**Report**

Backtracking Algorithm

My implementation of backtracking algorithm works in two phases. Firtsly, the algorithm finds the shortest path from start point to the book then it finds the shortest path from the book to exit. But there are some cases when such approach can be wrong. As these paths are independent the algorithm will ignore the case when it is profitably to make a small retreat from the shortest path to book to pick up cloak that will be needed to reach exit in shorter time. That is why I launch my backtracking algorithm two times. At first, it finds simple shortest path and then it finds the shortest path that includes cloak finding before the book. At the end I compare these two path and return the shortest one.

In my variant of backtracking algorithm I do not consider all possible game trees. My algorithm stores the length of the shortest path which has already found and if the current path is longer than the shortest found path the algorithm returns from this call.

At each recursive call algorithm do the following:

1. Check whether this position is valid to step on. Coordinates should be in library space. Also Harry has ‘memory’ in the form of matrix where seen inspector’s zones and visited cells are marked.
2. Apply Harry’s vision. Algrorithm marks seen inspector’s zones to the ‘memory’ matrix.
3. Check whether the current path is longer than the shortest found. If it is so the algorithm returns from this call because there is no sence to consider this path futher.
4. Mark the current coordinate as visited in memory and check cell’s content for availability of book, exit or cloak.
5. Call backtracking for all 8 neighboring cell

Breadth-first search Algorithm

As all the movements through the cell cost identically we can adapt BFS algorithm for the shortest path finding. The algorithm works in three stages. Firstly, it computes the shortest path without cloak: start->book->exit. Then it computes the shortest path woth cloak finding before the book: start->cloak->book->exit. Finally, algorithm computes the shortest path with cloak finding after the book: start->book->cloak->exit. The answer is the shortest path among these three.

The main strucuture of the BFS algorithm was not changed. Decision for stepping into neighboring cells is based on Harry’s ‘memory’ matrix. Each step Harry check environment and writes obtained information intj the matrix. After being added to the queue the cell is checked for inspectors. If Harry steped in zone without the cloak or stumbled to the inspecotors it means that he made wrong decision and lost.

PEAS Description

**Performance meassure:** success or defeate, time of reaching the goal, whether the path is the shortest.

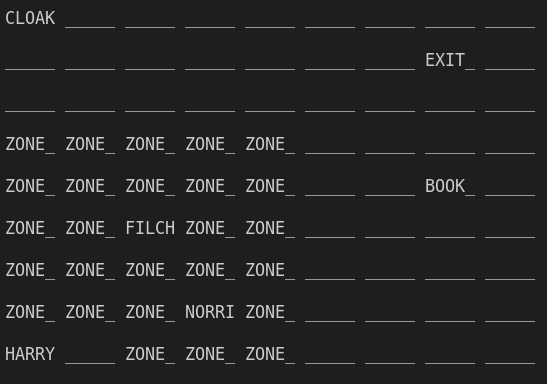
**Environment:** partially observable, single agent, deterministic, sequantial (cloal finding changes environment for Harry), static, descrete, known.

**Actuators:** Harry’s ability to travel through the cells

**Sensors:** Harry’s vision and ability to find objects in his cell

Algorithms analysis

There are some maps that are impossible to solve. For example:



If Harry is closed inside inspector’s zones intersection and can’t reach book or exit.

Statistical analysis

Backtracking algorithm works in average 13 seconds. BFS algorithm works in avrage 4 milliseconds regardless vision mode.

Both algorithms find the shortest path of the equal length with both vision modes.

With the first vision mode Harry never gets caught by inspectors.

With the second vision mode occur some cases where Harry is caught. It is caused by blind spots in Harry’s vision.

Conclusion

As a result of the work done two algorithms for shortest path finding were implemented. Backtracking algoritm works 1000 orders of magnitude longer than BFS for this task but both algorithms successfully complete the task. Second vision variant does not change much it adds cases where Harry can be caught by inspectors. Random generation of correct maps was also implemented, as well as the possibility of manual input that does not allow the creation of a map that violates the rules.