Ahmed Nouralla – B19-02 – AI Assignment 1 Report.

- Source code files are included in the same archive with this report.
- **Algorithms' description:**
  - **Backtracking search** (backtracking.pl)**:**
    - The idea was to use a recursive predicate go(StepCount, Path, NextMove, Protected) that explores the (auto-generated) map, by trying all possible valid paths while counting the number of steps along the way, minimizing StepCount, keeping the state of actor (whether he is Protected from covid or not), and storing the Path list.
    - Since the number of recursive calls will be huge (even for 9*9 lattice, the unguided search is expensive and better algorithms exist for shortest path problems), some optimizations were implemented to make it faster for typical cases, but the upper-bound complexity didn't change.
    - Example of such optimizations was to try the recursive calls that are more likely to get the actor home first, by realizing the position vector from the actor current location to home and guiding the search.
  - **A* search** (astar.pl)
    - The algorithm uses the diagonal distance heuristic $\max(|actor_x - home_x|, |actor_y - home_y|)$ to determine the best next move (because the actor can move in 8 directions)
    - Since the problem is not pure shortest path, the algorithm is modified to consider 3 possible cases.
      - Actor goes around "covid" to reach home.
      - Actor goes to the doctor (while avoiding covid), then directly to home
      - Actor goes to grab the mask (while avoiding covid), then directly to home.
    - After determining the 3 potential ways for the actor to reach home, the algorithm choses a one that has the shortest overall number of steps.
- **Statistical analysis** (average running time for random maps)

<table>
<tr><th></th><th>Variant 1</th><th>Variant 2</th></tr>
<tr><td>Backtracking</td><td colspan="2">

- Execution time really depends on the map, here are the results of running 10 random maps

```
% 268,525,624 inferences, 19.578 CPU in 19.620 seconds (100% CPU, 13715595 Lips)
% 688,646 inferences, 0.031 CPU in 0.049 seconds (64% CPU, 22036672 Lips)
% 4,715,881 inferences, 0.234 CPU in 0.230 seconds (102% CPU, 20121092 Lips)
% 1,025,490 inferences, 0.125 CPU in 0.130 seconds (96% CPU, 8203920 Lips)
% 11,712,864 inferences, 1.188 CPU in 1.204 seconds (99% CPU, 9863464 Lips)
% 63,455,594 inferences, 3.938 CPU in 3.936 seconds (100% CPU, 16115706 Lips)
% 22,032,688 inferences, 1.203 CPU in 1.223 seconds (98% CPU, 18312884 Lips)
% 57,253,668 inferences, 3.984 CPU in 4.016 seconds (99% CPU, 14369548 Lips)
% 49,708,836 inferences, 3.203 CPU in 3.203 seconds (100% CPU, 15518856 Lips)
% 13,844,723 inferences, 0.719 CPU in 0.731 seconds (98% CPU, 19262223 Lips)
```

- We can see that the process is overall not very efficient, especially for hard/impossible maps.
- Notice that the process is almost always CPU intensive, due to the large number of recursive calls
- Variants won't make a difference, since backtracking will check all possibilities anyway.

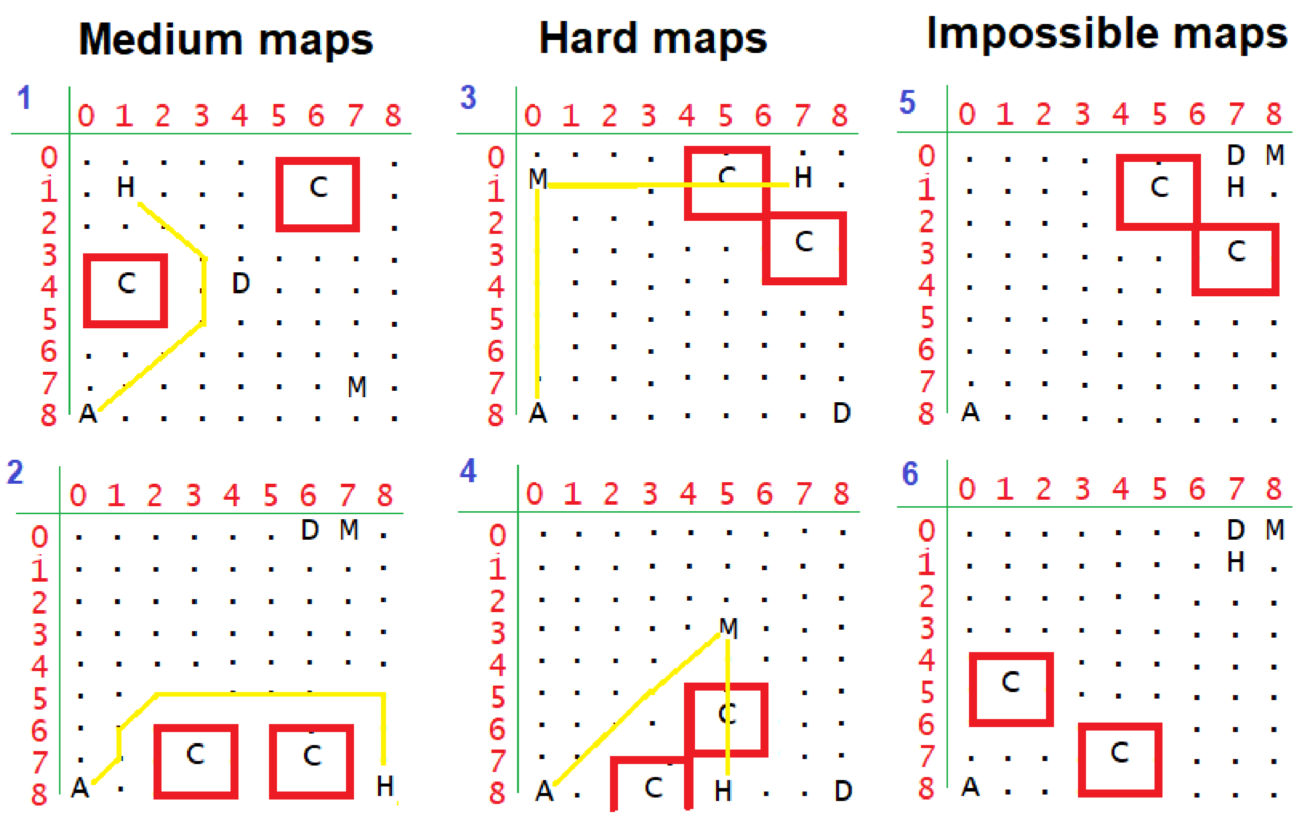
</td></tr>
<tr><td>A*</td><td colspan="2">

- Execution time is almost always very small (less than 1 second) and the process is usually not CPU-intensive, since the algorithm is efficient and the map is of dimensions 9*9. Here are the results of running 10 random maps.

```
% 34,351 inferences, 0.016 CPU in 0.015 seconds (104% CPU, 2198464 Lips)
% 32,388 inferences, 0.016 CPU in 0.019 seconds (82% CPU, 2072832 Lips)
% 33,004 inferences, 0.000 CPU in 0.017 seconds (0% CPU, Infinite Lips)
% 33,743 inferences, 0.016 CPU in 0.013 seconds (120% CPU, 2159552 Lips)
% 34,109 inferences, 0.000 CPU in 0.012 seconds (0% CPU, Infinite Lips)
% 33,463 inferences, 0.000 CPU in 0.013 seconds (0% CPU, Infinite Lips)
% 33,761 inferences, 0.000 CPU in 0.011 seconds (0% CPU, Infinite Lips)
% 32,494 inferences, 0.000 CPU in 0.012 seconds (0% CPU, Infinite Lips)
% 33,641 inferences, 0.000 CPU in 0.014 seconds (0% CPU, Infinite Lips)
% 33,564 inferences, 0.000 CPU in 0.012 seconds (0% CPU, Infinite Lips)
```

</td></tr>
</table>

- **PEAS description with respect to the actor agent.**
    - **Performance measure:** the number of steps needed to reach home, whether the actors can reach it or not.
    - **Environment:** 9*9 square lattice, representing physical spots.
    - **Actuators:** the actor can **move** (legs) horizontally, vertically, and diagonally.
    - **Sensors:** the actor can **perceive** (eyes) objects around him, from different distances.
- **Graphical representation for sample maps used for testing:**



- **Statistical analysis for custom maps:**
    - **Map 1:**
        - Backtracking: success, 11,837,908 inferences, 0.531 CPU in 0.545 seconds
        - A*: success, 31,727 inferences, 0.000 CPU in 0.005 seconds
    - **Map 2:**
        - Backtracking: success, 1,816,199,138 inferences, 128.625 CPU in 128.802 seconds
        - A*: success, 430,249 inferences, 0.125 CPU in 0.129 seconds
    - **Map 3:**
        - Backtracking: will eventually success, although taking a lot of time.
        - A*: runs into an infinite loop
    - **Map 4:**
        - Backtracking: will eventually success, although taking a lot of time.
        - A*: success, 73,289,799 inferences, 36.188 CPU in 36.305 seconds
    - **Map 5:**
        - Backtracking: terminates with answer (no path exists)
        - A*: runs into an infinite loop (no solution exists).
    - **Map 6:**
        - Backtracking: terminates quickly (38,192 inferences, 0.016 CPU in 0.017 seconds) with answer (no path exists)
        - A*: runs into an infinite loop (no solution exists).

A* with Map 3
- Although there is a solution, A* will try to consider the option of a direct way home, which doesn't exist.
- This can be easily solved by stating a time-limit for the algorithm to run before considering other options, or by explicitly checking the edge case when a non-covid element is surrounded by covid.