COMP30024 – Project A report

Team members: Connor Cates, Jad Saliba

**How have you formulated the game as a search problem? (You could discuss how you view the problem in terms of states, actions, goal tests, and path costs, for example.)**

States: each move the white pieces make is a different board state

Actions: move up, down, left, right with a move distance ranging between 1 and the number of white pieces in the stack while constantly checking if the move is valid and does not go through any black pieces or move outside the board – Stacking on other white pieces is obviously allowed. Each move is also saved in a visited list as to not revisit the same node more than once.

Goal tests:check if the board state has no more black pieces left.

Path costs: each move costs 1.

**What search algorithm does your program use to solve this problem, and why did you choose this algorithm? (You could comment on the algorithm’s efficiency, completeness, and optimality. You could explain any heuristics you may have developed to inform your search, including commenting on their admissibility.)**

Our program uses a greedy search algorithm to solve the problem as it is an efficient way of not simulating every possible move. It first finds a list of target locations that have neighboring black pieces (sorted by the number of neighboring black pieces) and then uses a Manhattan distance heuristic to sort the white pieces to each the target location.

This search is complete in finite space with repeated state checking but is not optimal.

**What features of the problem and your program’s input impact your program’s time and space requirements? (You might discuss the branching factor and depth of your search tree, and explain any other features of the input which affect the time and space complexity of your algorithm.)**

Expected time requirement:

Expected space requirement: