AI Project 1 Report

Team Members

1. **Ahmed ElAmory 46-2859**
2. **Ahmed Belal 46-0642**
3. **Mohamed Sherif 46-12029**

Problem Description

You are a member of the coast guard force in charge of a rescue boat that goes into the sea to rescue other sinking ships. When rescuing a ship, you need to rescue any living people on it and to retrieve its black box after there are no more passengers thereon to rescue. If a ship sinks completely, it becomes a wreck and you still have to retrieve the black box before it is damaged. Each ship loses one passenger every time step. Additionally, each black box incurs an additional damage point every time step once the ship becomes a wreck. One time step is counted every time an action is performed. You reach your goal when there are no living passengers who are not rescued, there are no undamaged boxes which have not been retrieved, and the rescue boat is not carrying any passengers. You would also like to rescue as many people as possible and retrieve as many black boxes as possible. The area in the sea that you can navigate is an m x n grid of cells where 5 <= m; n <= 15.

The grid elements are restricted to the following.

- **Coast Guard Boat**: The coast guard boat is the agent; it is the only element that can move on the grid. The coast guard can enter any cell. It has a fixed capacity 30 <= c <=> 100 which indicates the number of passengers it can carry at one time. So it may have to make multiple visits to one ship to save all the passengers on it.
- **Ships**: Each ship has a certain number of passengers and, at every time step, one of them expires. The ship is considered sunk when all of them are expired or all are picked up In that case, it becomes a wreck. Each ship contains a black box that can be retrieved after it sinks as long as it has not been completely damaged.
- **Wrecks**: Once the ship no longer has any passengers (all expire or all are picked up), it becomes a wreck. When the ship becomes a wreck (once its last passenger either expires or is picked up), in the next time step, the black box starts counting damage from 1 all the way up to 20. Once damage reaches 20, the black box is no longer retrievable.
- **Stations**: Stations are fixed and do not have any capacity limits. To count a passenger saved, they need to be dropped at a station. Initially, the grid is configured as follows.
- The coast guard boat is at a random location
- Several ships are scattered at random locations, and each has a random initial number of passengers p, where 0 < p <= 100.
- Several stations are at random locations.
- There are no wrecks.
- No two items are in the same cell.

The coast guard can perform the following actions.

- **Pick-up**: The coast guard picks up as many passengers off a ship as its remaining capacity allows. This can be done on a ship that is in the same cell as the coast guard and it only affects this ship. Once a passenger is picked up by the coast guard, they will not expire and will stay on the coast guard boat until they are dropped at a station.
- **Drop**: The coast guard drops all passengers it is currently carrying at a station. This can only be done when the coast guard and the station are in the same cell and it resets the remaining capacity of the coast guard boat to 0.
- **Retrieve**: The coast guard boat retrieves a black box. This can only be done when the coast guard boat is in the same cell as a wreck with a black box which has not been completely damaged yet. This action does not affect the coast guard's remaining capacity at all.
- Movement in any of the 4 directions (up, down, left, right) within the grid boundaries.

Problem Discussion

- The problem is a search problem, we have to find the best path to reach the goal state.
- The problem is a fully observable problem, the agent can see the whole state of the problem.
- The problem is a single agent problem, we have to find the best path for one agent to reach the goal state.
- The agent is the coast guard boat.

To takle the problem we need:-

- First we need to represent the state of the world including the agent position (i.e. The coast guard boat) at each step.
- In our problem, the state is represented by the position of the coast guard boat in the grid, the number of people on the coast guard boat and the state of the each ship in the grid (i.e. the number of alive people on the ship, is it wrecked and the damage of the blackbox).
- At every step, the agent chooses an action to do and the state changes accordingly until we reach a goal state.
- The goal state is when all the people on the ships are rescued or dead and all blackboxes are retrieved or damaged and can not be retrieved.
- We need to reach the goal state with the minimum deaths and maximum number of blackboxes retrieved.

Problem Solution

Search-Tree Node Abstract Data Type.

To represent our state we created a Node ADT that contains the following attributes:

- **parent**: The parent node of this node.
- **action**: The action that was performed to reach this node from the parent node.
- **depth**: The depth of this node in the search tree.
- **left**: The result state (node) of the acion left of this node.
- **right**: The result state (node) of the acion right of this node.
- **up**: The result state (node) of the acion up of this node.
- **down**: The result state (node) of the acion down of this node.
- **pickUp**: The result state (node) of the acion pickUp of this node.

- **dropoff**: The result state (node) of the acion drop of this node.
- **retrieve**: The result state (node) of the acion retrieve of this node.
- **x**: The x coordinate of the coast guard boat in the grid.
- **y**: The y coordinate of the coast guard boat in the grid.
- **numberOfPeopleOntheCoastGuard**: The number of people on the coast guard boat.
- **ships

Search Problem Abstract Data Type.

CoastGuard Problem

Main Function Implemented

Algorithms Implementation

Breadth First Search

Depth First Search

Iterative Deepening Search

Uniform Cost Search

Greedy Best First Search

A* Search

Heuristics used in Greedy Best First Search and A* Search

Heuristic 1

Heuristic 2

Results

Breadth First Search

Optimality

Breadth First Search is optimal because it expands the shallowest nodes first.

Completeness

Breadth First Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

Depth First Search

Optimality

Depth First Search is not optimal because it expands the deepest nodes first.

Completeness

Depth First Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

Iterative Deepening Search

Optimality

Iterative Deepening Search is optimal because it expands the shallowest nodes first.

Completeness

Iterative Deepening Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

Uniform Cost Search

Optimality

Uniform Cost Search is optimal because it expands the nodes with the lowest cost first.

Completeness

Uniform Cost Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

Greedy Best First Search

Optimality

Greedy Best First Search is not optimal because it expands the nodes with the lowest heuristic value first.

Completeness

Greedy Best First Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

A* Search

Optimality

A* Search is optimal because it takes into consideration the cost of the path and the heuristic value. It expands the nodes with the lowest cost + heuristic value first.

Completeness

A* Search is complete because it expands all the nodes in the search tree.

RAM Usage

CPU Utilization

Number of Expanded Nodes

Comments

Conclusion