COMP30024 PROJECT 1 PART A REPORT

With reference to the lectures, which search strategy did you use? Discuss implementation details, including choice of relevant data structures, and analyse the time/space complexity of your solution.

Our approach employed a combination of the Breadth First Search (BFS) and Depth First Search (DFS) search strategies. The overall strategy was to find all possible cells the red frog could have moved in one action and repeat until the red frog has reached the final row and then backtracking.

This process involved looping through all neighbouring cells via the use of a collection of direction vectors. There are three possible outcomes of the neighbouring state:

1. The cell does not contain a lilypad: invalid move, so ignore this cell.
2. The cell contains a lily pad: valid move, so append this cell to the queue.
3. The cell contains a blue frog: check if there is a lily pad directly behind the blue frog in the same direction. If so, the red frog can jump over it.

While the BFS implementation handles standard single-step moves, the mechanism of chaining jumps sees it necessary to employ an additional DFS implementation. Since each jump in the chain could have been reached by the red frog in one move, we must recursively exhaust the possible jump path and append each jump to the queue.

When reaching the end of the board, we have to backtrack our path to find the optimal solution. We save the parent node and the direction to dictionaries with the key being the value of the node currently being explored. This way, we can simultaneously backtrack through the parent dictionary to track the path to the start, along with the directions used to get to each new node.

The time complexity of this approach should be O(b^d), where b = branching factor, d = longest possible path until the red frog reaches the end row. Commencing from the starting node, each valid neighbouring cell is appended to the queue, and this process is repeated for each of the neighbouring cells. This follows an exponential trend where each level multiplies by at most b cells, until eventually the cell reaches the last row of the board. Given the constraints of this game, we can narrow our definition of b to be:

$$0 <= b <= 5$$

This is because there are a maximum of five directions in our direction vector, so there can only be at most 5 valid neighbours for each cell.

The space complexity should be O(b^d) as in the worst case, every cell on the board is stored in the queue. The extra storage of the parent dictionary does not change the scale of the space complexity as O(nb^d) converges to O(b^d) where n is an integer.

If you used a heuristic as part of your approach, clearly state how it is computed and show that it speeds up the search (in general) without compromising optimality. If you did not use a heuristic based approach, justify this choice.

Our approach did not use a heuristic as we employed a general Breadth First Search (BFS). While a heuristic could potentially improve the efficiency of our search, such an implementation would have required unnecessarily complex computation and design for a seemingly simple representation of the game.

We acknowledge that as we move forward to more complicated representations of the project (ie. Part B), a consideration of heuristics and more efficient approaches would be wise, however since the constraints of Part A were finite and relatively small, BFS seemed to be a reliable choice that embraced the simplicity of the problem whilst guaranteeing optimality.

Imagine that all six Red frogs had to be moved to the other side of the board to secure a win (not just one). Discuss how this impacts the nature of the search problem, and how your team's solution would need to be modified in order to accommodate it.

Adding six red frogs to the design would significantly complicate the implementation of the project.

It seems the biggest change of the new implications would be the introduction of a new dynamic in which red frogs are now allowed to interact with each other. These interactions may be positive or negative as new strategies and possibilities are introduced at the expense of complications and caveats.

Accordingly, the nature of the search problem must be modified to account for the moves of the other red frogs. For example, some strategies that could be introduced may include:

a. Potential to link up chain links
b. Moving forward as a pack to prevent any red frogs from having to traverse without being able to jump over other frogs

Some caveats to be wary of:

a. Avoid blocking other red frogs (ie. Can unintentionally block another red frog by stealing their lilypad).
b. By extension, avoid moving to a cell that blocks a jump chain formed by other red frogs.
c. Manoeuvring the frogs to be spread out on the end lily pads.

A solution to such a problem would involve introducing a metric to consider future implications of each move on other red frogs. For example, if a red frog moves to this particular square, does that then allow another frog to reduce the amount of moves they require to reach the end row by utilising a jump chain? This could be included in the heuristic as we move to a more A* search aligned solution.