# AI Assignment 1 Report

March 2022

## 1 PEAS Description of agent

### 1.1 Performance measure

Finding the book and reaching the exit in a minimum number of steps without getting caught.

### 1.2 Environment

The environment is a 9x9 cellular field with Harry as an actor we can control and two other inspector agents. Environment properties:

- **Partially observable**: we can only perceive a part of it, either a Moore neighbourhood of 1 (scenario 1) or a circle of radius 2 (scenario 2);
- **Multi-agent**: besides Harry, there are two adversarial inspectors - Argus Filch and Mrs. Norris; encounter with them leads to loss of the game;
- **Sequential**: Harry's decisions on previous steps determine his current position and state (has cloak or not);
- **Deterministic**: we can predict how the environment will behave in response to some actions of an agent;
- **Static**: the environment does not change as an agent is planning his decisions;
- **Discrete**: the environment at any time instant can be characterized by a finite set of positions of agents and objects;
- **Known**: the rules of the environment are known.

### 1.3 Actuators

Actuators are:

- **Walk on the cells**;
- **Interact with objects**;
- Interact with other agents (die by their hand, to be precise).

### 1.4 Sensors

- Harry's perception zone, either a Moore neighbourhood of 1 (scenario 1) or a circle of radius 2 (scenario 2).

## 2 Brief overall program flow description

### 2.1 Launch parameters

Program can be launched in 3 modes (so-called generation modes):

- **manual generation**: input maps and perception scenario manually
- **semi-auto generation**: map is generated randomly, user can input the perception scenario

- **full-auto generation**: map is generated randomly, the perception scenario is pre-chosen by the user for all tests.

In the last case, one also must specify the perception scenario via commandline parameters.

```
artem@latitude:~                                                    Assignment1 (master) $ java Assignment1 param=invalid
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true -Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.GTKLookAndFeel
Argument parsing error: Parameter param not found
Usage:
 generationmode={manual, semiauto, fullauto} tests=<number of tests> scenario=<scenario number for fullauto mode> showmaps={true,false}

provides generation of test cases, either manual, semi-auto (scenario is put manually every test), or full-auto (scenario is chosen at start)
no arguments is equivalent to generationmode=manual tests=1 showmaps=true
```

Figure 1: Commandline parameters.

### 2.1.1 Data Input, Generation and Validation

The program generates the test cases randomly (or reads input from user) and validates that *obtained positions do not violate the rules of the game*, namely:

- The book and cloak **are not in perception zone** of any inspector;
- The **exit is not located** in inspectors' cells;
- Book and exit are **not in the same cell**.

### 2.1.2 Data visualization

A typical run is the following: `java Assignment1 generationmode=fullauto tests=1000 scenario=2 showmap=true` The input data is visualized in the console as follows. Agents and objects are represented by Unicode characters and may not be displayed correctly on some devices. As a backup for such situation, a sequence of steps represented as points is displayed as well.

```
Map corresponding to input:
Legend: Harry (👓) at [0,0], Filch (☹) at [4,2], Cat (😾) at [2,7], inspectors' perceptions zones are –
    book (📖) at [7,4], exit (🚪) at [1,4], cloak (🕸) at [0,8]
8   🕸  –  –  –  ▢  ▢  ▢  ▢  ▢
7   ▢  –  😈  –  ▢  ▢  ▢  ▢  ▢
6   ▢  –  –  –  ▢  ▢  ▢  ▢  ▢
5   ▢  ▢  ▢  ▢  ▢  ▢  ▢  ▢  ▢
4   ▢  🚪  –  –  –  –  –  📖  ▢
3   ▢  ▢  –  –  –  –  –  ▢  ▢
2   ▢  ▢  –  –  ☹  –  –  ▢  ▢
1   ▢  ▢  –  –  –  –  –  ▢  ▢
0   👓  ▢  –  –  –  –  –  ▢  ▢
    0  1  2  3  4  5  6  7  8
```

Figure 2: Visualization of input data.

If the generation is valid, we can proceed to searching. It is performed independently by two different algorithms - Backtracking and A*. Their implementation is described in the next section. For now, let us describe what happens after we obtain solutions.

Each solution can be represented as follows.

To reduce stdout flooding in cases with many tests, one can set `showmap=false`. Thin arrows correspond to a path from begin to a book. Bold arrows represent a path from book to exit. Directions of arrows correspond to a path direction. Please note that arrows from start to book and/or cloak may be overwritten by bold ones from book to exit.

```
======================
Used search algorithm: A_STAR
Perception scenario: 1
Outcome: Win
Number of steps to reach exit door: 16
Execution time in microseconds: 32223.6
======================


Path from start to exit:
[0,0] [1,1] [0,2] [0,3] [1,4] [2,5] [3,5] [4,5] [5,5] [6,5] [7,4] [6,5] [5,5] [4,5] [3,5] [2,5] [1,4]
Solution on a map:
8   🏚  –  –  –  ▢  ▢  ▢  ▢  ▢
7   ▢  –  👹 –  ▢  ▢  ▢  ▢  ▢
6   ▢  –  –  –  ▢  ▢  ▢  ▢  ▢
5   ▢  ▢  ⬋  ◆  ◆  ◆  ◆  ▢  ▢
4   ▢  🚪 –  –  –  –  –  📖 ▢
3   ↗  ▢  –  –  –  –  –  ▢  ▢
2   ↑  ▢  –  –  😡 –  –  ▢  ▢
1   ▢  ↖  –  –  –  –  –  ▢  ▢
0   ↗  ▢  –  –  –  –  –  ▢  ▢
    0  1  2  3  4  5  6  7  8
```

Figure 3: Report on a solution.

After a series of test, a statistics report message is printed.

```
**********************************
Statistics for search algorithm BACKTRACK, perception scenario 1:
Tests: 100      Wins: 71      Loses: 29      Win rate: 0.71
Avg. runtime (uS) (sample mean): 3598.1      Avg. # of steps (sample mean): 64.8
Runtime sample variance: 21670819.6      Steps sample variance:  783.1
======================
Statistics for search algorithm A_STAR, perception scenario 1:
Tests: 100      Wins: 71      Loses: 29      Win rate: 0.71
Avg. runtime (uS) (sample mean): 4241.4      Avg. # of steps (sample mean): 11.0
Runtime sample variance: 28307124.8      Steps sample variance:  12.0
```

Figure 4: Statistics report message.

# 3 Algorithms description

## 3.1 Backtracking

There are two notable features of our agent implementation (will be discussed bellow):

1. Building smallest possible path from multiple options;
2. Techniques of informed avoidance of adversaries, namely, keeping track of "dangerous cells" on the map which were perceived earlier. These cells are avoided as long as the agent has any better options.

According to the requirements, we should decided to let the backtracking algorithm be "mortal" in the sense that it immediately loses when enters the inspectors' perception zones.

Being an instance of uninformed search, as we will see later, the backtracking produces considerably large amount of unnecessary steps; it returns the first path ever found.

## 3.2 A* search

In our approach, we use A* as a second search algorithm. The heuristic for node n is the following:

$$h(n) = \begin{cases} 0, & \text{goal point location is unknown} \\ \infty, & \text{current point is a dead end (leads to loss)} \\ \lfloor \sqrt{(x - n.x)^2 - (y - n.y)^2)} \rfloor, & \text{goal point is} (x, y), \text{ current point is } (n.x, n.y) \end{cases} \quad (1)$$

The heuristic function covers all cases we can encounter: at the beginning, we apply the first case to find the cloak and/or book. Then, we can apply the third case. The second case assigning infinity is used to ensure we avoid cells leading to loss.

We can see that such heuristic is admissible and consistent, and thus guarantees the optimal path: indeed, since we use floored Euclidean distance or zero distance, we would never overestimate the direct distance from given node or violate a "triangle inequality" for some of successors. The second case, although assigning infinity, still keeps the heuristic admissible, since the actual path from the dead end cell doesn't exist at all and assigning infinity is more of a technical move to make our computations uniform.

## 3.3 Resulting Path - Which Way to Go?

As soon as we have cloak in our environment, a problem arises: should we pick it or not? This is not very obvious. For example, consider the example input from the assignment statement:

```
[0,0] [4,2] [2,7] [7,4] [0,8] [1,4]
1
```

```
======================
Used search algorithm: BACKTRACK
Perception scenario: 1
Outcome: Win
Number of steps to reach exit door: 33
Execution time in microseconds: 21512.3
======================


Path from start to exit:
[0,0] [1,0] [0,1] [1,1] [0,2] [1,2] [0,3] [1,3] [0,4] [1,4] [0,5] [1,5] [2,5] [3,5] [4,5] [5,5] [6,5] [7,4] [7,3] [7,2] [7,1] [7,0] [8,0] [8,1]
 [8,2] [8,3] [8,4] [7,5] [6,5] [5,5] [4,5] [3,5] [2,5] [1,4]
Solution on a map:
```

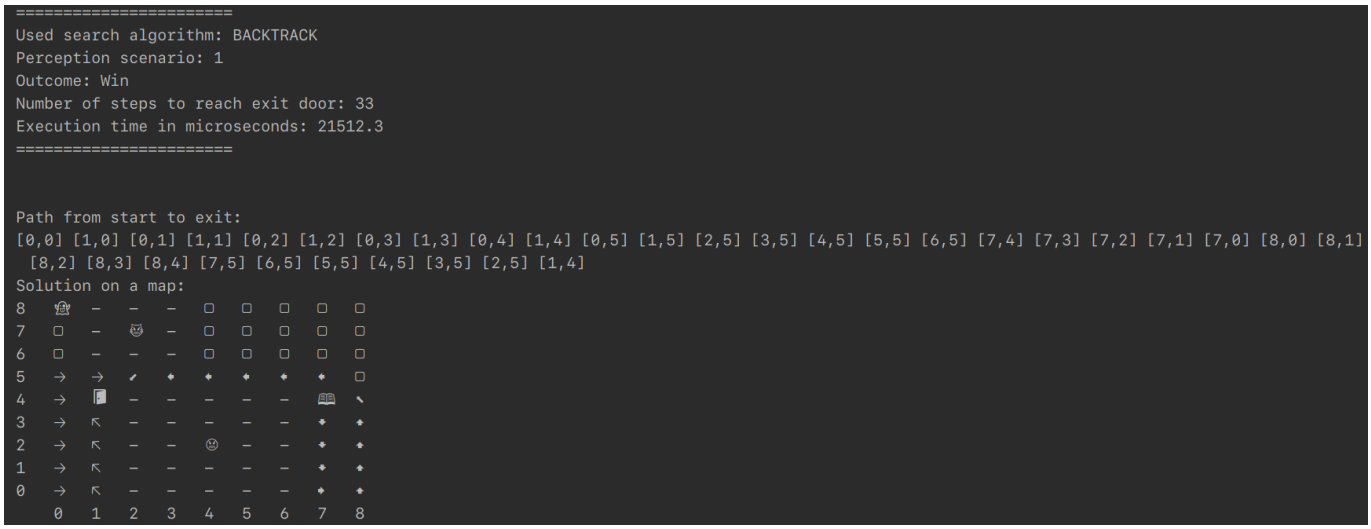

Figure 5: Solution that doesn't pick a cloak.

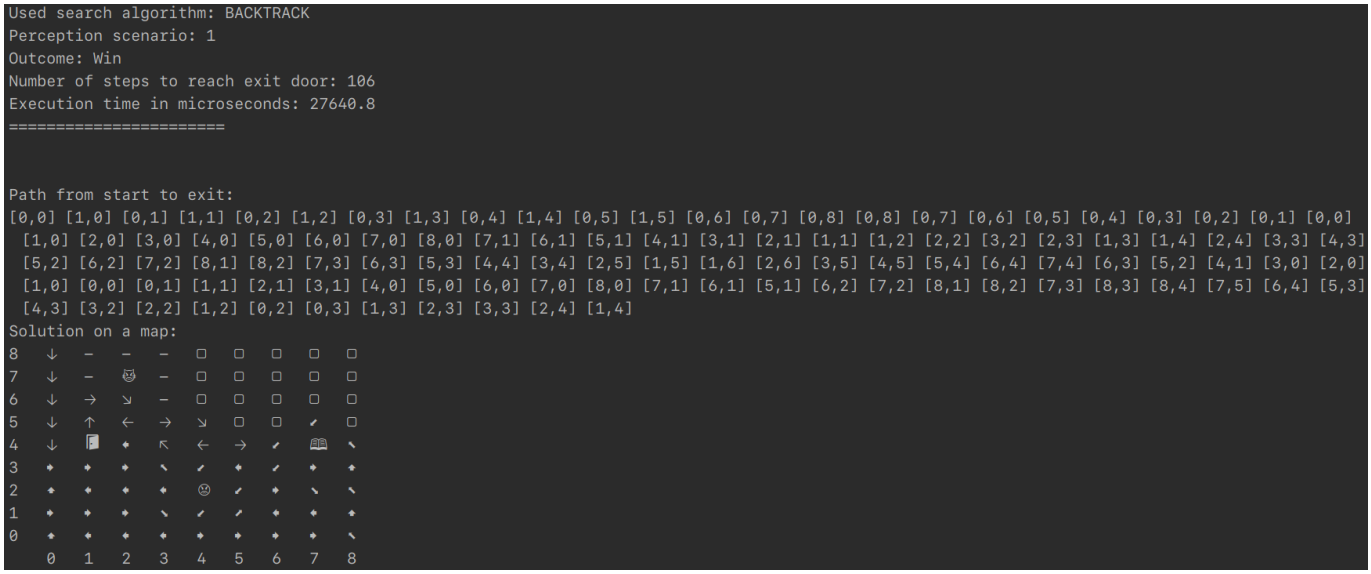


Figure 6: Solution that uses a cloak.

We see that with the cloak we end up with a path more than three times longer compared to the previous one! Thus, it is natural to introduce our path as the shortest among three possible paths:

$$path(initialState \to winState) = min\{initialState \to bookState \to exitState,$$

$$initialState \to cloakState \to bookState \to exitState,$$

$$initialState \to bookState \to cloakState \to exitState\}$$

Moreover, some of this paths may be unavailable. For example, the only way to pass map
`[0,0] [4,2] [1,6] [7,4] [0,4] [1,4]`
1

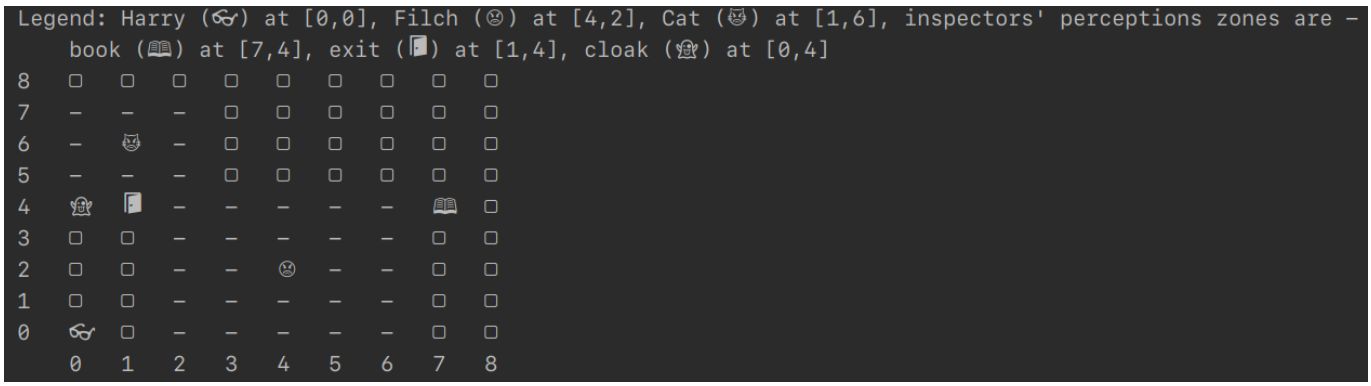


Figure 7: Map only passable by taking a cloak

is to take cloak at cell `[0,4]`:

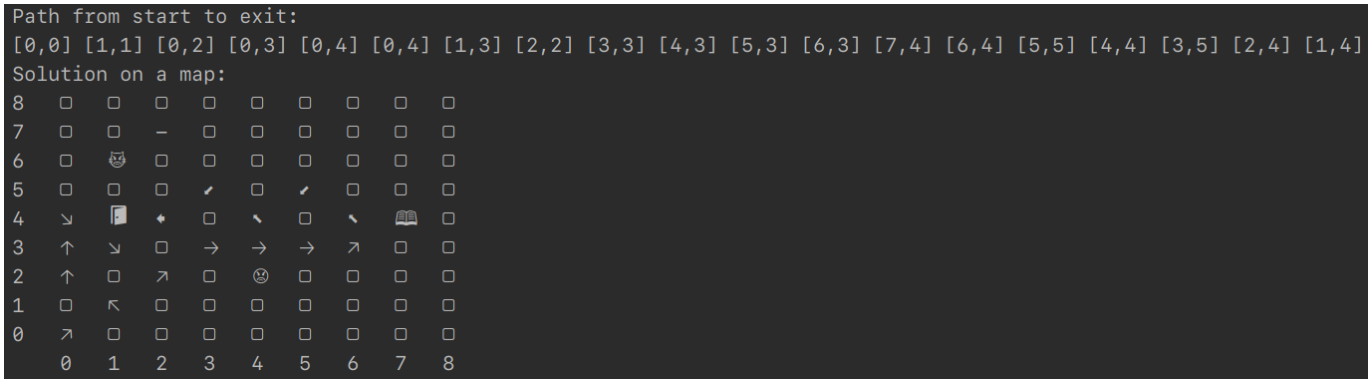


Figure 8: A* solved the map using the cloak.

# 4 Statistical Analysis

Results of tests on samples $N_1 = N_2 = 10000$ are shown below.

## 4.1 Runtime and Steps

For statistical analysis, let us formulate four pairs of hypotheses **(null and alternative ones)**:

1. - There **is no** significant difference in the **mean runtime** for backtracking and A* searches in **scenario 1**
   - There **is** a significant difference in the **mean runtime** for backtracking and A* searches in **scenario 1**
2. - There **is no** significant difference in the **mean number of steps** for backtracking and A* searches in **scenario 1**
   - There **is** a significant difference in the **mean number of steps** for backtracking and A* searches in **scenario 1**
3. - There **is no** significant difference in the **mean runtime** for backtracking and A* searches in **scenario 2**
   - There **is** a significant difference in the **mean runtime** for backtracking and A* searches in **scenario 2**
4. - There **is no** significant difference in the **mean number of steps** for backtracking and A* searches in **scenario 2**
   - There **is** a significant difference in the **mean number of steps** for backtracking and A* searches in **scenario 2**

Let us check this for first and second perception scenarios. Our solution keeps track of test results; the report contains mean values, variances and t-values for runtime and number of steps.

```
**********************************
Statistics for search algorithm BACKTRACK, perception scenario 1:
Tests: 10000               Wins: 9962             Loses: 38              Win rate: 0.99620
Avg. runtime (uS) (sample mean): 739.3         Avg. # of steps (sample mean): 65.2
Runtime sample variance: 425179.8              Steps sample variance:  899.8
======================
Statistics for search algorithm A_STAR, perception scenario 1:
Tests: 10000               Wins: 9962             Loses: 38              Win rate: 0.99620
Avg. runtime (uS) (sample mean): 691.4         Avg. # of steps (sample mean): 10.8
Runtime sample variance: 231525.6              Steps sample variance:  15.8
======================
t value for runtime: 5.896805
t value for steps: 179.629670
======================
```

Figure 9: Statistics for scenario 1 with $N = 10000$ tests.

```
**********************************
Statistics for search algorithm BACKTRACK, perception scenario 2:
Tests: 10000               Wins: 9017              Loses: 983              Win rate: 0.90170
Avg. runtime (uS) (sample mean): 830.1          Avg. # of steps (sample mean): 76.4
Runtime sample variance: 628093.1               Steps sample variance:  1328.2
======================
Statistics for search algorithm A_STAR, perception scenario 2:
Tests: 10000               Wins: 9031              Loses: 969              Win rate: 0.90310
Avg. runtime (uS) (sample mean): 725.9          Avg. # of steps (sample mean): 11.9
Runtime sample variance: 620785.5               Steps sample variance:  21.3
======================
t value for runtime: 8.855181
t value for steps: 166.515944
======================
```

Figure 10: Statistics for scenario 2 with $N = 10000$ tests.

According to this data and $1.96$ t-value for $95\%$ confidence interval, we can conclude:

- We can **reject the hypothesis of equal mean runtime** for backtracking and A* for **scenario 1**.
- We can **reject the hypothesis of equal mean number of steps** for backtracking and A* for **scenario 1**.
- We can **reject the hypothesis of equal mean runtime** for backtracking and A* for **scenario 2**.
- We can **reject the hypothesis of equal mean number of steps** for backtracking and A* for **scenario 2**.

## 4.2 Win rate

The win rate heavily depends on map generation. However, since the input constraints prohibit maps such that Harry starts in inspectors' perception zones (and thus loses immediately), occurrences of maps that have no solutions are rare (usually less than 1%, as we will see below). One example of such map is:

```
[0,0] [4,2] [1,6] [7,4] [8,8] [1,4]
1
```

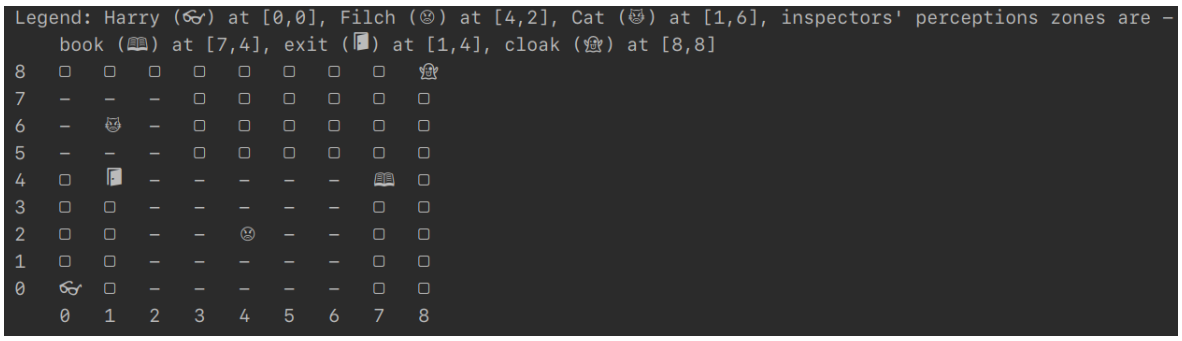


Figure 11: Map impossible to solve.

In this example, **there is no way Harry could reach a book without stepping into inspectors' perception zones** (which are lethal for him because the cloak is also out of his reach). In that case, agent loses after exhausting all safe points.

**When the map is solvable, in scenario 1 both algorithms always produce correct solution**, so their win rates are identical (and approach 100% with increasing number of tests) (see figures in Section 4.1).

A different situation occurs in scenario 2, where Harry's modified perception zone yields **"blind spots"** such that Harry can **cluelessly step into the inspector's zone**. Unfortunately, in such situation, both algorithms can be tricked with nearly same success. **About 10% of generated maps can trick the agent with perception scenario 2 in that way, so the mean winrate for scenario 2 is around 90%.** Win rates for A* and Backtracking are almost the same (differ by 1% or less); in some cases A* can perform slightly better because of its **informed search** nature (see figures in Section 4.1).

A (random-generated!) example of such situation is the following. Having a blind zone of 1 Moore neighborhood, the actor is easily tricked into stepping on $4^{th}$ row cells:

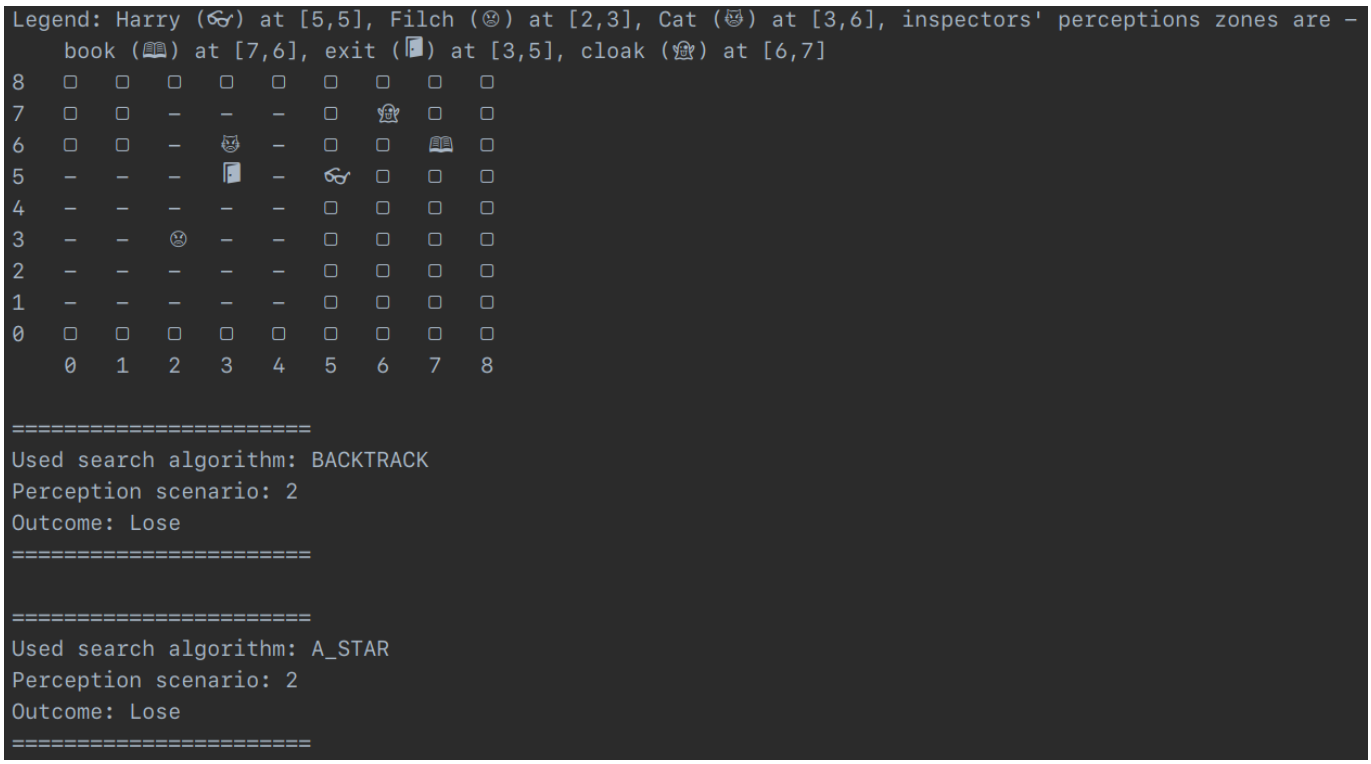


Figure 12: Map that exploits Harry's blind spots.