Eli Werstler - Project Part A Report

We opted to utilize the A* search algorithm for the first part of the project. We felt this was optimal amongst our options due to its efficiency and effectiveness in pathfinding and graph traversal problems, as covered in lectures. A* search is particularly suitable for this problem due to its ability to find the shortest path to the goal state using an informed search strategy. The core data structures used in the implementation include dictionaries and priority queues. We used dictionaries to map Coord objects to PlayerColor enums and ‘heapq’ priority queues for managing the search frontier. The priority queue stores nodes in the order of their estimated total cost—actual cost from the start node plus the heuristic estimate—so that hopefully the most promising nodes are explored first. The time complexity of this search process is O(b^d), where ‘b’ is the average number of successors per state, and ‘d’ is the depth of the shortest path. However, due to the nature of the game, ‘b’ can be quite large due to the numerous potential placements of each shape. The size of ‘d’ can be variable as well, depending on the initial board setup and the target. The use of the heuristic function helps to reduce the explored search space, although the worst-case time complexity can still be quite large. The same is true for the space complexity, which is also O(b^d), because the algorithm stores all generated nodes in the priority queue and explored set. This also depends on the efficiency of the heuristic.

Our heuristic function is designed to estimate the number of moves required to reach the target based on the number of empty spaces in the target row or column. It simplifie to focusing on the row or column closer to completion and assumes an average efficiency of tetromino placements affecting 2-3 spaces towards completing the line. The heuristic is designed to not overestimate the true cost to reach the goal and ensures that the cost estimate to the goal decreases along a path. By estimating moves based on the concentration of empty spaces and the average impact of the tetromino, it guides the search towards more promising board states. This speeds up the search by reducing the exploration of less optimal paths without sacrificing optimality.

In a scenario where all Blue tokens must be removed would increase the search’s time and space complexity. The expanded goal makes the solution space much bigger and also increases both the branching factor and solution depth, given the multifaceted approach needed for comprehensive Blue token elimination. The search space would be enlarged greatly, likely requiring different strategies for heuristics to maintain problem solvability without using excess resources. Specifically we would likely have to incorporate a more complex strategy factoring in the distribution of blue tokens, potential for simultaneous line completions, and strategic placements for maximal board impact. Furthermore, our successor function would have to be modified to prioritize placements conducive to multiline completions or strategic board clearing.