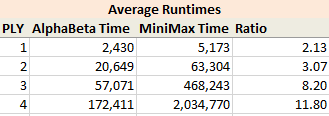
**Cambridge Coding Academy: Coding ++ Assessment**

**By Ishita Swaroop**

**Alpha-Beta Pruning AI for Connect Four**

**Run Time Comparison Of AlphaBetaPruning vs MinimMax Time For Different PLYs**

The time each AI spent choosing a move was computed using pygame.time.get\_ticks() and measured in milliseconds. The data shows that AlphaBeta Pruning cuts down the search times substantially.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

The plot of average runtime for the two AI shows that time taken grows exponentially as PLY, i.e. depth of search is increased. I plotted the graph again with logarithmic vertical axis (base-9, as we had 9 possible moves). This graph reveals that AlphaBeta is substantially faster than MiniMax.

**Raw Data For Different PLYs, Multiple Runs, winners, time spent by AI**

I set the first mover (red) to be AlphaBetaAI and second mover (blue) to be MiniMaxAI. The initial implementation had the search of moves from the first column onwards in increasing order (from left to right). This always resulted in the second player winning, which was a bit surprising. I changed the column search order to be random; and the data for different PLYs is shown below.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Runtimes and Winners for PLY=1** | | |  | **Runtimes and Winners for PLY=2** | | |  | **Runtimes and Winners for PLY=3** | | |
| **WINNER** | **AlphaBeta Time** | **MiniMax Time** |  | **WINNER** | **AlphaBeta Time** | **MiniMax Time** |  | **WINNER** | **AlphaBeta Time** | **MiniMax Time** |
| AlphaBeta | 2450 | 5190 |  | MiniMax | 16082 | 57480 |  | AlphaBeta | 54615 | 451637 |
| AlphaBeta | 2401 | 5015 |  | AlphaBeta | 32187 | 93610 |  | AlphaBeta | 37815 | 313436 |
| AlphaBeta | 2317 | 4742 |  | MiniMax | 21353 | 69898 |  | AlphaBeta | 74358 | 599573 |
| AlphaBeta | 2591 | 5002 |  | AlphaBeta | 26890 | 82983 |  | AlphaBeta | 61496 | 508325 |
| AlphaBeta | 2193 | 4727 |  | MiniMax | 25096 | 72713 |  | **Average** | **57071.00** | **468242.75** |
| AlphaBeta | 2366 | 4973 |  | MiniMax | 15632 | 49940 |  |  |  |  |
| AlphaBeta | 2359 | 5006 |  | AlphaBeta | 15377 | 48340 |  |  |  |  |
| AlphaBeta | 2217 | 4460 |  | MiniMax | 20822 | 62789 |  |  |  |  |
| AlphaBeta | 2414 | 5472 |  | AlphaBeta | 13268 | 40191 |  |  |  |  |
| AlphaBeta | 2536 | 5600 |  | MiniMax | 19779 | 55095 |  |  |  |  |
| AlphaBeta | 2860 | 6498 |  | **Average** | **20648.60** | **63303.90** |  |  |  |  |
| AlphaBeta | 2108 | 4732 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2437 | 4815 |  |  |  |  |  |  |  |  |
| AlphaBeta | 1911 | 4697 |  |  |  |  |  |  | When PLY=1 for AlphaBeta and MiniMax, AlphaBeta had a 100% win rate. This is most likely due to the first player advantage in the setup of the game, rather than AlphaBeta being better than MiniMax. In first play advantage, the 2nd player is only preventing the 1st from scoring rather than playing so that It maximises its own points – defending rather than attacking. Furthermore, the 1st mover will always reach the target first when the AIs are evenly matched. When ply = 1, the AIs only maximise their win chances and do not look at opponent’s possible moves.  For PLY=2, with win-ratio of the two AI is almost a 50-50% split, so AlphaBetaAI is weaker at this depth. This is because both player AIs are looking two steps ahead, where they are not only trying to maximise their own score but also trying to minimise their opponent’s score. This negates the first player advantage.  The first player advantage is restored for PLY=3, wherein the player AI gets to look two moves ahead. While PLY=2 is obviously weaker because of the smaller depth, allowing the opponent to win more often, PLY=3 has a greater depth. This, coupled with first player advantage, may be why AlphaBeta had a 100% win rate.  I was very surprised at how the 2nd player, MiniMax, always won when PLY=4, but could not figure out the reason. ss  **I would certainly choose the AlphaBetaAI in a tournament as it has better run time performance, I would use it with a PLY of 2 as it seems to negate the first mover advantage. If my AI is the first mover, I will choose a PLY of 3.** |  |
| AlphaBeta | 1848 | 3929 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2340 | 4754 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2522 | 5115 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2244 | 5099 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2202 | 4600 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2887 | 6045 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2452 | 5044 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2775 | 6020 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2369 | 4604 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2198 | 5032 |  |  |  |  |  |  |  |  |
| AlphaBeta | 3337 | 7220 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2521 | 5258 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2522 | 5378 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2087 | 4628 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2483 | 4923 |  |  |  |  |  |  |  |  |
| AlphaBeta | 2945 | 6616 |  |  |  |  |  |  |  |  |
| **Average** | **2429.73** | **5173.13** |  |  |  |  |  |  |  |  |