Assignment #1

Starodumov Andrey, Group 05

e-mail: a.starodumov@innopolis.university

The Goal of the Assignment

Our aim is to reach exit **Door** with taken **Book** in as minimum as possible steps. But it can happen that algorithms will find not shortest but optimal path, or they won't find a solution at all.
All possible paths (if they exists):

1. `[0,0]` → `Book` → `Door`
2. `[0,0]` → `Cloak` → `Book` → `Door`
3. `[0,0]` → `Book` → `Cloak` → `Door`

Backtracking algorithm

The backtraking algorithm (in this task) is the modified depth-first search (DFS) one. While going through the map, the algorithm, for each cell, saves the amount of steps needed to get to them. And if on a certain step this amount is less than current lenght of the cosidered path, then algorithm takes one step back (**backtracking**). Also, algorithm will backtrack if the current lenght of the cosidered path is greater than the total lenght of the last found path to the object (**Book**, **Cloack**, **Door**).

If the perception scenario is 1, the algorithm is going on and on on the known safe cells. But with the 2nd perception scenario, the algorithm can step on the unknown cell if there is no known safe one. And it can happen that the Harry Potter can be caught. In that case it is possible to find only optimal (not shortest) solution.

BFS algorithm

Breadth Frist Search algorithm is going through the map from certain cell and finds the path to the item it needs to find (**Book**, **Cloack**, **Door**). It guarantees that if it finds a path to the item, it is the shortes possible path in the considered conditions (e.g. path from book to the door with cloak).

So the algorithm moves through the all unvisited safe or possibly safe cells which are around the cell the algorithm is in.
If the perception scenario is 1, the shortest path is found if it exists.
If the perception scenario is 2, there is only optimal solution can be found (or shortest in the best cases), or the Harry Potter can be caught.

Statistical table

200 different maps were generated. Based on them, an approximate anlysis was made:

	Average time, ms	Average overall steps	Path found, times	Path not found (impossible)	Path not found (Harry is caught)
Backtraking 1st scenario	42.08	20,064	195 out of 200	5 out of 200	0 out of 200
Backtraking 2nd scenario	37.40	17,559	176 out of 200	5 out of 200	19 out of 200
BFS 1st scenario	0.45	219	195 out of 200	5 out of 200	0 out of 200
BFS 2nd scenario	0.34	184	164 out of 200	5 out of 200	31 out of 200

It has small average time because after each sequantial test it take less and less time than usually.
So usually with manual testing one by one, the BFS will give 1-9 ms, while Backtracking will give 100-200 ms

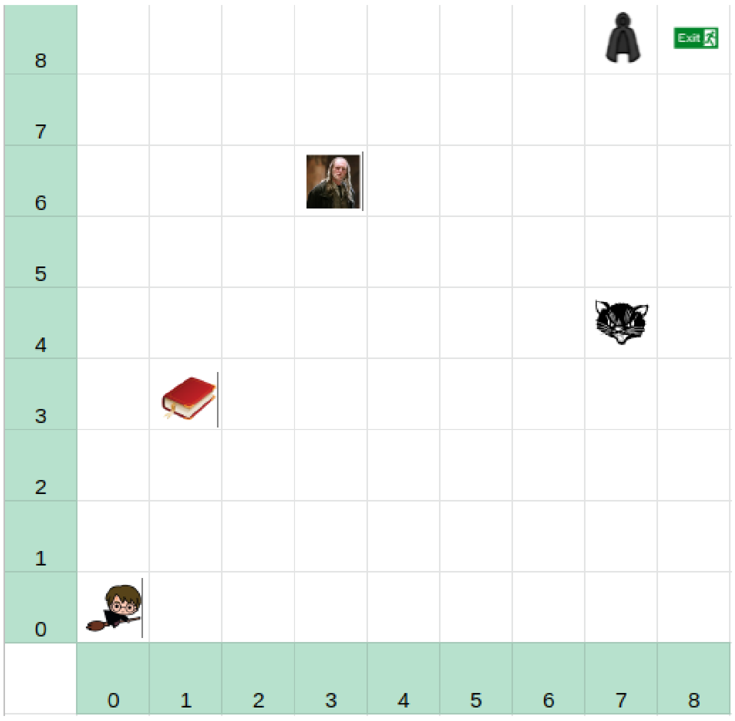
- **Bactracking 1st** is better than **Backtracking 2nd** because the found path is shortest one if it exists
- **BFS 1st** is better than **Backtracking 1st** because the found path is always the same, but **BFS 1st** takes much less time to find that path
- **Backtracking 1st** is better that **BFS 2nd** because it finds the shortest path, although it takes much more time
- **BFS 1st** is better than **Backtracking 2nd** because it finds the shortest path and takes much less time

- **Backtracking 2nd** is better than **BFS 2nd** because it finds an optimal solution in more cases of generated maps, although **BFS 2nd** finds the optimal solution taking much less time. The **BFS 2nd** has more cases when Harry Potter is caught
- **BFS 1st** is better than **BFS 2nd** because even the fact that, in average, it takes a little bit more time to find a solution, it finds the shortest path in all maps (if the path exists)

Maps on which it is impossible to find any solution-path

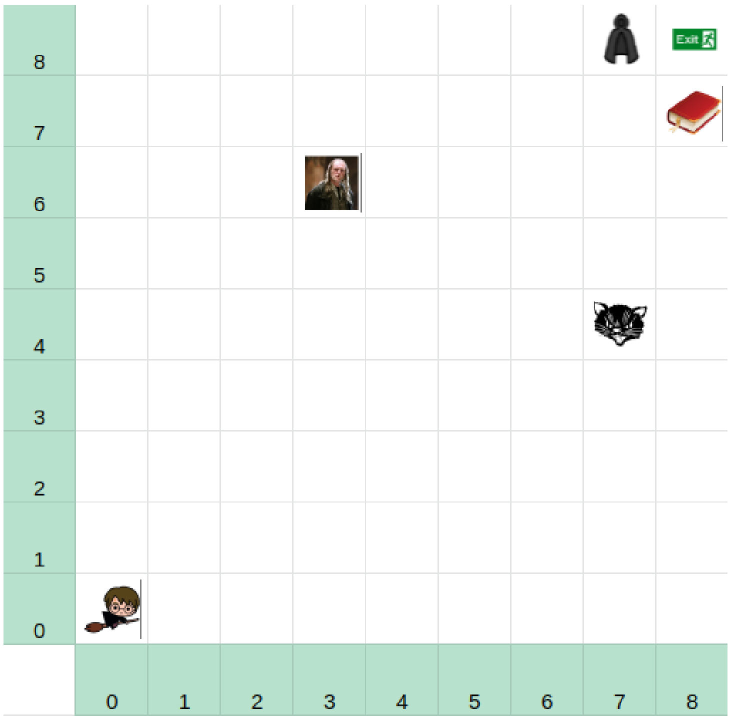
Notice: x-coordinates are vertical ones , and y-coordinates are horizontal ones

1. **Cloak and Exit are blocked by *Filch* and *Norris***



[0,0] [6,3] [4,7] [3,1] [8,7] [8,8]

2. **Cloak, Exit, and Book are blocked by *Filch* and *Norris***



[0,0] [6,3] [4,7] [7,8] [8,7] [8,8]

3. All other variations with such blocking of needed (**Book**, **Door**) items and the helpfull (**Cloak**) one. **Cloak** and at least one needed item should be blocked for the path does not exist

PEAS description with respect to the Actor (Harry Potter) agent

- Agent
  Harry Potter
- Perfomance measure
  Time spent with this algorithm to find the shortest or optimal path or to identify if there is no solution
- Environment
  1. Map
  2. Inspectors: **Filch** and **Norris**
  3. Items: **Book**, **Door**, and **Cloak**
  Properties of environment:
  a. **Partially observable** (as Harry does not see everything and can miss some things in 2nd perception scenario)
  b. **Multiple agent** (Harry is trying to avoid other agents **Filch** and **Norris**)
  c. 1st perception scenario - **Stochastic** (all seen is stable)
  2nd perception scenario - **Deterministic** (we can update uknown cells during exploration one path in Backtracking)
  d. **Sequential** (all present agent's actions define his future possible actions)
  e. Environment is **Static**
  f. **Discrete** (agent can move only inside the map through 81 cells)
  g. **Known**
- Actuators
  Harry makes a move using recursive call of a function, so it is:

1.
```
bfs_path_2(int[] queue_el, boolean cloak_on,
Characters what_to_find, ArrayList<int[]> where_to_write);
```

2.
```
backik_path(int raw, int col, boolean cloak_on,
Characters what_to_find, ArrayList<int[]> where_to_write);
```

- Sensors
  Harry uses his perception to see if there enemy on the map
  In the code the variable `i_see` is responsible for the range of the perception