Evsey Antonovich

Algorithm Flow

A* Algorithm flow: First, the algorithm initializes all tiles of the grid. Then we start at the agent's tile, setting its g cost to 0 and its h cost to the heuristic function (diagonal distance to the goal). Each iteration, we first pick the tile in open that has the lowest f cost, and if we have several of those, we pick the one with the lowest h cost, and if we have several of THOSE, then we pick the one with the lowest Manhattan distance to the goal, just to get a nicer looking path. Afterwards, we calculate either the diagonal distance to the kraken and check that it equals 1, in the first perception scenario, or we calculate the Manhattan distance to the Kraken, and check that it is smaller or equal than 2. If this is true, then we mark the kraken as discovered. Next, we check if we are on the goal tile or not. If we are, we traverse the tile parents to the start, creating a path out of the list of points and returning it. Otherwise, we continue and add all the neighbors to open (if they are not in the closed set), and update their g/h costs, setting g cost to the cost of the current tile + 1 and changing their parent, if the new g is smaller than that tile's current g cost, and setting h cost to the diagonal distance between the tile and the goal. Afterwards, we run this iteration again. If we run out of tiles in open, then we return null, as there is no path to the goal. To get the actual shortest path from the location of the Captain to the Dead Man's Chest, first we try to get the path directly from the start to the chest, and then we get four composite paths, which consist of a path to Tortuga, a path from Tortuga to one of the corners near the Kraken, and one of the corners of the Kraken to the chest. Out of these four composite paths (if they exist), we take the shortest one, and if it's shorter than the normal path (or if you can't go to the chest without killing the Kraken), we return this path instead.

Backtracking Algorithm flow: First, we introduce a GameState class, which stores the current location, goal, the tiles visited before reaching Tortuga, the tiles visited after reaching Tortuga, if we visited Tortuga or not, whether the Kraken is alive or not, and whether or not we discovered the Krane, Davy Jones and the Rock, alongside the parent GameState and current iteration. Alongside that, we keep track of the current shortest path in the algorithm, and the shortest iteration number for each of the tile before and after we reach Tortuga. First, we initialize the initial GameState, which starts at the captain's location, with an iteration value of 1. Then, at the beginning of the iteration, we immediately check if we are at the Dead Man's Chest. If we are, we create a path by traversing up the GameState tree through the parent value, and compare this path to the currently stored shortest path, and overwrite it if this new path is shorter or if no shortest path is present. If we aren't on the chest point, and a shortest path already exists, we check if our iteration number is higher or equal to the shortest path length. If so, that means we will not reach a shorter path by going further, and so we prune this branch. Afterwards, we check if our iteration number is higher than 25, and also prune this branch in that case, as there are no cases where that many iterations would be required to reach the end. Next, we check if we are on Tortuga Island, and if so, we mark Tortuga Island as visited, and change our goal from Tortuga Island to the chest. Next, we check if our iteration is not higher than the shortest iteration for the current tile visited (either before or after reaching Tortuga Island). If it's higher than the shortest iteration for the current tile, then we also prune this branch, because we can reach this tile in a shorter path, and therefore our current branch will obviously not lead to the shortest path. Next, we check if we have visited this cell already during our before/after Tortuga half. If we have, then we prune this branch, as we have no reason to visit a cell twice, unless we reached Tortuga Island, in which case we *can* visit a cell again. Next, we check if we are on a corner near the Kraken and if we have visited Tortuga Island, and if both of these conditions are true, we mark the kraken as dead. Lastly, we get all the neighbors of this cell we can visit, and add all the neighbors that we have not visited yet to the priority queue of next possible states, duplicating our current state but increasing the iteration number and changing the state location. If we already have a shortest path, then we make sure to not add neighbors that would be further away from the chest than the distance of this shortest path, as that would not lead us to a better path. Lastly, while we have possible states in the next possible states priority queue, we poll a state from the queue and recursively call this function on that state, giving priority to states that are closer to the Dead Man's Chest or the Tortuga Island, based on the current state's goal, which changes from the Tortuga Island to the chest after visiting Tortuga Island (basic heuristics). After our recursive call tree ends, we simply return the shortest path stored, and if there was no path at all, we return null.

Statistical Analysis

	Backtracking	A*	Backtracking	A*
Scenario Number	1		2	
Execution Time Mean (ms)	1.1016	0.0139	1.0907	0.0134
Execution Time Mode (ms)	0.0008	0.0032	0.0002	0.0021
Execution Time Median (ms)	0.3865	0.0041	0.3738	0.0034
Execution Time Standard Deviation (ms)	1.8868	0.0454	1.8627	0.0258
Number of wins	988			
Number of losses	12			
Percentage of wins	98.8			
Percentage of losses	1.2			

Statistical Analysis Report generated on Desktop PC with an AMD Ryzen 5 5600X 6-Core processor system, n=1000

Evsey Antonovich

PEAS description

Agent Type: Actor

Performance Measure: Distance to Dead Man's Chest

Environment: The Sea

Actuators:

- Moving the Black Pearl/itself
- Taking rum casks from Tortuga Island
- Blowing up the Kraken with rum casks

Sensors:

- The Compass to see Tortuga Island location
- The Compass to see Dead Man's Chest location
- Spyglass to see nearby perception zones
- Super Spyglass to see nearby perception zones

Environment Properties:

- **Partially Observable,** as the Spyglass cannot see the entire Environment at once.
- **Single Agent,** as the enemies are really just static hazards that cannot interact too much with the Environment and there is only one Jack Sparrow.
- **Deterministic,** as only the actions of our Agent can influence the Environment in any way, and we can fully predict how the Environment will change because of those actions.
- **Sequential,** as picking up the Tortuga has an influence on future actions, as in allowing us to fire the rum casks at the Kraken.
- **Static,** as the Environment does not change much unless the Agent does some specific actions.
- **Discrete,** as the Environment changes only in turns, when the Agent moves.
- **Known,** as the Agent knows the rules of the Environment.

Evsey Antonovich

Unsolvable maps

An assortment of randomly generated maps that are impossible to solve: