# Coast Guard Search Problem

16.12.2022

Omar Aref 46-5432

Shawky Haitham 46-3502

Hana Adel 46-0377

## Problem Discussion

Being in charge of a rescue boat that enters the water to save other sinking ships puts you in the position of a coast guard officer. As soon as there are no more passengers on board to save, you must save everyone who is still alive aboard the ship and then find its black box. You must still get the black box out of a sinking ship before it sustains damage because it will now be a wreck.

## Tree Node

The Tree Node data structure contains the elements present in the lecture. This includes the following variables:

1. A Tree Node Parent
2. The operator applied to reach this Node
3. The State which this Node holds
4. The Depth of the Node in the Tree
5. The Path Cost of the Node

## Search Problem

The Search problem is the class that models the problem as a search problem and includes the following variables:

1. An arraylist of strings of operators
2. An initial State
3. A state space

It also has some methods that should be defined in the Coast Guard and any other problems that extends the search problem which are:

1. The Goal Test which defines if the state is a goal state
2. A Path Cost which takes an array of actions and compute their cost
3. The expand function which takes an operator and a state and returns a new state

## Coast Guard Problem

The Coast Guard problem is a subclass of the search problem abstract class; this means it has to define the functions above.

The goal test is defined as if there are no passengers left in the ships and no passengers left with the coast guard.

The path cost is defined as the number of deaths and number of retrieved black boxes.

The expand function moves the agent if the operator is moving in one of the four directions and retrieves black boxes or pickup or drop passengers. This returns a new state if it satisfies some of the preconditions of the action.

## Search Techniques

First there was an abstract class called QueuingFunction that has an abstract method called queuingFunction and all search techniques inherit from this class.

The queuingFunction method takes a Node and a NodesList and returns the NodesList after inserting the node in it.

Each search strategy implements it in its own way of insertion, for example BFS puts the node at the end of the list while DFS puts it in front.

The different search strategies are implemented as present in the lecture.

## Heuristic Functions

1. We could estimate the total number of deaths by counting how many passengers we have in total and the total number of ships and decrease them by how many we estimate to die by dividing them by the Coast guard capacity. This leads to goal states having a zero cost in deaths since the passengers left on ships are zero.
2. Same as above however we take into consideration the number of retrieved black boxes..

## Performance Comparison

We test the performance using the same grid test string which is `"7,5;40;2,3;3,6;1,1,10,4,5,90;".` Which produced these results below

| | BF | DF | ID | GR1 | GR2 | AS1 | AS2 |
|---|---|---|---|---|---|---|---|
| RAM(MB) | 319 | 6 | 340 | 16 | 13 | 13 | 13 |
| Time(S) | 25 | 0.499 | 109 | 0.253 | 0.232 | 0.184 | 0.2 |
| Nodes | 779 | 79 | 5533 | 73 | 73 | 65 | 65 |

We can notice that these results match the findings of from the lectures since we find that the A* search techniques produced the least number of expanded nodes and had the best time complexity.