Report for Intro to AI Assignment I

Sviatoslav Sviatkin, CS-05

October 2023

Algorithms flow

A* Algorithm

The A* algorithm is a search algorithm employed for determining the shortest path between an initial node and a target node. Its operation relies on the utilization of a priority queue to manage the nodes that necessitate exploration. The priority queue is organized in accordance with the f-score of each node, where the f-score represents the summation of the g-score and the h-score. The g-score signifies the distance from the starting node to the current node, while the h-score is an approximation of the distance from the current node to the goal node.

The A* algorithm functions by repetitively removing the node with the lowest f-score from the priority queue and subsequently expanding it. The expansion of a node entails the inclusion of all its neighboring nodes into the priority queue, provided they are not already present. In this process, the g-score of each neighboring node is updated to reflect the distance from the start node to the current node, in addition to the cost associated with transitioning from the current node to the neighbor. The h-score of each neighbor is estimated through the utilization of a heuristic function.

The termination of the A* algorithm occurs when the goal node is successfully reached. The shortest path from the initial node to the target node can be reconstructed by tracing the parent pointers of the nodes in the priority queue.

I have made certain modifications to the algorithm; specifically, I have altered the manner in which Thanos updates the map. Given that the A* algorithm is capable of "teleportation," which is not permissible in the given task, I have devised a path that passes through the (0, 0) coordinate. This approach ensures that the entire path is thoroughly explored.

The behavior of the algorithm experiences slight variations based on alterations in the variants of the perception zone. In the second variant, Thanos will update the map more frequently within his perception zone, resulting in an increase in the time required to complete a test. However, this increase is relatively insignificant since Thanos is not endeavoring to ascertain the entirety of the path, unlike in the Backtracking method. Consequently, the map does not undergo extensive updates in this context.

Backtracking Algorithm

The backtracking algorithm is a versatile problem-solving approach that systematically explores all potential solutions through recursive iterations. This method involves maintaining a record of a partial solution and incrementally extending it step by step. When an extended solution is determined to be unviable, the backtracking algorithm retraces its steps to the preceding partial solution and endeavors to explore an alternative extension.

The backtracking algorithm can be effectively employed to address the task of discovering a path from an initial node to a target node within an uncharted terrain. In this context, the partial solution consists of a sequence of nodes that guides from the starting point to the current node. The algorithm proceeds to expand the partial solution by seeking to incorporate a new node into the sequence. However, if the newly considered node falls within the observation range of an adversary, the algorithm initiates a backtrack operation and explores alternative extensions.

The termination of the backtracking algorithm transpires upon reaching the goal node. Subsequently, the shortest path from the starting node to the target node can be reconstructed by reversing the sequence of nodes within the partial solution.

The Depth-First Search (DFS) algorithm operates through the recursive exploration of all child nodes associated with a given node before advancing to the next node in the sequence. This iterative process persists until all nodes within the graph have been exhaustively examined. Notably, DFS represents a specialized case of the Backtracking algorithm, making it adaptable with slight modifications. To this end, I have adapted the DFS method in a manner that incorporates the continuous updating of the map by Thanos, which includes marking nodes as either vacant or obstructed.

It is worth noting that the algorithm's behavior experiences minimal variations in response to adjustments in the perception zone variants. In the second variant, Thanos engages in more frequent map updates within his observation range, leading to an increment in the time required for test completion.

Statistical Comparison

-	A* (var1)	A* (var2)	Backtracking (var1)	Backtracking (var2)
Mean (Time)	0.1448s	0.14529s	0.20935s	0.21017s
Mode (Time)	0.11361s	0.11699s	0.11529s	0.16894s
Median (Time)	0.1127s	0.11218s	0.11737s	0.12187s
STDev (Time)	0.04411s	0.03671s	0.09473s	0.09123s
Mean (Wins)	905	905	911	911
Mode (Time)	905	905	911	911
Median (Wins)	905	905	911	911
STDev (Wins)	0	0	0	0
Mean (Loses)	95	95	89	89
Mode (Time)	95	95	89	89
Median (Loses)	95	95	89	89
STDev (Loses)	0	0	0	0

PEAS Description with Respect to the Thanos Agent

PEAS :

Performance Measure: The performance measure for the actor agent is the length of the shortest path from the (0, 0) to the Infinite Stone.

Environment: The environment is an unknown map with Thanos, the Avengers, and the Infinity Stone.

Enviroment properties :

- Partially observable
- Single agent
- Deterministic
- Sequential

- Dynamic
- Discrete
- Known

Actuators: The actuator of the actor agent is interacting system with move outputs

Sensors: The sensors of the actor agent are the perception zones of Thanos.

Actor Agent: The actor agent is Thanos.

Other notes

Impossible maps

Some maps are impossible for my code to solve. For example :

```
A . . . . . . . .
. . . . . . . . .
. . P P P . . . .
. . P T P . . . .
. . P P P . . P .
. . P P P . P H P
. P P M P P . P .
. . P P P . . S .
. . . P I . . . .
```

This map is unsolvable because Thanos can't reach Stone by any way.

I need to mention that some maps like this :

```
A . . . . . . . .
S . . . . . . . .
. . P P P . . . .
. . P T P . . . .
. . P P P . . P .
. . P P P . P H P
. P P M P P . P I
. . P P P . . . .
. . . P . . . . .
```

are unsolvable for me too, because I didn't figure how to use shield properly. If I add any idea I have, codeforces tests are failing. I can add that Karim sent one test to Philosophy chat in Telegram and there were NO using of shield. It will be less moves to take stone using shield, but answer is like Thanos is just ignoring it.

3

Time: 62 ms, **memory**: 548 KB

Verdict: WRONG_ANSWER

Input

```
1
A....P...
S...PPP..
...PPMPP.
....PPP..
P....P.I.
HP.......
PPP......
PTP......
PPP......
```

Participant's output

```
100002
```

Jury's answer

```
13
```

Checker comment

```
wrong answer expected '13', found '100002'
```