Introduction

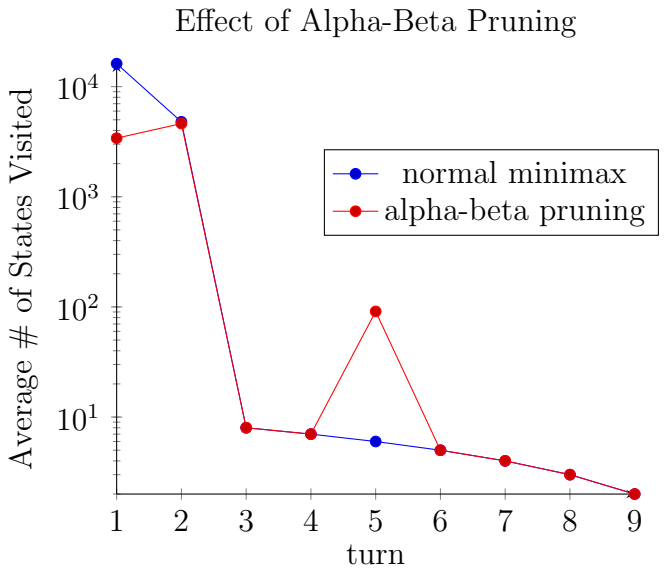
For the third week, I worked on alpha-beta pruning for the minimax algorithm. Once this was completed, I updated the board hash to store a `hash` variable, which updates with each move, rather than rehashing completely each time.

I also added an evaluation function, by which moves would be ordered. This works separately from the result function (leaf nodes), but does begin by checking if the game is in a terminal state and returning $-\infty$, 0, or ∞ when appropriate. Otherwise, all game lines are considered, and if the opponent has the opportunity to win in the next move, $-\infty$ is returned. If the current player has a fork (two opportunities to win, so the opponent has no chance), ∞ is returned. Otherwise, for each line that the player or opponent has control over, the number of squares taken in that line is added/subtracted (respectively) from the score. This score is then returned as the evaluation.

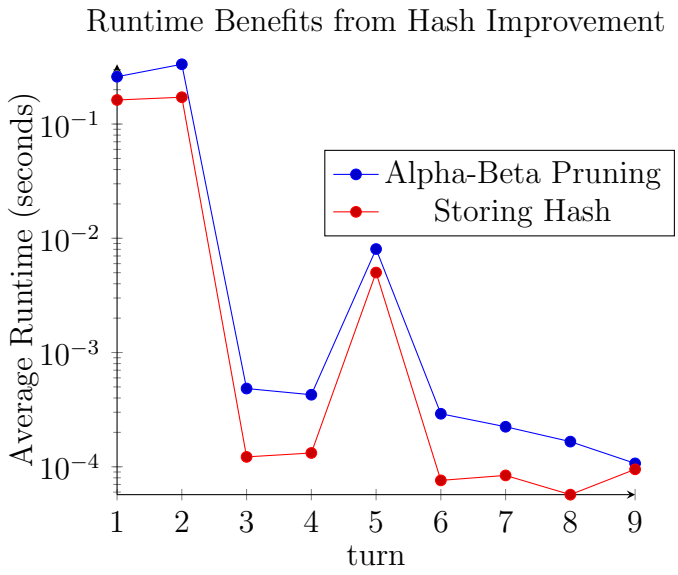
As shown by the following results, more improvements may need to be made to the evaluation function. This can be achieved through further testing and experimentation.

Results

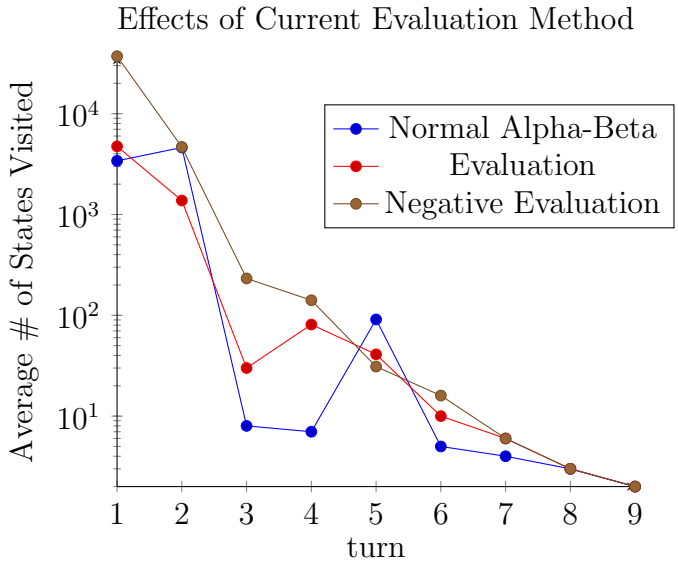
The following graph shows the result of applying pruning. This is most effective on the first move, where the pruned method visits only about 20% as many states as the original. There happens to be more states visited during the 5th move, as there aren't as many cached, but this is overexaggerated by the graph scale (compared to the larger difference on the first move).



The next graph highlights that the new hashing method (storing the hash rather than rehashing) has a consistent runtime improvment.



The following graph shows a comparision of states visited from a normal alpha-beta search, and two searches where moves are ordered by the current evaluation, in both a normal/positive and a reverse/negative fashion. Despite some inconsistencies, the normal move ordering seems to improve on random ordering overall (particularly on the second move), and is generally better than its reverse.



The first couple moves clearly take up a majoirity of the runtime (by a drastic amount), so this will be the main priority when implementing further tests. Further tests on larger boards will be useful in understanding the effectiveness of move-ordering methods.