Assignment 1 - Report

Dmitriy Okoneshnikov, B22-CS-01

November 4, 2023

Contents

1 Algorithm flow

The general idea for both of my algorithms is following:

1. Explore the map without picking up the Shield
2. Explore the map going through the cell with the Shield
3. Compare the two distances and choose the shortest

1.1 Backtracking search

For backtracking I have used Depth-First Search (described on the lecture). Two versions of DFS were used: for exploration and for exploitation. The former uses the interactor and the latter does not use the interactor. Exploration DFS finds the first path to the goal and exploitation DFS finds the shortest path. Latter has optimizations for length of the current path and to exit the recursion if no short paths were found. All the information about the characters is stored in a map.

I start of by using the exploration DFS to put the characters onto the map and find out whether it is possible to reach the goal without the Shield. If the Stone was reached, then using the exploitation DFS I find the shortest path to it. After that I check whether the Shield was found and its cell could be reached. If so, I move to the cell with the Shield and explore the map with it. If the Stone was reached, then using the exploitation DFS I find the shortest path to the Shield and to the Stone from it. Finally, I find the shortest path among the two.

1.2 A* search

For A* I have used the algorithm described in Wikipedia (`https://en.wikipedia.org/wiki/A*_search_algorithm#Pseudocode`). The only change in the algorithm was to prevent teleportation: for each step I move back the interactor to the start and move from there to the new cell. All the information about the characters is stored in a map.

I start of by running A* to the Stone. If the Shield was found and can be reached then I run A* to the Shield and from it to the goal. Finally, I find the shortest path among the two.

2 Statistical comparison

For generating maps and running the solutions on them, a generator and an interactor scripts were written by me and tested by Matvey Korinenko. Note that these scripts were shared with other students. They can be found in the following Github Gist: `https://gist.github.com/MagicWinnie/d8899001c41c851bdc1ae9a10def8fb4`

The scripts were very helpful while debugging and counting the statistics. The generator creates n random maps following the rules stated in the problem. The interactor can be used with a solution on any language as it gets a command to run the solution with. Also it receives the directory with generated tests and the variant number. The interactor creates a CSV file with tests' file names, received answers from the solution, and the execution times. The CSV file then can be used to count the statistics manually or through other programs.

		Backtracking		A*	
		Variant 1	Variant 2	Variant 1	Variant 2
Time	Mean, s	0.289	0.290	0.049	0.061
	Mode, s	0.083	0.093	0.037	0.045
	Median, s	0.086	0.117	0.050	0.049
	σ, s	1.672	1.521	0.012	0.020
Tests	Wins	914			
	Wins, %	91.4			
	Losses	86			
	Losses, %	8.6			

Figure 1: Comparison of statistics for all algorithms

The results in table 1 were received by running the interactor on the same 1000 tests for all combinations of algorithms and perception variants. All of them gave the same answers for the tests. Therefore, number of wins, number of losses, and their percentages are equal for all columns. For counting modes the execution times were rounded to 3 digits after the decimal point.

2.1 Backtracking (variant 1) compared to A* (variant 1)

It is obvious from table 1 that A* outperforms Backtracking in everything. In mean time A* is ≈ 6 times faster, but when we look at values that are not that affected by outliers, then actually A* is faster for only ≈ 2 times. The tests on which outliers are created can be accessed in section 5. Also it is interesting to look at the standard deviations: it is much larger for Backtracking due to my implementation. A* works with almost the same time for all tests.

2.2 Backtracking (variant 2) compared to A* (variant 2)

It, basically, repeats the information given in subsection 2.1.

2.3 Backtracking (variant 1) compared to Backtracking (variant 2)

Mean, mode, and median times for variant 2 are a little bit higher compared to variant 1. This is possibly due to more information being read from the interactor and writing it to the map. Standard deviation for variant 2 is a little bit smaller and, currently, I do not know why this has happened.

2.4 A* (variant 1) compared to A* (variant 2)

It is, basically, the same as in subsection 2.3: variant 2 is a little bit slower compared to variant 1. However, the standard deviation for variant 2 is a little bit higher than in variant 1.

3 PEAS description

3.1 Performance Measure

The performance measure for Thanos would be reaching the Infinity Stone in the fastest way possible.

3.2 Environment

The environment for Thanos would be our map.

3.2.1 Properties of environment

Partially Observable. Thanos' vision is limited to his perception zone.
Multiple Agent. The agents are Thanos and Avengers. They all have sensors.
Deterministic. The next state of the map can be determined given the previous state and the action applying.
Sequential. Current decisions in our map can have consequences on future actions: picking up the Shield removes the perception zones of Hulk and Thor.
Static. When Thanos is staying still and is thinking about his actions, the environment does not change.
Discrete. We are operating with snapshots of the map and do not see, for example, Thanos smoothly moving from one cell to another.
Known. All the rules of the environment were given in the problem statement.

3.3 Actuators

Thanos can move into adjacent cells, pick up the Shield and the Infinity Stone.

3.4 Sensors

Thanos has two variants of his vision (perception zone) that allow him to perceive characters (Avengers, Shield, Infinity Stone).

4 Impossible maps

These are some examples of impossible maps found by my solution (from 1000 of total maps generated):

A	P							
P	H	P	P	P			S	
	P	P	T	P				
P	P	M	P	P				
	P	P	P					
		P						
				I				

Figure 2: Cannot exit start cell

A						P	P	P
					P	P	M	P
						P	P	P
			P	P	P		P	
			P	T	P			
			P	P	P			I
		P						
	P	H	P					
		P				S		

Figure 3: Shield and Infinity Stone are unreachable from start

A				P	T	P		
				P	P	P		
		P						S
	P	P	P			P		
P	P	M	P	P	P	H	P	
	P	P	P			P		
I		P						

Figure 4: Shield is reachable, but Infinity Stone is surrounded by Captain Marvel

5 Interesting outcomes or maps

There were some tests (14 from 1000 of total maps generated) on which my implementation of Backtracking would not fit into the time limit set on Codeforces. This happens due to the presence of large chunks of free cells on the map as the Avengers were generated close to each other and on the edges of the map. Some examples of these tests:

A		P	H	P				
			P		P	I		
				P	P	P		
			P	P	M	P	P	
				P	P	P		
					P			
				S				
P	P	P						
P	T	P						

Figure 5: Execution time is 7.655 seconds

A								
					S			
		P						
	P	P	P					
P	P	M	P	P				
	P	P	P	P	P			
	I	P	P	T	P			P
			P	P	P		P	H
								P

Figure 6: Execution time is 18.821 seconds

A			P	T	P	I		
			P	P	P			
S				P	P	P		
			P	P	M	P	P	
				P	P	P		
					P			
		P						
	P	H	P					

Figure 7: Execution time is 32.684 seconds