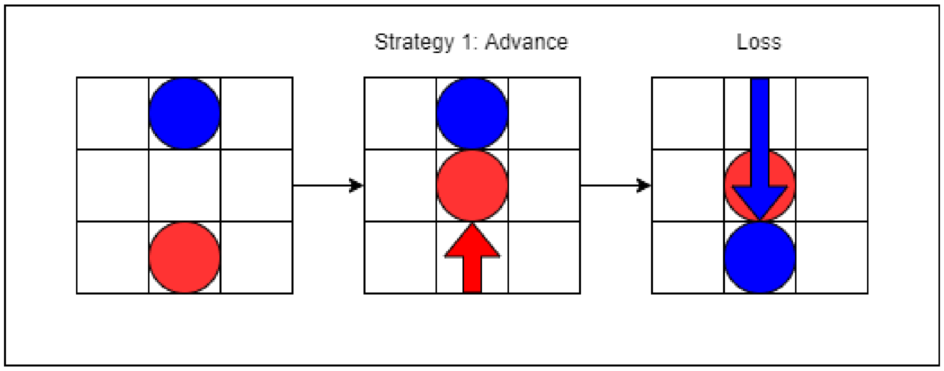
Combinatorics Project Final Report

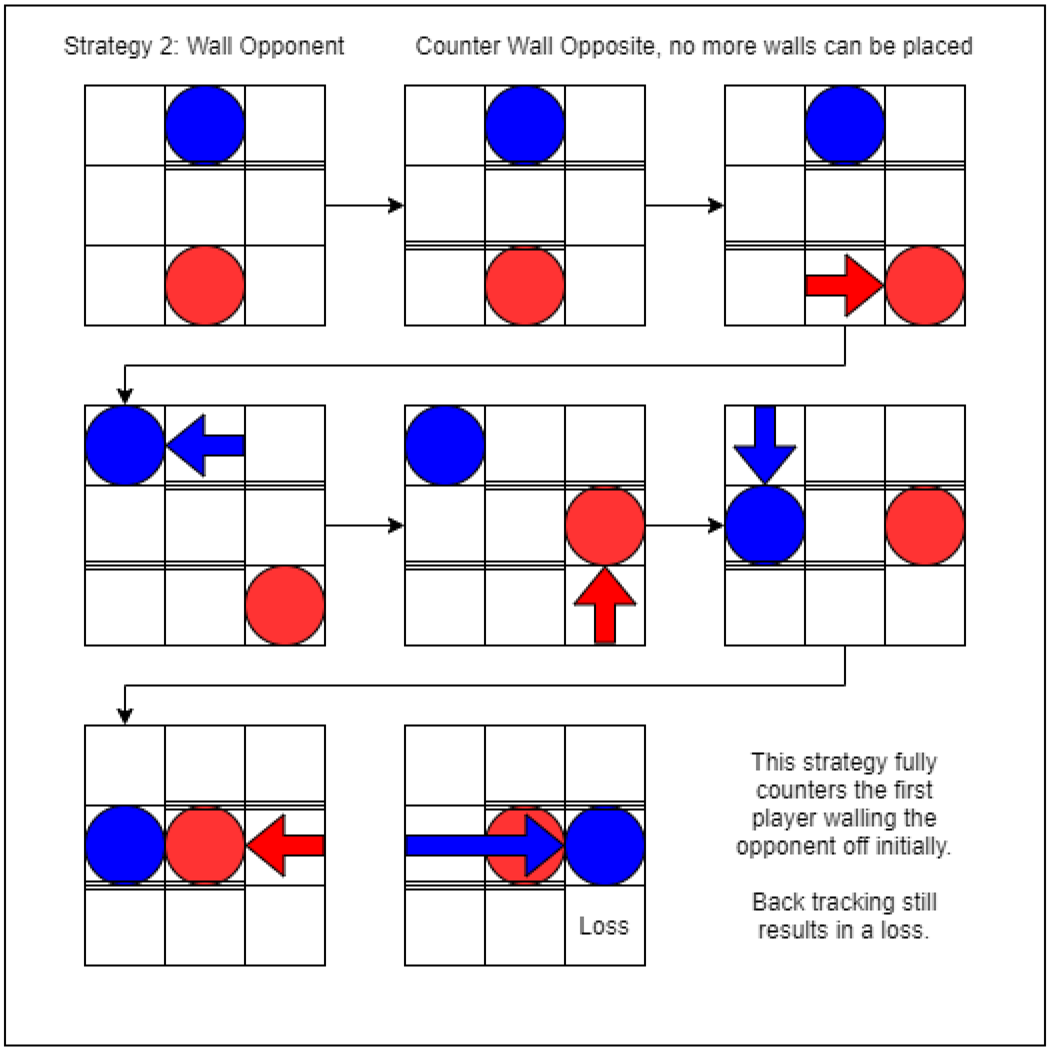
Matan Diamond
Logan Short
Mohammed Husain
Vishakha Holsambre
William Braga

Progress

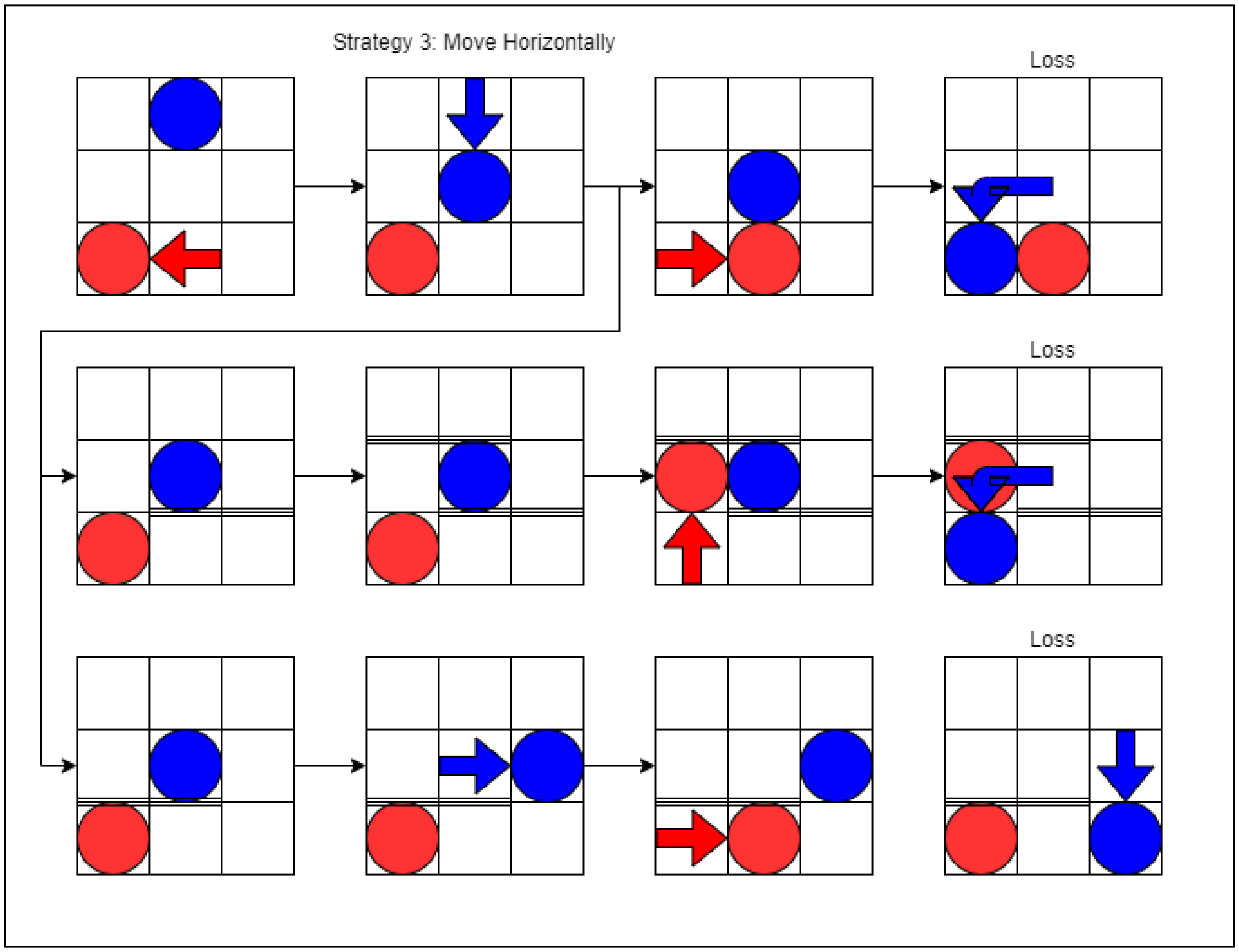
Our team sought to analyze a 3x3 board (as 2x2 and smaller are trivial) and determine if there was a clear strategy that would win the game. There are ultimately four openings, each of which we determined would lead to the opponent *winning* the game, if they played optimally. Each of these are included below.



Strategy 1, advancing the pawn initially is the most trivial as it results in the opponent immediately winning the game by jumping over your pawn.

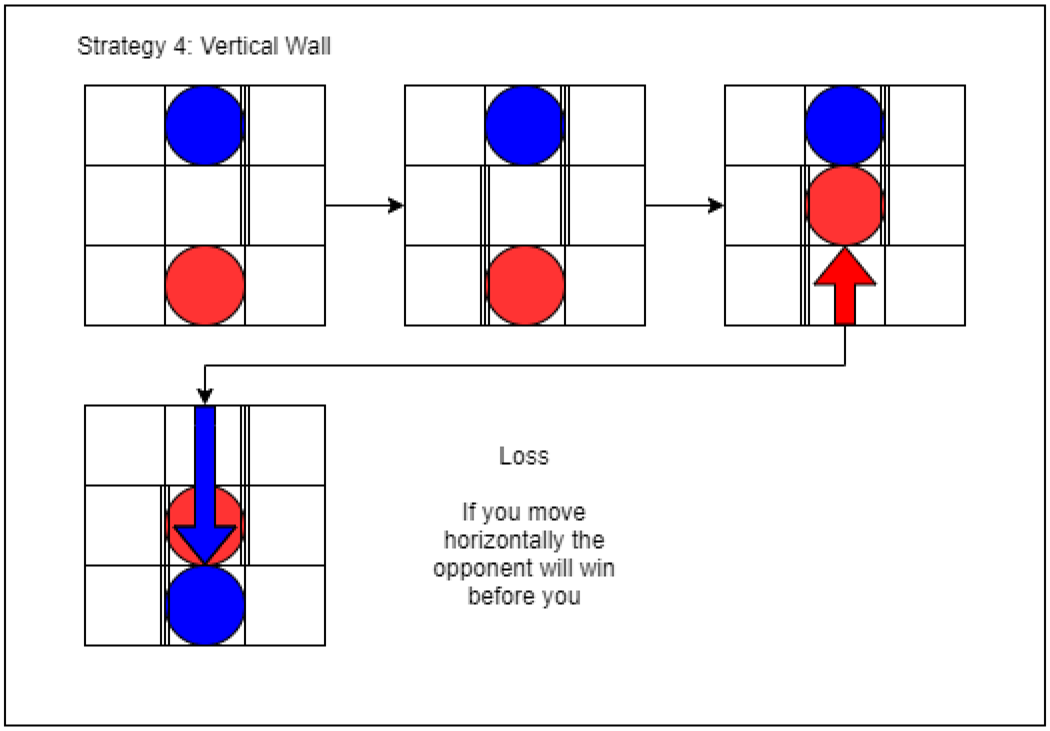

Strategy 2: Wall Opponent
Counter Wall Opposite, no more walls can be placed
Loss
This strategy fully counters the first player walling the opponent off initially.
Back tracking still results in a loss.

Strategy 2, walling off your opponent (or yourself) with a horizontal wall is countered by the opponent placing another horizontal wall on the opposite side, as depicted in the diagram. This game state prevents additional walls from being placed (except in the case where both players have crossed each other, in which case the wall would be trivial). Once in this situation, the board can almost be looked at as a long hallway if you were to take an isomorph of the board. Of course, the red player must make the first move which results in the opponent again jumping them and reaching the end first. There is no way to prevent this as the only alternative is to move backwards which is countered by the opponent simply advancing further.



Strategy 3, moving horizontally, is the most complex. It is sufficiently countered by the opponent moving their pawn forward. This state leaves three potential strategies:

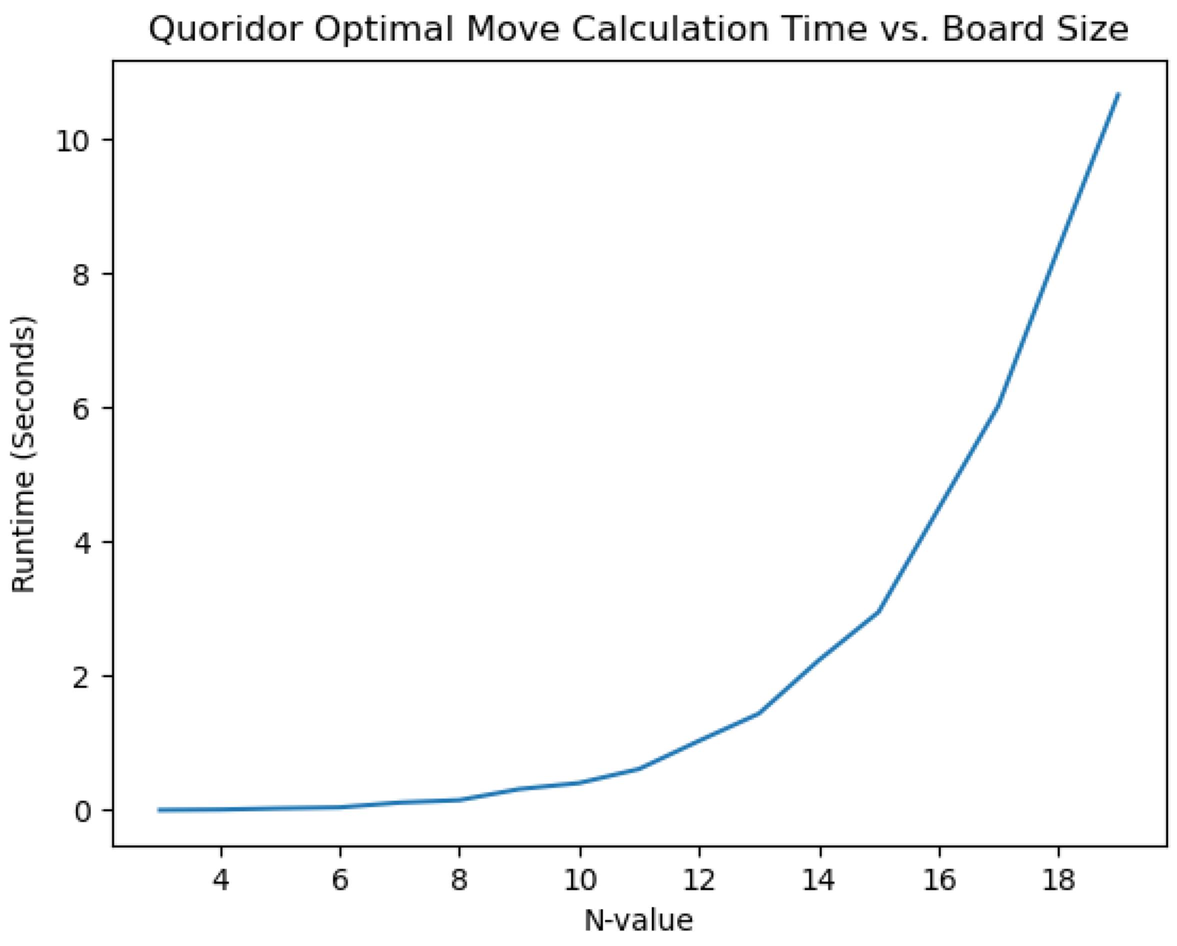
1. Moving your pawn either forward or to the side. Both of these result in an immediate loss as the opponent can reach the end next turn regardless (using the face-to-face rule if necessary).
2. Walling off the opponent, forcing them to go toward your pawn. This is countered by the opponent simply placing another wall on the opposite side of the board to again create a "hall" isomorph of the board that the red player cannot win on due to the board position.
3. Walling off the opponent, forcing them to go away from your pawn. This is countered by the opponent simply moving toward the remaining gap where they outpace you.


Strategy 4: Vertical Wall
Loss
If you move horizontally the opponent will win before you

Strategy 4, a vertical wall is countered by the opponent placing another vertical wall to again create a situation that forces the red player to move, resulting in the blue player winning the race either through a face-to-face jump or simply outpacing.

To address the computation complexity question proposed in our progress report, we built a simulator to find the best possible move. Reiterating, a move may either be a physical move of one's own piece, or the placement of a wall on the board. In order to represent this graphically, we used a dictionary in python, where each square is represented by a tuple which maps to adjacent tuples as edges. Dummy vertices were also used with edge lengths of zero from the top and bottom of the board to signify "goal" vertices for each player. When a wall is placed, corresponding edges are removed from the graph. To test the optimal move, we needed to have our program test each of the potentially four physical moves (north, south, east, west), and every potential wall placement location. A score is given for each move, calculated by subtracting one's own distance to the goal from the opponent's distance to their goal. When testing a physical move, the distance is recalculated using Dijkstra's. After placing a wall, both player's distances must be recalculated using Dijkstra's to get the new score. The maximum score achieved is the optimal move. As we found out with our program, there are often many equally optimal moves, and most often, the best possible move only achieves a score of "1", by simply moving the piece forward once.

We also plotted a runtime graph for this optimization function, found below. The trajectory of the runtime as boardsize increases is visually exponential, and this lines up with our prediction from the progress report. Since Dijkstra's is O(nlogn), and our optimization method calls Dijkstra's ~n times to account for each possible wall placement and physical move, the algorithm is O(n^2logn). While it is hard to visually distinguish between an O(n^2), O(n^2logn), and (nlogn) purely by analyzing a graph, we can reasonably assume that O(n^2logn) is the correct complexity. Code for the graph below and game optimization can be found at https://github.com/lshort82/Quoridor.



Code repositories for this project:
https://github.com/bragalotwill/Combo_Quoridor
https://github.com/lshort82/Quoridor

Future Work

Going forward there are two distinctions to be made, that being between future work that is currently feasible, and future work that is not yet / may never be possible. For the former, future work that is currently feasible, this would include further analyses of the code we have written, and anything that would simply be an extension on this project right now. For future work that is not yet possible or may never be possible, this would include technological constraints like runtime and processing power, along with physical constraints.

One possible route of future work is answering some of the questions we proposed at the beginning of the semester that we were not able to fully flesh out. One such example was how many possible game states there are given some *n* for an *nxn* board size. The main difficulty here came from the sheer amount of complexity that comes with meeting all of the rules of the game. Initially we wanted to use the product rule to decipher the number of positions possible for two players given the *n* input. From there we planned to use the inclusion-exclusion principle to specifically obtain the positions possible that do not make it impossible for one player to reach the opposite side (a necessary rule). We could easily computer the number of positions using C(n, 2) to find all combinations of both players' positions. Where this begins to become a problem too complex for us to solve in a short period of time is when considering how many walls could be placed and where they could be placed, as you cannot put walls just anywhere. Instead, they are constrained by determining if a path is now invalid as it cuts off the other player. We aimed to solve this for a reasonably-small value of n, and came up with a way to diagram and show the possible game states for a 3x3 version of Quoridor (shown above in progress). Going forward a possible approach could be to chart the increasing possible number of game states from 3x3 up to an n of 7 (7x7 board size). Doing this could help give us insight into what kind of mathematical model could best represent the scaling of possible game states given a variable board size.

Another future venture that is currently feasible, albeit very slow at a sufficiently high n value, is to continue our data collection for our optimal move calculation time versus board size for increasingly higher values of n. We have the programming capabilities to do this, as it requires repeated calls of Dijkstra's algorithm, but as shown above in the graph and in the Progress section, this implementation has a $O(n^2 \log n)$ runtime, which will very quickly become complex beyond our capabilities (as students).

A currently impossible future work could be to 'solve' the game of Quoridor, meaning the outcome (win, lose or draw) can be correctly predicted from any position, assuming that both players play perfectly. As of this moment, Quoridor is still too complex to "solve" in that there is no way to guarantee either a win or loss from a given starting position and optimal play. In that regard it is very similar to chess, and perhaps in some-odd decades or centuries it will be computable, but as of now it falls into the same category as chess and go, unsolvable. For example, the decision to move your player one space to the left or the right could mean the difference between winning or losing the game – but it's 20 moves from now, and at this point for a sufficiently high game board size (including the original Quoridor board size, 9x9) these calculations are simply not yet possible.