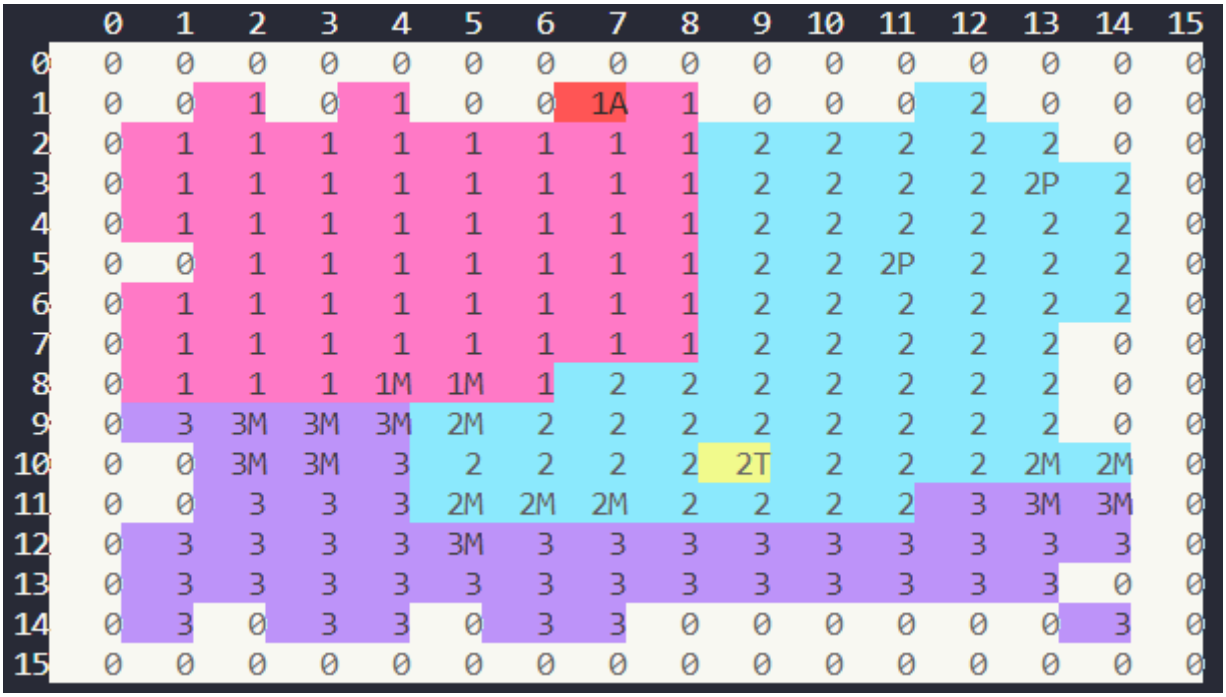
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1X	0	1X	0	0	1X	1X	0	0	0	2	0	0	0
2	0	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	0	0
3	0	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2P	2	0
4	0	1AX	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	2	0
5	0	0	1X	1X	1X	1X	1X	1X	1X	2	2	2P	2	2	2	0
6	0	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	2	0
7	0	1X	1X	1X	1X	1X	1X	1X	1X	2	2	2	2	2	0	0
8	0	1X	1X	1X	1MX	1MX	1X	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hint 04: A large rectangle area that has the treasure.

Generate Hint: Randomly choose a tile on map. From that tile, expand in all 4 directions a number of rows and columns. That number is chosen by randomly generate and it must be large enough. We consider a number that is greater that half of the map's size is a large number.

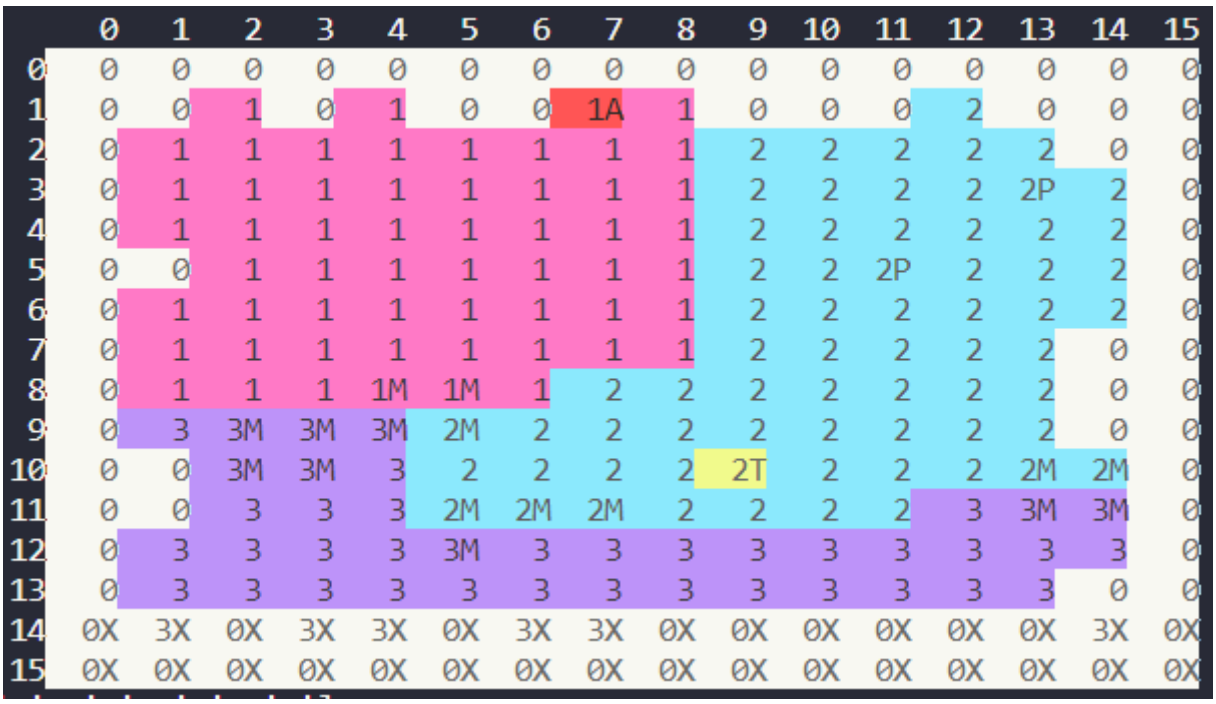
Verify Hint: If all of the tiles in the generated region do not contain treasure, then the hint is inaccurate

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1A	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> This area contains the treasure: From row 0 to row 13, from column 0 to column 15



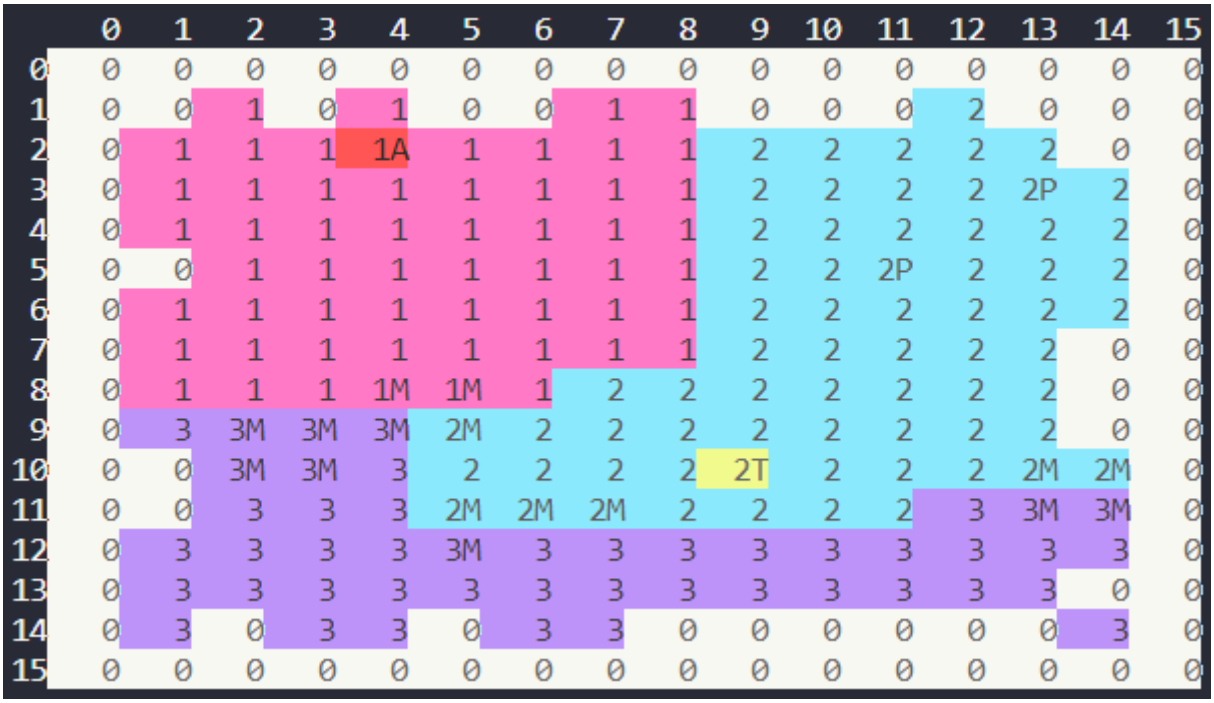
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1A	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 05: A small rectangle area that doesn't has the treasure.

Generate Hint: Randomly choose a tile on map. From that tile, expand in all 4 directions a number of rows and columns. That number is chosen by randomly generate and it must be small enough. We consider a number that is smaller that a quarter of the map's size is a small number.

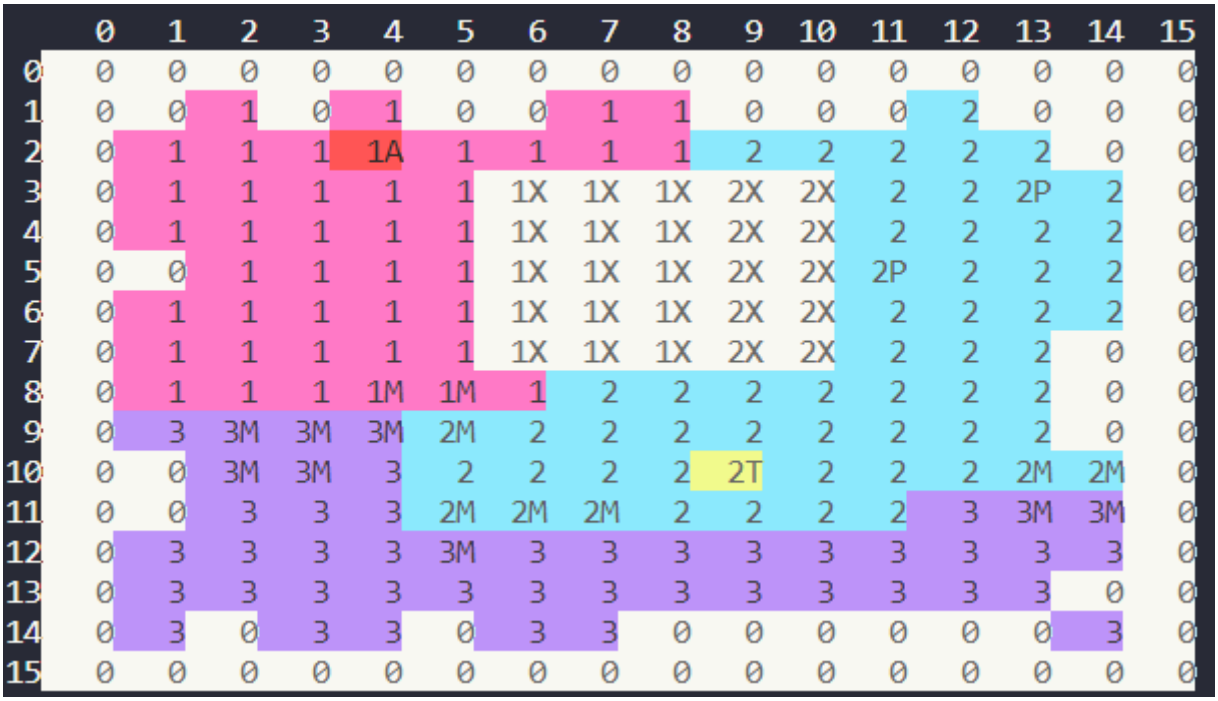
Verify Hint: If one of the tiles in the generated region contains treasure, then the hint is inaccurate

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1A	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> This area does not contain the treasure: From row 3 to row 7, from column 6 to column 10



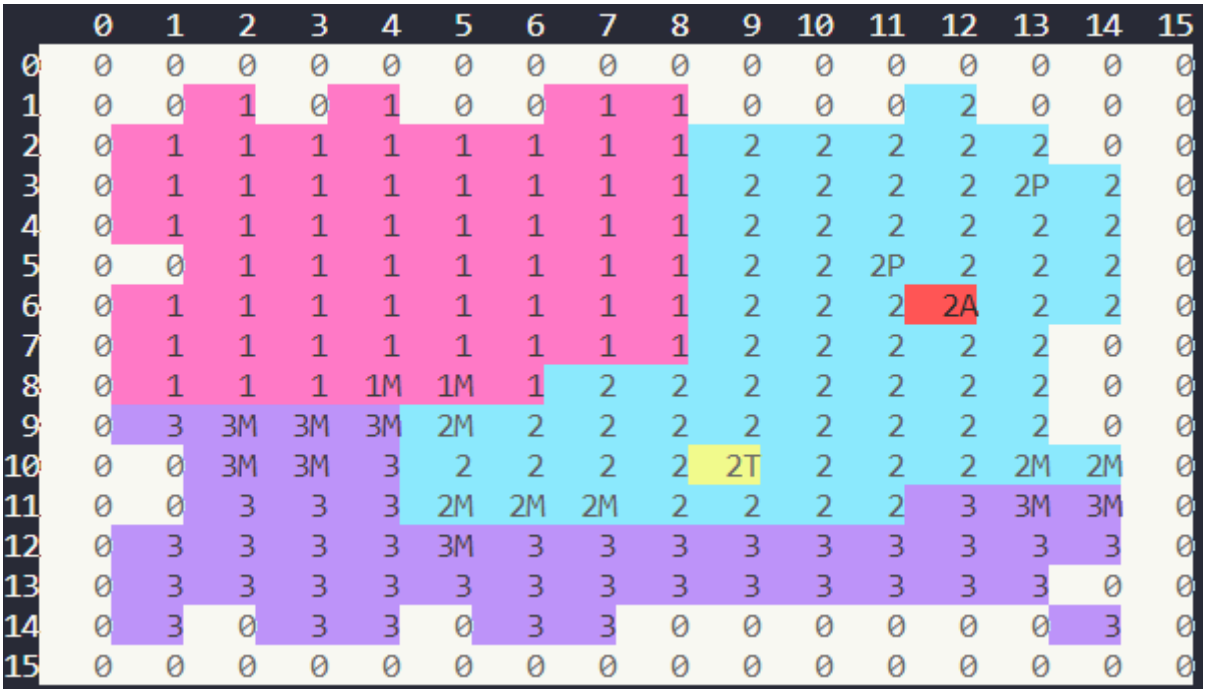
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1A	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1X	1X	1X	2X	2X	2	2	2P	2	0
4	0	1	1	1	1	1	1X	1X	1X	2X	2X	2	2	2	2	0
5	0	0	1	1	1	1	1X	1X	1X	2X	2X	2P	2	2	2	0
6	0	1	1	1	1	1	1X	1X	1X	2X	2X	2	2	2	2	0
7	0	1	1	1	1	1	1X	1X	1X	2X	2X	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hint 06: He tells you that you are the nearest person to the treasure (between you and the prison he is staying).

Generate Hint: Do nothing, we just return a number that indicates this is hint 6

Verify Hint: Comparing the distance of agent and pirate to the treasure by calculating the smallest number of tiles they have to move to get to the treasure. If the pirate is nearer to the treasure, the hint is inaccurate.

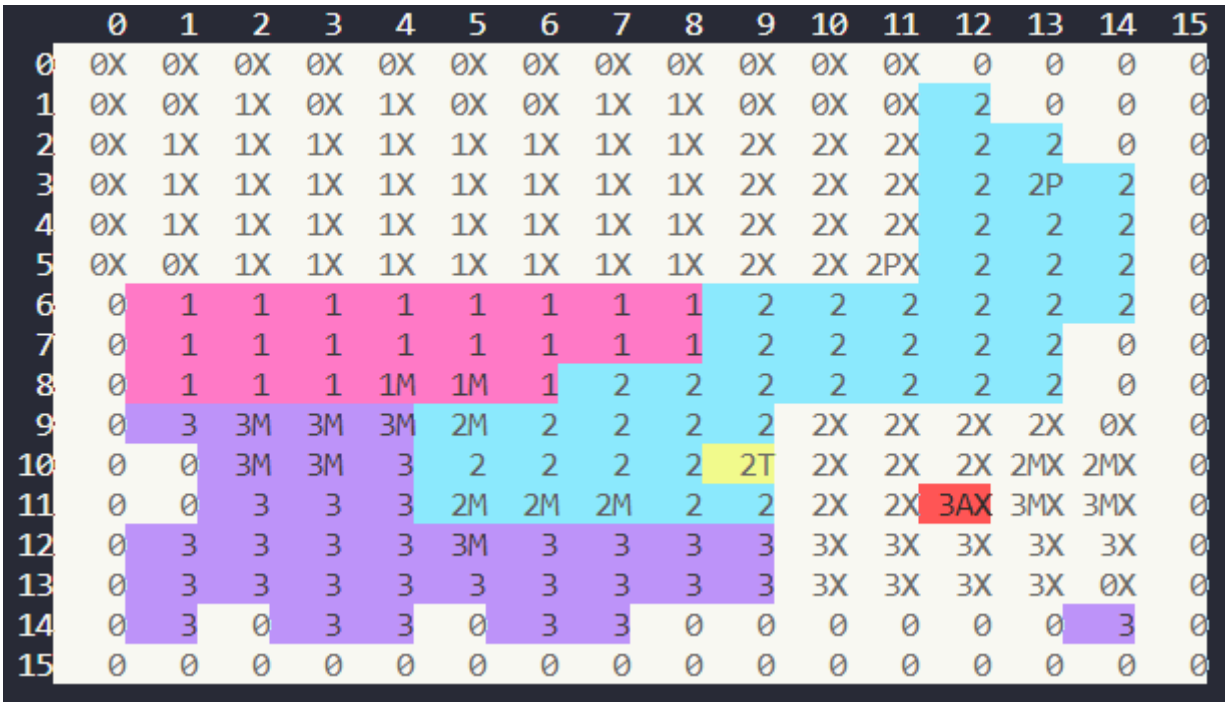
Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2A	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> You are the nearest person to the treasure (between you and the prison the pirate is staying)

In this game, the prison of pirate locate at x=5,y=11. In this hint, we calculate the Euclidean distance to mask the map



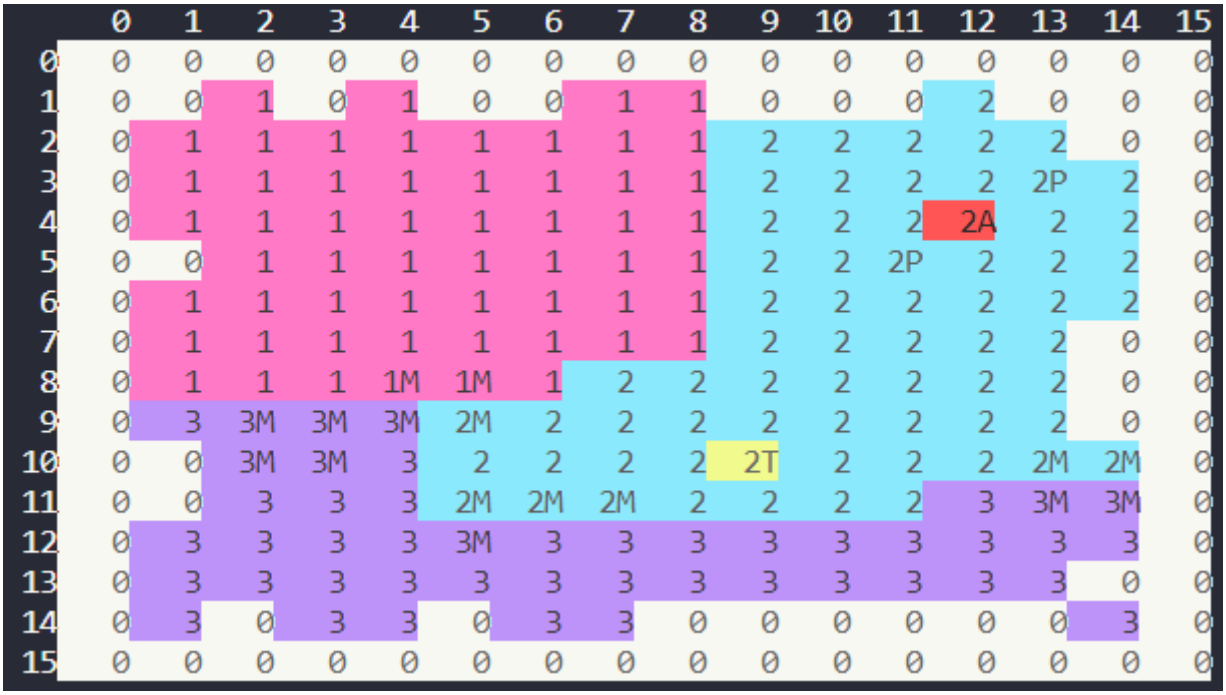
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0	0	0	0
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2	0	0	0
2	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2	2	0	0
3	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2	2P	2	0
4	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2	2	2	0
5	0X	0X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2PX	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2X	2X	2X	2X	0X	0
10	0	0	3M	3M	3	2	2	2	2	2T	2X	2X	2X	2MX	2MX	0
11	0	0	3	3	3	2M	2M	2M	2	2	2X	2X	3AX	3MX	3MX	0
12	0	3	3	3	3	3M	3	3	3	3	3X	3X	3X	3X	3X	0
13	0	3	3	3	3	3	3	3	3	3	3X	3X	3X	3X	0X	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hint 07: A column and/or a row that contain the treasure (rare).

Generate Hint: Choose randomly a row index or column index

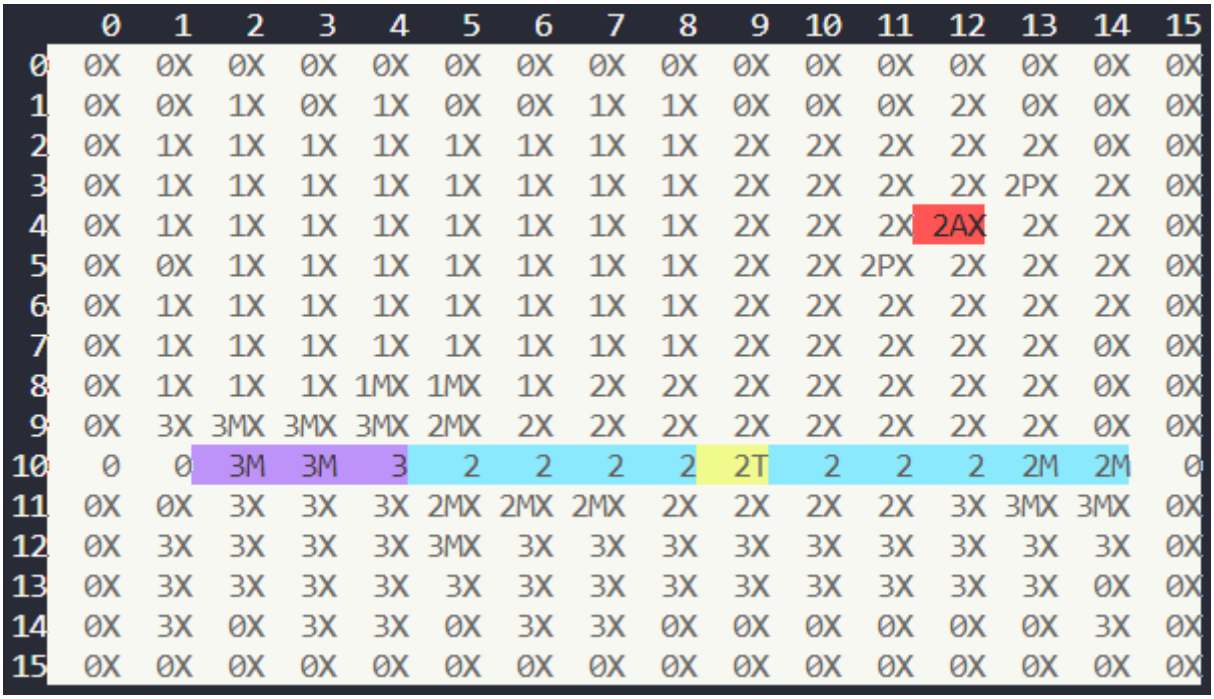
Verify Hint: If that row / column does not contain treasure, then this hint is inaccurate

Illustration



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2A	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> Row 10 contains the treasure



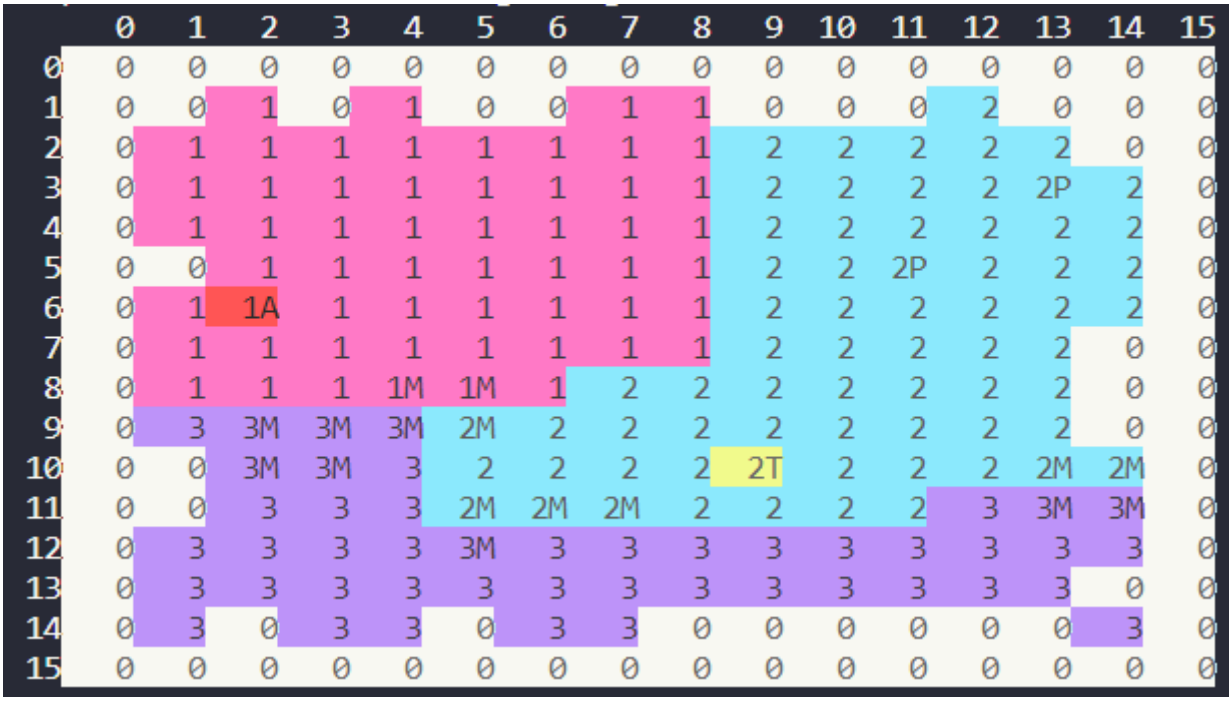
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2X	0X	0X	0X
2	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
3	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2PX	2X	0X
4	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2AX	2X	2X	0X
5	0X	0X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2PX	2X	2X	2X	0X
6	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	2X	0X
7	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
8	0X	1X	1X	1X	1MX	1MX	1X	2X	2X	2X	2X	2X	2X	2X	0X	0X
9	0X	3X	3MX	3MX	3MX	2MX	2X	2X	2X	2X	2X	2X	2X	2X	0X	0X
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0X	0X	3X	3X	3X	2MX	2MX	2MX	2X	2X	2X	2X	3X	3MX	3MX	0X
12	0X	3X	3X	3X	3X	3MX	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X
13	0X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X	0X
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 08: A column and/or a row that do not contain the treasure.

Generate Hint: Choose randomly a row index or column index

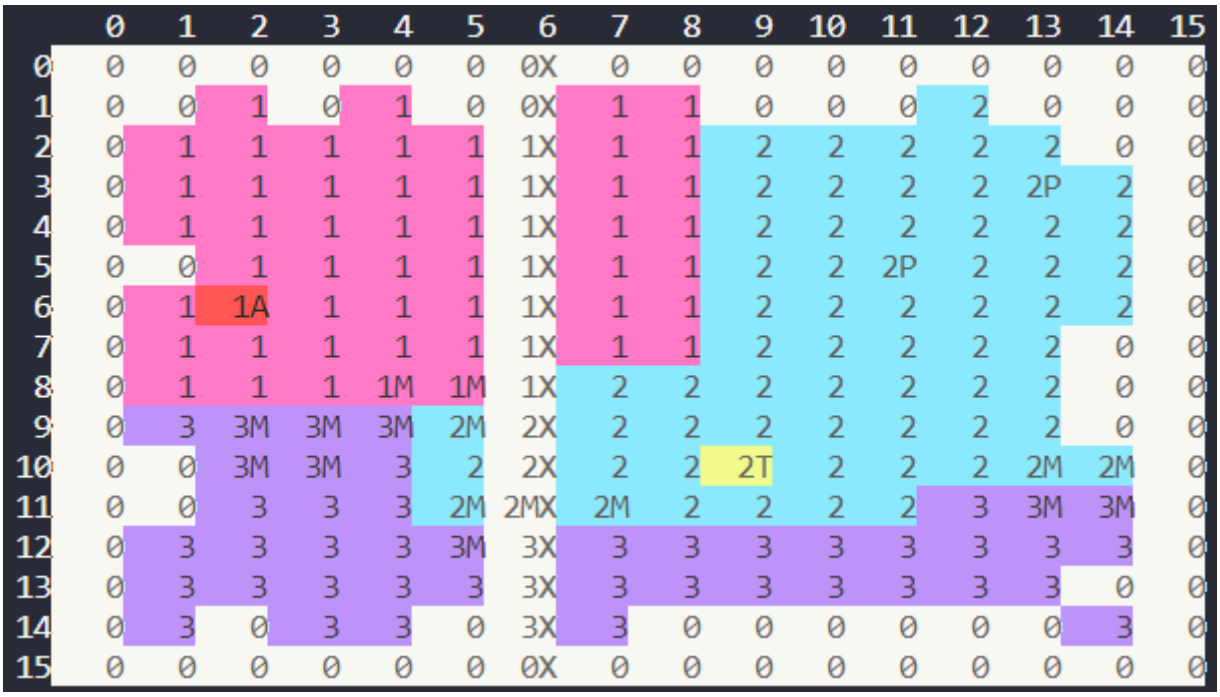
Verify Hint: If that row / column contains treasure, then this hint is inaccurate

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1A	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> Column 6 does not contain the treasure



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0X	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0X	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1X	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1X	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1X	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1X	1	1	2	2	2P	2	2	2	0
6	0	1	1A	1	1	1	1X	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1X	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1X	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2X	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2X	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2MX	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3X	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3X	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3X	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0X	0	0	0	0	0	0	0	0	0

Hint 09: 2 regions that the treasure is somewhere in their boundary.

Generate Hint: First, we store all the the regions, which is not sea, that have the same boundary. Then we randomly choose one in the list.

Verify Hint: Since we know the position of treasure, we simply check the region of treasure is one of region in the hint and at least one of the next tile of treasure is in the other region.

Illustration:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1T	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2A	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> Treasure is somewhere in these two regions' boudaries: Region 1 and region 2

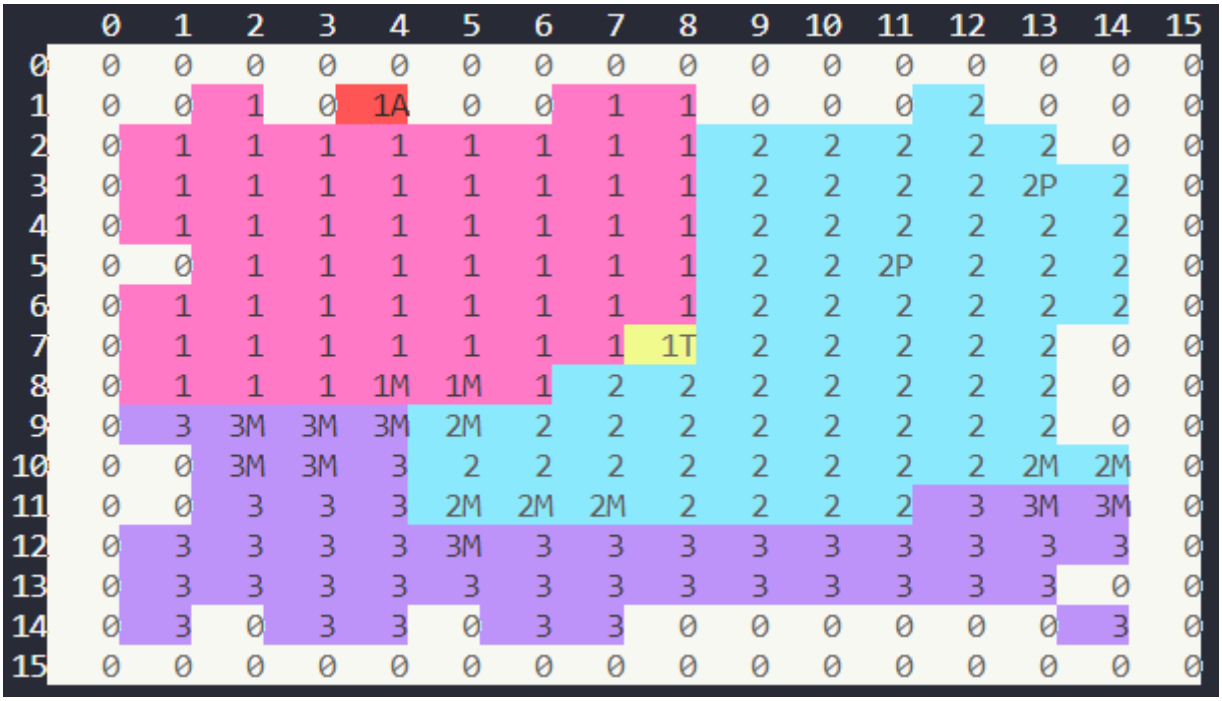
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2X	0X	0X	0X
2	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2X	0X	0X
3	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2PX	2X	0X
4	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2X	2X	0X
5	0X	0X	1X	1X	1X	1X	1X	1X	1	2	2X	2PX	2X	2X	2X	0X
6	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2X	2X	0X
7	0X	1X	1X	1X	1X	1X	1X	1	1T	2	2X	2X	2X	2X	0X	0X
8	0X	1X	1X	1X	1MX	1M	1	2	2	2X	2X	2X	2X	2X	0X	0X
9	0X	3X	3MX	3MX	3MX	2M	2	2X	2X	2X	2X	2X	2X	2X	0X	0X
10	0X	0X	3MX	3MX	3X	2X	2X	2X	2X	2X	2X	2X	2X	2MX	2MX	0X
11	0X	0X	3X	3X	3X	2MX	2MX	2MX	2X	2AX	2X	2X	3X	3MX	3MX	0X
12	0X	3X	3X	3X	3X	3MX	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X
13	0X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X	0X
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 10: The treasure is somewhere in a boundary of 2 regions.

Generate Hint: Do nothing, we just return a number that indicates this is hint 10

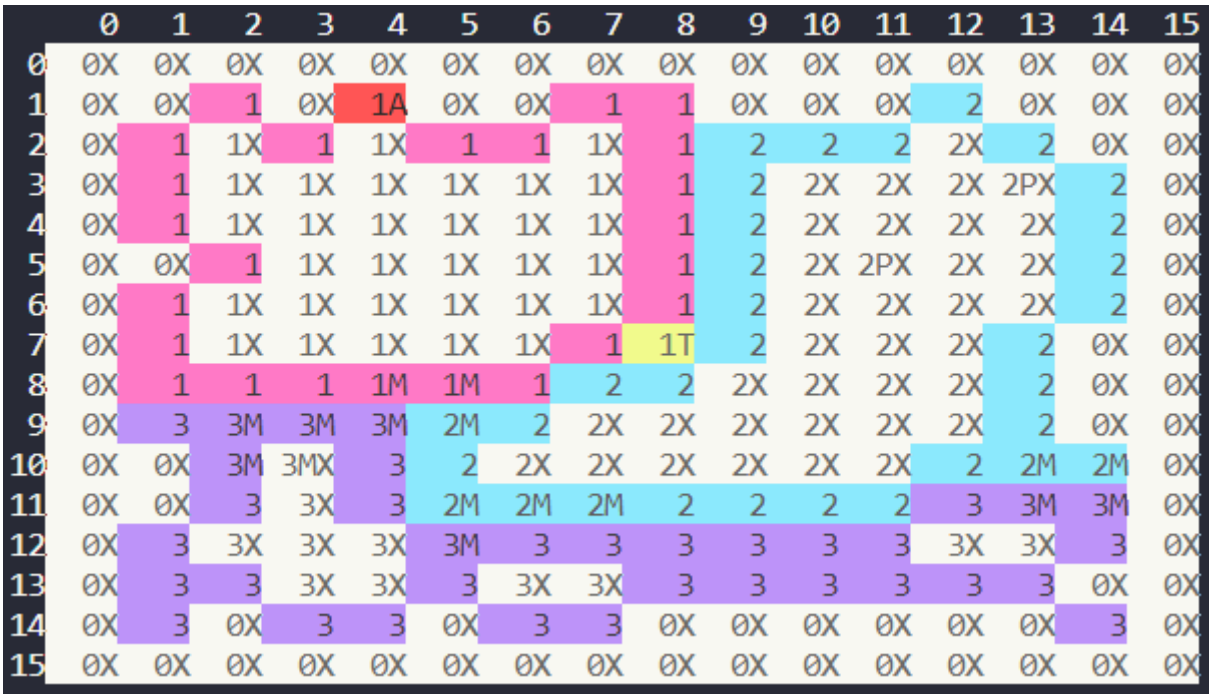
Verify Hint: Since we know the position of treasure, we check if the region the treasure in is different at least one of the region of the next tile.

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1A	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1T	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> The treasure is somewhere in a boundary of 2 regions



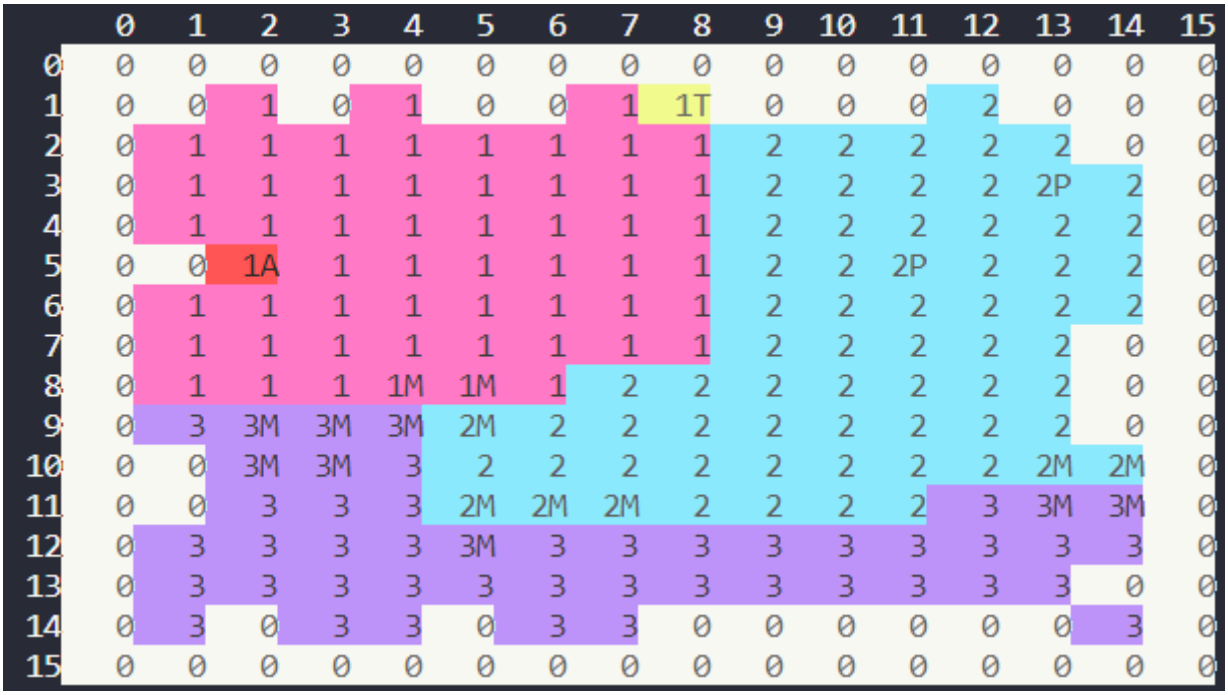
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1	0X	1A	0X	0X	1	1	0X	0X	0X	2	0X	0X	0X
2	0X	1	1X	1	1X	1	1	1X	1	2	2	2	2X	2	0X	0X
3	0X	1	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2PX	2	0X
4	0X	1	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2X	2	0X
5	0X	0X	1	1X	1X	1X	1X	1X	1	2	2X	2PX	2X	2X	2	0X
6	0X	1	1X	1X	1X	1X	1X	1X	1	2	2X	2X	2X	2X	2	0X
7	0X	1	1X	1X	1X	1X	1X	1	1T	2	2X	2X	2X	2	0X	0X
8	0X	1	1	1	1M	1M	1	2	2	2X	2X	2X	2X	2	0X	0X
9	0X	3	3M	3M	3M	2M	2	2X	2X	2X	2X	2X	2X	2	0X	0X
10	0X	0X	3M	3MX	3	2	2X	2X	2X	2X	2X	2X	2	2M	2M	0X
11	0X	0X	3	3X	3	2M	2M	2M	2	2	2	2	3	3M	3M	0X
12	0X	3	3X	3X	3X	3M	3	3	3	3	3	3	3X	3X	3	0X
13	0X	3	3	3X	3X	3	3X	3X	3	3	3	3	3	3	0X	0X
14	0X	3	0X	3	3	0X	3	3	0X	0X	0X	0X	0X	0X	3	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 11: The treasure is somewhere in an area bounded by 2-3 tiles from sea.

Generate Hint: We randomly choose 2 or 3, as the number of tiles from sea.

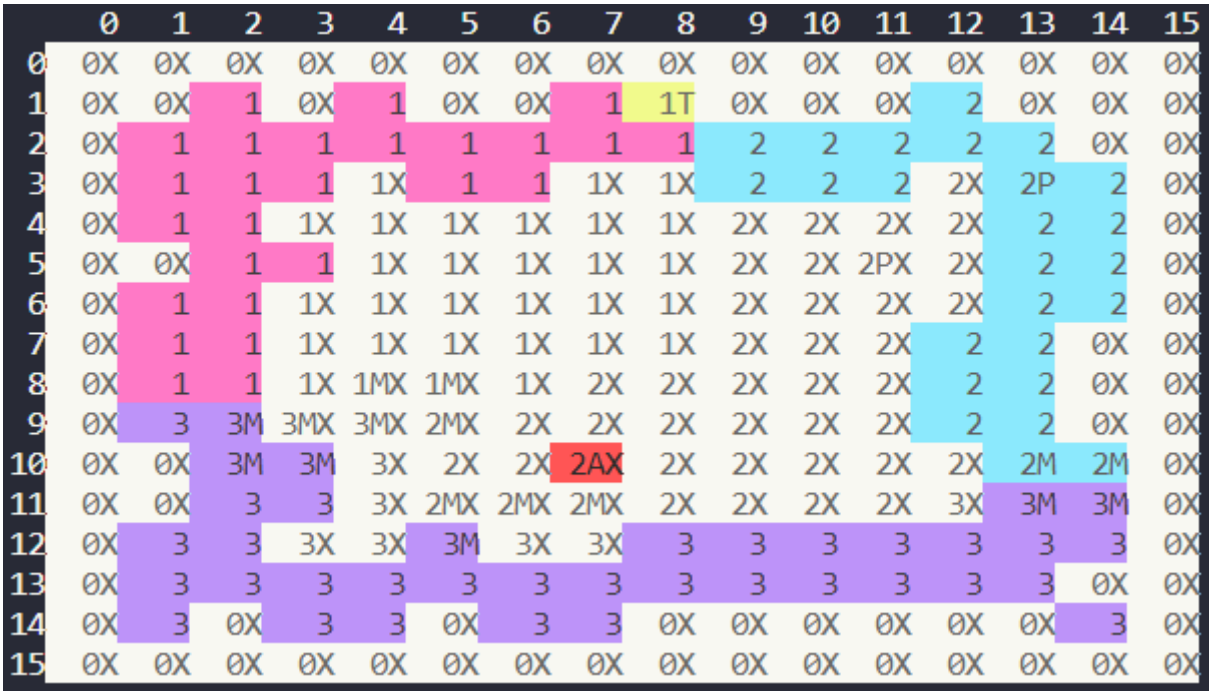
Verify Hint: Since we know the position of treasure, we simply check the shortest distance from the treasure to the sea is smaller or equal to the given hint

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1T	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1A	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> The treasure is somewhere in an area bounded by 2-3 tiles from sea



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1	0X	1	0X	0X	1	1T	0X	0X	0X	2	0X	0X	0X
2	0X	1	1	1	1	1	1	1	1	2	2	2	2	2	0X	0X
3	0X	1	1	1	1X	1	1	1X	1X	2	2	2	2X	2P	2	0X
4	0X	1	1	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2	2	0X
5	0X	0X	1	1	1X	1X	1X	1X	1X	2X	2X	2PX	2X	2	2	0X
6	0X	1	1	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2	2	0X
7	0X	1	1	1X	1X	1X	1X	1X	1X	2X	2X	2X	2	2	0X	0X
8	0X	1	1	1X	1MX	1MX	1X	2X	2X	2X	2X	2X	2	2	0X	0X
9	0X	3	3M	3MX	3MX	2MX	2X	2X	2X	2X	2X	2X	2	2	0X	0X
10	0X	0X	3M	3M	3X	2X	2X	2AX	2X	2X	2X	2X	2X	2M	2M	0X
11	0X	0X	3	3	3X	2MX	2MX	2MX	2X	2X	2X	2X	3X	3M	3M	0X
12	0X	3	3	3X	3X	3M	3X	3X	3	3	3	3	3	3	3	0X
13	0X	3	3	3	3	3	3	3	3	3	3	3	3	3	0X	0X
14	0X	3	0X	3	3	0X	3	3	0X	0X	0X	0X	0X	0X	3	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 12: A half of the map without treasure (rare).

Generate Hint:

We randomly choose a number from 1 to 4 as:

1. The treasure is not in the right half.
2. The treasure is not in the left half.
3. The treasure is not in the top half.
4. The treasure is not in the bottom half.

Verify Hint:

Since we know the position of the treasure, let x, y be the treasure position and m be the map size.

1. If $y < \frac{m}{2}$, clearly the treasure is in the left half and not in the right half of the map.
2. If $y \geq \frac{m}{2}$, the treasure is in the right half and not in the left half of the map.
3. If $x \geq \frac{m}{2}$, clearly the treasure is in the top half and not in the bottom half of the map.
4. If $x < \frac{m}{2}$, the treasure is in the bottom half and not in the top half of the map.

Illustration:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1T	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2A	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> A half left of the map does not have treasure

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0	0	0	0	0	0	0	0
1	0X	0X	1X	0X	1X	0X	0X	1X	1T	0	0	0	2	0	0	0
2	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2	2	2	2	0	0
3	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2	2	2	2P	2	0
4	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2	2	2	2	2	0
5	0X	0X	1X	1X	1X	1X	1X	1X	1	2	2	2P	2	2	2	0
6	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2	2	2	2	2	0
7	0X	1X	1X	1X	1X	1X	1X	1X	1	2	2	2	2	2	0	0
8	0X	1X	1X	1X	1MX	1MX	1X	2X	2	2	2	2	2	2	0	0
9	0X	3X	3MX	3MX	3MX	2MX	2X	2X	2	2	2	2	2	2	0	0
10	0X	0X	3MX	3MX	3X	2X	2X	2X	2	2	2	2	2	2M	2M	0
11	0X	0X	3X	3X	3X	2MX	2MX	2MX	2	2	2	2A	3	3M	3M	0
12	0X	3X	3X	3X	3X	3MX	3X	3X	3	3	3	3	3	3	3	0
13	0X	3X	3X	3X	3X	3X	3X	3X	3	3	3	3	3	3	0	0
14	0X	3X	0X	3X	3X	0X	3X	3X	0	0	0	0	0	0	3	0
15	0X	0X	0X	0X	0X	0X	0X	0X	0	0	0	0	0	0	0	0

Hint 13: From the center of the map/from the prison that he's staying, he tells you a direction that has the treasure (W, E, N, S or SE, SW, NE, NW) (The

shape of area when the hints are either W, E, N or S is triangle).

Generate Hint:

We randomly choose a number 1 or 2:

1. The hint is from the center of the map
2. The hint is from the prison of the pirate

And we also randomly choose a number from 1 to 8 as:

1. The direction that has the treasure is North.
2. The direction that has the treasure is South.
3. The direction that has the treasure is West.
4. The direction that has the treasure is East.
5. The direction that has the treasure is South East.
6. The direction that has the treasure is South West.
7. The direction that has the treasure is North East.
8. The direction that has the treasure is North West.

Verify Hint:

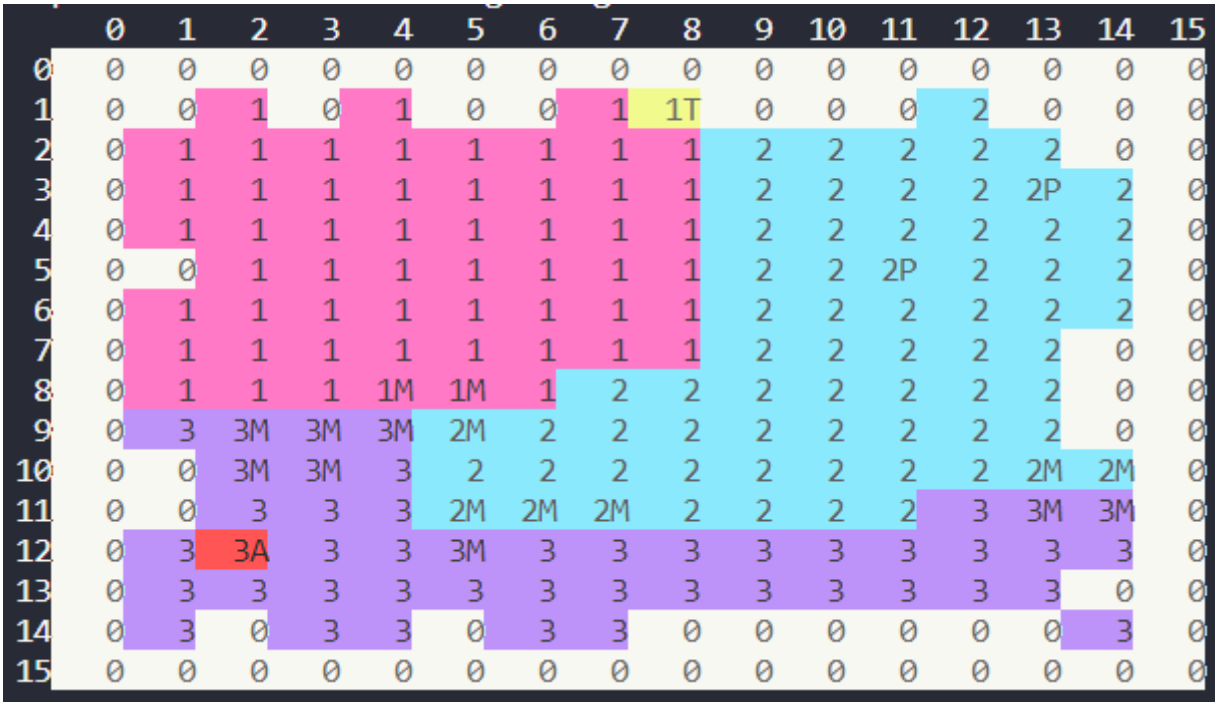
Since we know the position of the treasure, let `treasureX`, `treasureY` be the position of the treasure, m be the map size, pos be the from the center of the map or from the prison of the pirate, and direction be the direction in the hint. There are 2 main cases and each main case has 8 cases:

1. The given hint is from the center of the map:
1. If the given direction is North, the `treasureX` must be less than $\frac{m}{2}$. Let `dist` $= \frac{m}{2} -$ `treasureX`. Then if the treasure is in the north from center of the map, it must satisfy $\frac{m}{2} -$ `dist` $\leq$ `treasureY` $\leq \frac{m}{2} +$ `dist` -1.
2. If the given direction is South, the `treasureX` must be greater or equal to $\frac{m}{2}$. Let `dist` $=$ `treasureX` $- \frac{m}{2}$. Then if the treasure is in the south from center of the map, and it must satisfy $\frac{m}{2} -$ `dist` $\leq$ `treasureY` $\leq \frac{m}{2} +$ `dist` -1.
3. If the given direction is West, the `treasureY` must be less than $\frac{m}{2}$. Let `dist` $= \frac{m}{2} -$ `treasureY`. Then if the treasure is in the west from center of the map, it must satisfy $\frac{m}{2} -$ `dist` $\leq$ `treasureX` $\leq \frac{m}{2} +$ `dist` -1.
4. If the given direction is East, the `treasureY` must greater or equal to $\frac{m}{2}$. Let `dist` $=$ `treasureY` $- \frac{m}{2}$. Then if the treasure is in the east from center of the map, it must satisfy $\frac{m}{2} -$ `dist` $\leq$ `treasureX` $\leq \frac{m}{2} +$ `dist` -1.
5. If the given direction is South East, it must satisfy $\frac{m}{2} \leq$ `treasureX` and $\frac{m}{2} \leq$ `treasureY`.
6. If the given direction is South West, it must satisfy $\frac{m}{2} \leq$ `treasureX` and $\frac{m}{2} \leq$ `treasureY`.
7. If the given direction is North East, it must satisfy $\frac{m}{2} \leq$ `treasureX` and $\frac{m}{2} \leq$ `treasureY`.
8. If the given direction is North West, it must satisfy $\frac{m}{2} \leq$ `treasureX` and $\frac{m}{2} \leq$ `treasureY`.

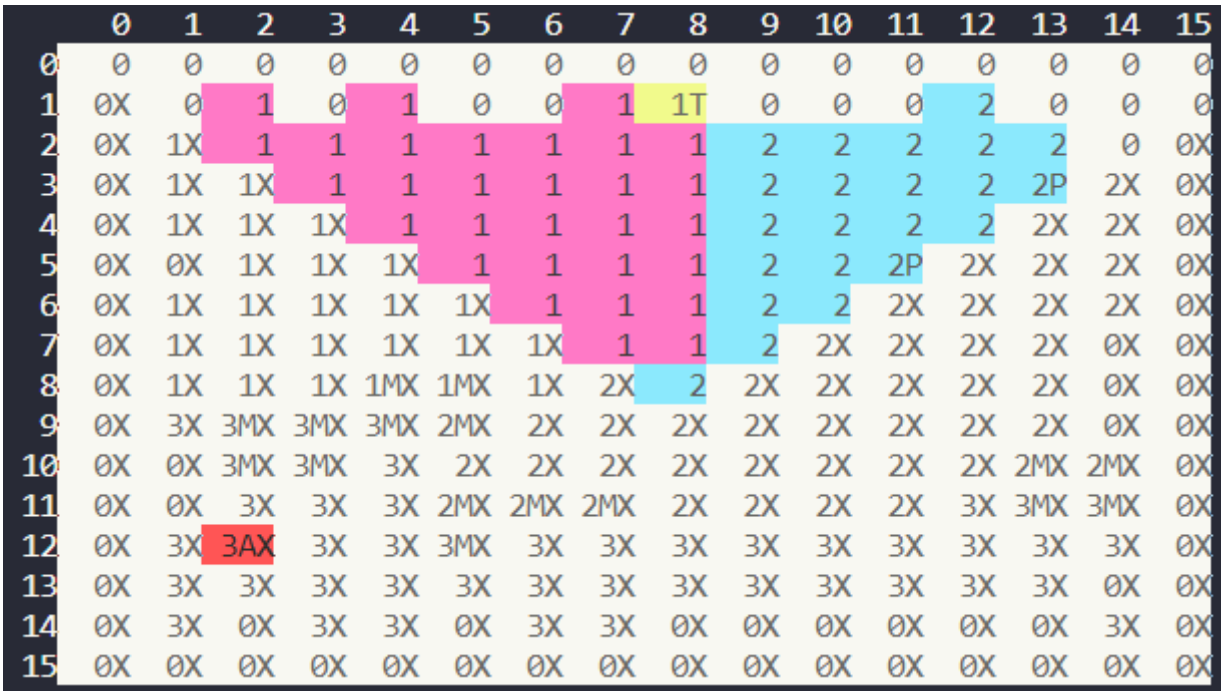
2. The given hint is from the prison of the pirate: let `prisonX`, `prisonY` is the position of the prison of pirate

1. If the given direction is North, the `treasureX` must be less than `prisonX`. Let `dist` $=$ `prisonX` $-$ `treasureX`. Then if the treasure is in the north from the prison of the, it must satisfy `prisonY` $-$ `dist` $\leq$ `treasureY` $\leq$ `prisonY` $+$ `dist` -1.
2. If the given direction is South, the `treasureX` must be greater or equal to `prisonX`. Let `dist` $=$ `treasureX` $-$ `prisonX`. Then if the treasure is in the south from the prison of the pirate, and it must satisfy `prisonY` $-$ `dist` $\leq$ `treasureY` $\leq$ `prisonY` $+$ `dist` -1.
3. If the given direction is West, the `treasureY` must be less than `prisonY`. Let `dist` $=$ `prisonY` $-$ `treasureY`. Then
if the treasure is in the west from the prison of the pirate, it must satisfy `prisonX` $-$ `dist` $\leq$ `treasureX` $\leq$ `prisonX` $+$ `dist` -1.
4. If the given direction is East, the `treasureY` must greater or equal to `prisonY`. Let `dist` $=$ `treasureY` $-$ `prisonY`. Then if the treasure is in the east from the prison of the pirate, it must satisfy `prisonX` $-$ `dist` $\leq$ `treasureX` $\leq$ `prisonX` $+$ `dist` -1.
5. If the given direction is South East, it must satisfy `prisonX` $\leq$ `treasureX` and `prisonY` $\leq$ `treasureY`.
6. If the given direction is South West, it must satisfy `prisonX` $\leq$ `treasureX` and `prisonY` $<$ `treasureY``.
7. If the given direction is North East, it must satisfy `prisonX` $<$ `treasureX` and `prisonY` $\leq$ `treasureY`.
8. If the given direction is North West, it must satisfy `prisonX` $<$ `treasureX` and `prisonY` $<$ `treasureY`.

Illustration:



> From the prison where the pirate's staying, the treasure is in the North direction



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0X	0	1	0	1	0	0	1	1T	0	0	0	2	0	0	0
2	0X	1X	1	1	1	1	1	1	1	2	2	2	2	2	0	0X
3	0X	1X	1X	1	1	1	1	1	1	2	2	2	2	2P	2X	0X
4	0X	1X	1X	1X	1	1	1	1	1	2	2	2	2	2X	2X	0X
5	0X	0X	1X	1X	1X	1	1	1	1	2	2	2P	2X	2X	2X	0X
6	0X	1X	1X	1X	1X	1X	1	1	1	2	2	2X	2X	2X	2X	0X
7	0X	1X	1X	1X	1X	1X	1X	1	1	2	2X	2X	2X	2X	0X	0X
8	0X	1X	1X	1X	1MX	1MX	1X	2X	2	2X	2X	2X	2X	2X	0X	0X
9	0X	3X	3MX	3MX	3MX	2MX	2X	2X	2X	2X	2X	2X	2X	2X	0X	0X
10	0X	0X	3MX	3MX	3X	2X	2X	2X	2X	2X	2X	2X	2X	2MX	2MX	0X
11	0X	0X	3X	3X	3X	2MX	2MX	2MX	2X	2X	2X	2X	3X	3MX	3MX	0X
12	0X	3X	3AX	3X	3X	3MX	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X
13	0X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X	0X
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 14: 2 squares that are different in size, the small one is placed inside the bigger one, the treasure is somewhere inside the gap between 2 squares. (rare)

Generate Hint:

We randomly choose 8 numbers which are top, bottom, left, right of big square and top, bot tom, left, right of small square respectively.

Let m be the map size.

For big square, we randomly choose `topBig` and `leftBig` from 0 to m - 3 since we the smal lest size of big square is 3x3 so that the small square can be 1x1. Next, we randomly choose `bottomBig` from `topBig` + 2 to m - 1 and `rightBig` from `leftBig` + 2 to m - 1. And to mak e this rectangle become square, let `size` = min{`bottomBig` - `topBig`, `rightBig` - `leftBig`}, we modify `bottomBig` and `rightBig` as `bottomBig` = `topBig` + `size` and `rightB ig` = `leftBig` + `size`.

The same process for small square, but we randomly choose `topSmall` from `topBig` + 1 to `bottomBig` - 1, `leftSmall` from `leftBig` + 1 to `rightBig` - 1, `bottomSmall` from `topSmall` to `bottomBig` - 1, and `rightSmall` from `leftSmall` to `rightSmall` - 1. To make this rectangle become suare, let `size` = min{`bottomSmall` - `topSmall`, `rightSmall` - `leftSmall`}, we modi fy `bottomSmall` = `topSmall` + `size` and `rightSmall` + `size`.

Verify Hint:

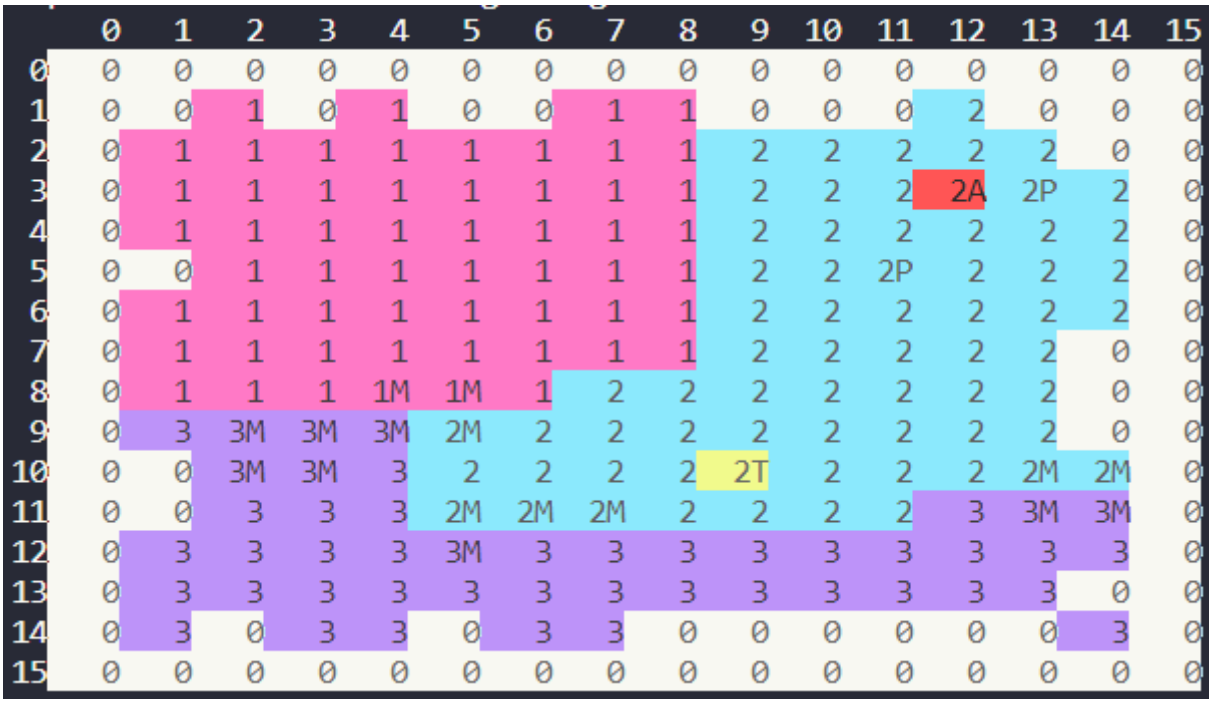
Let x, y be the position of the treasure, `topBig`, `bottomBig`, `leftBig`, `rightBig` be the top, b ottom, left, right of big square and `topSmall`, `bottomSmall`, `leftSmall`, `rightSmall` be the t

op, bottom, left, right of small square.

There are 2 main cases:

1. If `topBig` $\leq$ x $<$ `topSmall` or `bottomSmall` $<$ x $\leq$ `bottomBig`, then it must satisfy `leftBig` $\leq$ y $\leq$ `rightBig`.
2. If `leftBig` $\leq$ y $<$ `leftSmall` or `rightSmall` $<$ y $\leq$ `rightBig`, then it must satisfy `topBig` $\leq$ x $\leq$ `bottomBig`.

Illustration:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2A	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> The treasure is somewhere inside the gap between 2 squares:

```
    The bigger square bounded by row 8 on top; row 11 at the bottom; column 8 on the left and column 11 on the right
    The smaller square bounded by row 9 on top; row 9 at the bottom; column 10 on the left and column 10 on the right
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2X	0X	0X	0X
2	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
3	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2AX	2PX	2X	0X
4	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	2X	0X
5	0X	0X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2PX	2X	2X	2X	0X
6	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	2X	0X
7	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
8	0X	1X	1X	1X	1MX	1MX	1X	2X	2	2	2	2	2X	2X	0X	0X
9	0X	3X	3MX	3MX	3MX	2MX	2X	2X	2	2	2X	2	2X	2X	0X	0X
10	0X	0X	3MX	3MX	3X	2X	2X	2X	2	2T	2	2	2X	2MX	2MX	0X
11	0X	0X	3X	3X	3X	2MX	2MX	2MX	2	2	2	2	3X	3MX	3MX	0X
12	0X	3X	3X	3X	3X	3MX	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X
13	0X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X	0X
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Hint 15: The treasure is in a region that has mountain.

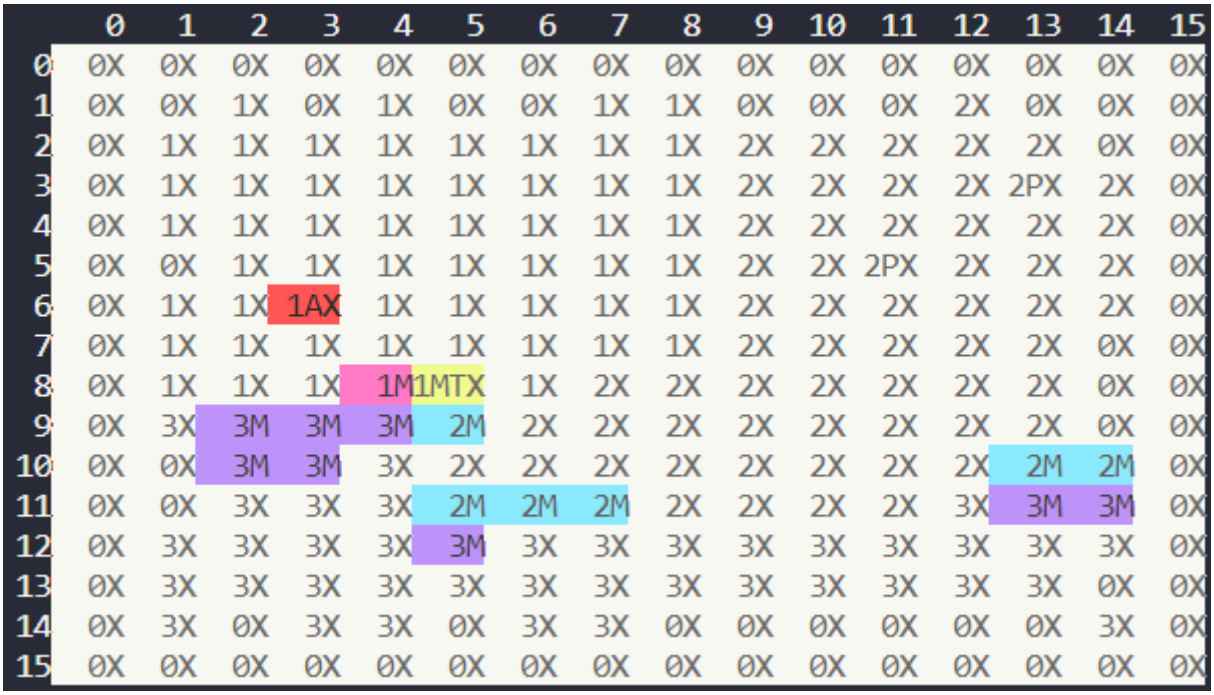
Generate Hint: Do nothing, we just return a number that indicates this is hint 15

Verify Hint: Checking the accuracy of the hint by checking whether there exists a mountain at the tile that contains treasure exists or not. If no, then the hint is inaccurate

Illustration:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1A	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1MT	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
13	0	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0
14	0	3	0	3	3	0	3	3	0	0	0	0	0	0	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

> The treasure is in a region that has mountain.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X
1	0X	0X	1X	0X	1X	0X	0X	1X	1X	0X	0X	0X	2X	0X	0X	0X
2	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
3	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2PX	2X	0X
4	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	2X	0X
5	0X	0X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2PX	2X	2X	2X	0X
6	0X	1X	1X	1AX	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	2X	0X
7	0X	1X	1X	1X	1X	1X	1X	1X	1X	2X	2X	2X	2X	2X	0X	0X
8	0X	1X	1X	1X	1M	1MTX	1X	2X	2X	2X	2X	2X	2X	2X	0X	0X
9	0X	3X	3M	3M	3M	2M	2X	2X	2X	2X	2X	2X	2X	2X	0X	0X
10	0X	0X	3M	3M	3X	2X	2X	2X	2X	2X	2X	2X	2X	2M	2M	0X
11	0X	0X	3X	3X	3X	2M	2M	2M	2X	2X	2X	2X	3X	3M	3M	0X
12	0X	3X	3X	3X	3X	3M	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X
13	0X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	3X	0X	0X
14	0X	3X	0X	3X	3X	0X	3X	3X	0X	0X	0X	0X	0X	0X	3X	0X
15	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X	0X

Game Play

Algorithm's Description

First, when game starts, we initialize agent by randomly select a position on map and place the agent there. Then, we repeat the process until the agent find the treasure or it loses to the pirate:

1. If it is turn for revealing the pirate's prison: Reveal the position on LOG
2. If it is turn for releasing the pirate: Release the pirate and announce on LOG
3. If the pirate has been released: Move the pirate
4. Each turn, the agent get one hint
5. Each turn, the agent verified the previous hint
6. Masking the map based on the authenticity of the hint
7. The agent chooses an action and execute the action
8. Print out the updated map

Move Pirate

Explanation

Each time the pirate moves, it compares the horizontal and vertical distance between itself and the treasure. If the pirate is in the same row, it only moves vertically; otherwise, if the pirate is in the same column with the treasure, it moves horizontally. On the other hand, if the pirates in neither on the same column nor the same row, it move both vertically and horizontally, each direction costs one step.

Pseudo Code

```
def pirate_move
    get position of pirate
    compare with treasure's position
    IF on same row:
        step_count = min(2, distance between treasure and pirate)
    END IF
    IF on same column:
        step_count = min(2, distance between treasure and pirate)
    END IF
    IF neither same column nor same row:
        move vertically one step closer to treasure
        move horizontally one step closer to treasure
    END IF
    mark the new position with "P" to represent the pirate
```

Map Masking

Explanation

Each time the agent verifies hint, it first check the accuracy of the hint, then based on the accuracy, it crossed out positions that do not contain treasure so that in future turns, it does not need to scan or move to those positions.

For example, if the hint tell that a certain row contains treasure; however, later on, the agent verifies the hint is inaccurate, then it will marked that certain row with `XX` symbol to indicate that it does not need to check that row in the future because definitely, the treasure does not lies in that row

Pseudo Code

```
def generate_mask(hint)
    verify hint
    add positions that surely do not contain treasure to mask list
    mask those positions
```

Agent Actions

Explanation

Each turn, the agent picks random two actions among "verification", "move large", "scan and move small", "scan large" and "teleport". However, the agent can only teleport once

Then the agent executes that two actions. If the pirate has been released, we first move the pirate, and teleport the agent as soon as the pirate appeared on map.

Pseudo Code

```
def play
    initialize the agent's position
    WHILE the result is not determined:
        IF it's time to reveal the pirate's prison
            reveal location where the private stayed
        END IF
        IF it's time to release the pirate
            release pirate
        END IF
        get the hint
        choose action 1
        WHILE true
            choose action 2
            IF action 2 different from action 1
                break out of loop
            END IF
        END WHILE
        IF pirate has been released:
            move the pirate
            teleport agent at the first turn after pirate appeared on map
            execute action 1
            execute action 2
        END IF
        IF pirate has not been released:
            execute action 1
            execute action 2
        END IF
    END WHILE
```

Logical Agent

The agent is capable of executing five actions: "verification", "move large", "scan and move small", "scan large" and "teleport".

- Small scan: Scan in horizontal and vertical direction with the total length of 3
- Large scan: Scan in horizontal and vertical direction with the total length of 5
- Small move: Move 3 or 4 steps
- Large move: Move 1 or 2 steps only
- Verification: Verify whether a hint is accurate or not.
- Teleport: Teleport the agent to another tiles

Scan

Description

The function scans around where the agent stands with the given size. If the agent scans and finds treasure, it announces in LOG and wins the game. Otherwise, it masks out tiles that contain nothing

Pseudo Code

```
def scan(size):
    x, y <- agent position
    tx, ty <- treasure position
    FOR i FROM x - size // 2 TO x + size // 2
        FOR j FROM y - size // 2 TO y + size // 2
            IF encounter treasure
                return result 'WIN'
            END IF
            ELSE
                mask out the positions where nothing is found
            END ELSE
        END FOR
    END FOR
```

Move Agent

Description

First, random the step agent has to move. If the pirate has been released, the agent simply follow the pirate since the pirate's path to treasure is implemented as the shortest path.

Otherwise, we calculate the number of unscanned tile in all four direction from where the agent is standing and rank them in descending order. Then, the direction is randomly picked from two highest result. It goes without saying that the probability of choosing the direction that has highest number of unscanned tiles is higher than the second-highest one.

However, if the agent unluckily hitting the edges while moving, it immediately goes back to the original position and move another direction.

Pseudo Code

```
def agent_move:
    ax, ay <- agent position
    IF pirate has been released
        follow pirate
```

```
    END IF
    ELSE
        direction_count <- dictionary{'N':0,'E':0,'W':0,'S':0}
        count the number of unmasked tiles in each direction
        sort direction_count in descending order
        pick randomly one direction from two directions that are largest in
direction_count
    END ELSE
    move agent
    IF agent hit edges
            return back to original positions
            WHILE True:
                    step <- 1
                    TRY:
                            IF the position of agent is Ocean, Mountain or
Prison
                                    move 1 step in the current chosen direction
                            END IF
                    EXCEPT:
                            pick randomly a direction again
            END WHILE
    END IF
        move agent
    update agent's position on map
```

Teleport

Description

First, we get the direction that the pirate will move in order to reach the treasure. Then, the agent will teleport to the position that lies 3 or 4 steps further from the pirate and in the direction that the pirate moves.

Pseudo Code

```
def teleport_agent(pirate_direction)
    agent_position <- pirate_position
    ax, ay <- agent_position
    step <- 4
    FOR i from 0 to length of pirate_direction - 1:
        IF i equals the last number in range:
            tmp_step <- step
        END IF
        ELSE
```

```
            tmp_step = step // 2
        END ELSE
        move the agent `tmp_step` toward the ith direction
        decrease step by tmp_step
    END FOR
    update the agent's position
```

Verification

Description

Get the first hint among hints that agent has received. Then, depend on what hint it is, the corresponding verify_hint function will be called and return the result.

Pseudo Code

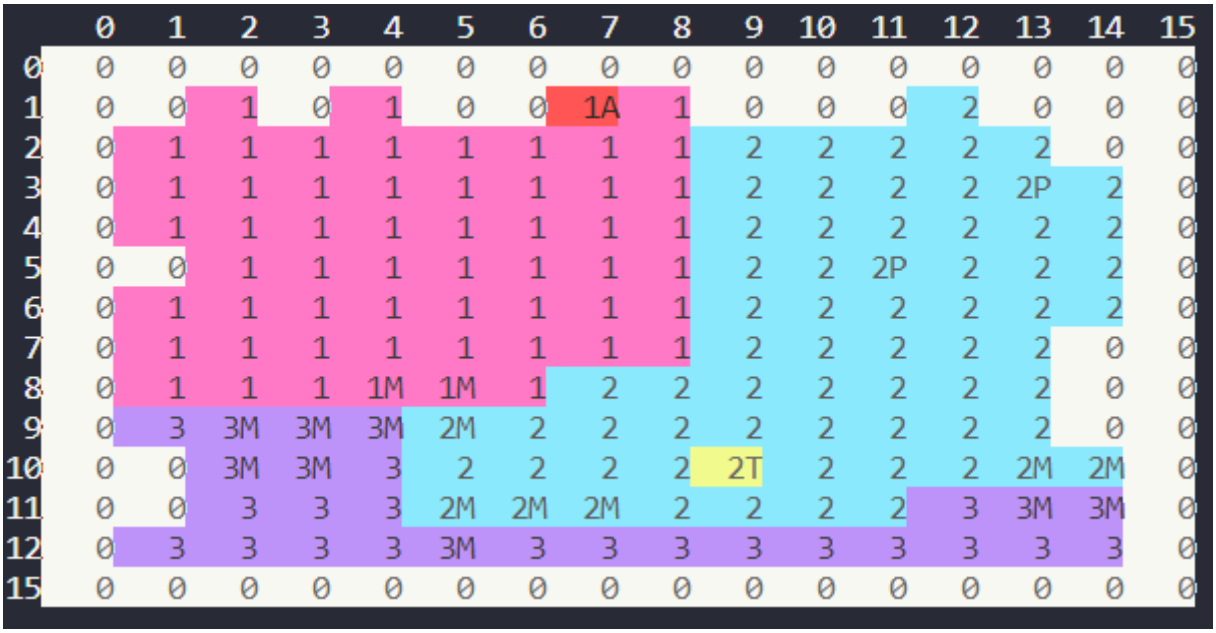
```
def verification
    pop the first stored hint
    call the corresponding verify_hint function
    print result to LOG
```

Visualization

We use `termcolor` library for visualization.

- OCEAN: White
- TREASURE: Yellow
- MASKED TILES: White
- AGENT: Red
- PIRATE: Red
- REGION:
  1. Magenta
  2. Cyan
  3. Blue
  4. Grey
  5. Green

As illustated below:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	1A	1	0	0	0	2	0	0	0
2	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
3	0	1	1	1	1	1	1	1	1	2	2	2	2	2P	2	0
4	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
5	0	0	1	1	1	1	1	1	1	2	2	2P	2	2	2	0
6	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	0
7	0	1	1	1	1	1	1	1	1	2	2	2	2	2	0	0
8	0	1	1	1	1M	1M	1	2	2	2	2	2	2	2	0	0
9	0	3	3M	3M	3M	2M	2	2	2	2	2	2	2	2	0	0
10	0	0	3M	3M	3	2	2	2	2	2T	2	2	2	2M	2M	0
11	0	0	3	3	3	2M	2M	2M	2	2	2	2	3	3M	3M	0
12	0	3	3	3	3	3M	3	3	3	3	3	3	3	3	3	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Test cases

We generate 5 maps for test cases:

- 16 x 16
- 32 x 32
- 64 x 64
- 80 x 80
- 90 x 90

Action LOG for each test case is recorded in the corresponding text file stored in `output` folder.

The game run successfully in all 5 test cases.

Evaluation

The agent is not guaranteed to win the game or find out the optimal strategy to defeat the pirate; However, the agent is still manage to have acceptable probability of winning. To evaluate the performance of the agent, our group run a total of 10 games for each map size, record the result as well as the duration to have a brief view of how many time the agent succeeds to find the treasure.

Map Size	Total Win	Duration	Average Step
16 x 16	8/10	0.01 seconds	8 steps
32 x 32	7/10	0.04 seconds	18 steps
64 x 64	7/10	0.5 seconds	38 steps
80 x 80	4/10	1.2 seconds	57 steps

Map Size	Total Win	Duration	Average Step
90 x 90	9/10	0.6 seconds	39 steps

We notice that the final result is dependent on how we organize the map and initialize agent, treasure, mountains, etc. Additionally, the result is effected quite a lot by the give hints. Therefore, the recorded result we gave above does not fully reflect how high is the agent's chance of winning because of many random factors; but it is enough to see that the agent still manages to reach the result in some cases and overall, the result is acceptable.

Comments

We manage to successfully implement all of 15 hints. After several time testing the game, we think that every hint is generated correctly and so is the verification.

However, we have not found out a better way to implement the verifying function for Hint 13. Currently, the function is quite long as the verifying process is based on 16 cases for 8 directions from center of the map and from the prison where pirate stays.

Conclusion

In conclusion, even though the agent depends quite a lot on many random factors, the game runs smoothly and is able to give the final result with acceptable winning rate in a short amount of time and steps.