2048: Minimax vs Minimax + alpha-beta

By Prajnyique, Jackson, Angel

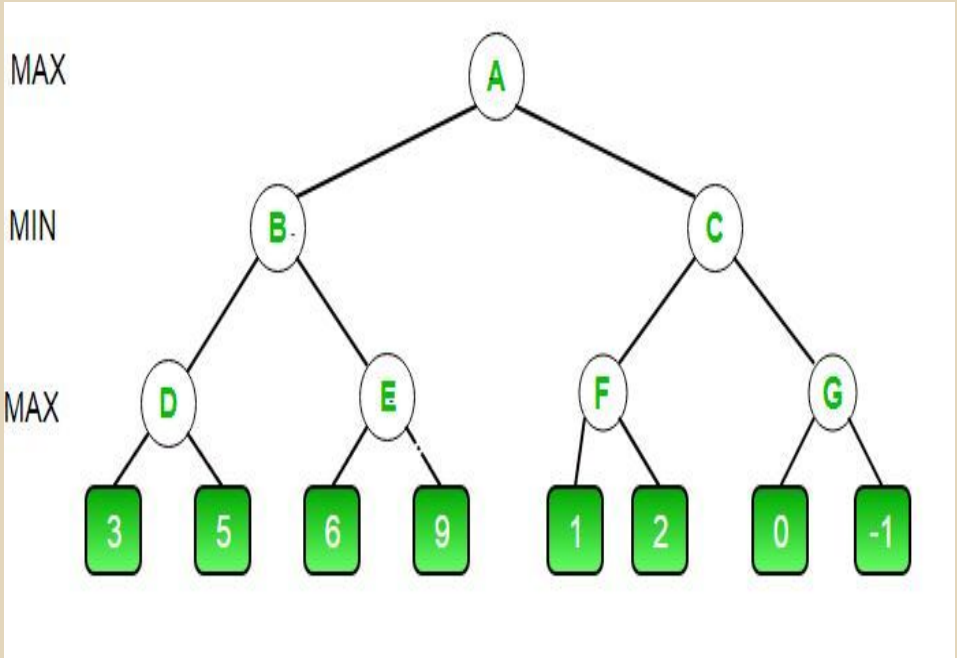
2048

Problem

The goal of this project is to build an AI agent that can play the game 2048 by selecting optimal moves. We compare two adversarial search algorithms:

1. Minimax
2. Minimax with Alpha-Beta Pruning

Our objective is to evaluate how effectively each algorithm makes decisions in 2048, and compare their efficiency and speed.



Background

2048 is a single-player sliding-tile puzzle where tiles merge into higher values when moved in four directions. Normally, the game includes randomness because a new tile (2 or 4) appears after each move.

For experimental consistency, we model the environment as adversarial (vs. computer), where the “opponent” places the worst possible tile. This makes the game compatible with Minimax logic.

- Minimax evaluates future board states by exploring all possible moves and adversarial tile placements.

- Alpha-Beta Pruning is an optimization of Minimax that eliminates branches that cannot possibly influence the result, enabling faster search while producing identical decisions.

Hypothesis

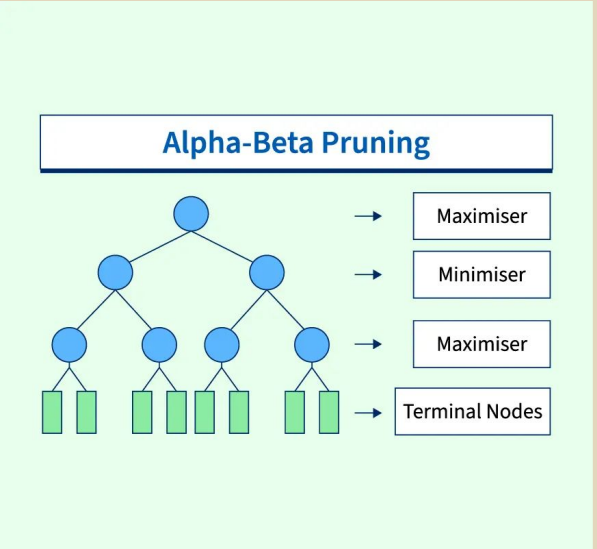
We expect that:

- Minimax with Alpha-Beta Pruning will explore far fewer game states, enabling deeper search levels within the same time constraints.
- Decision quality (the chosen move) will be similar between the two algorithms, but the Alpha-Beta version will be significantly more efficient.
- As search depth increases, Alpha-Beta Pruning should provide better overall gameplay performance because it can search deeper and quicker while Minimax becomes too slow.

Algorithm

- Minimax: Minimax is a search algorithm used in adversarial games

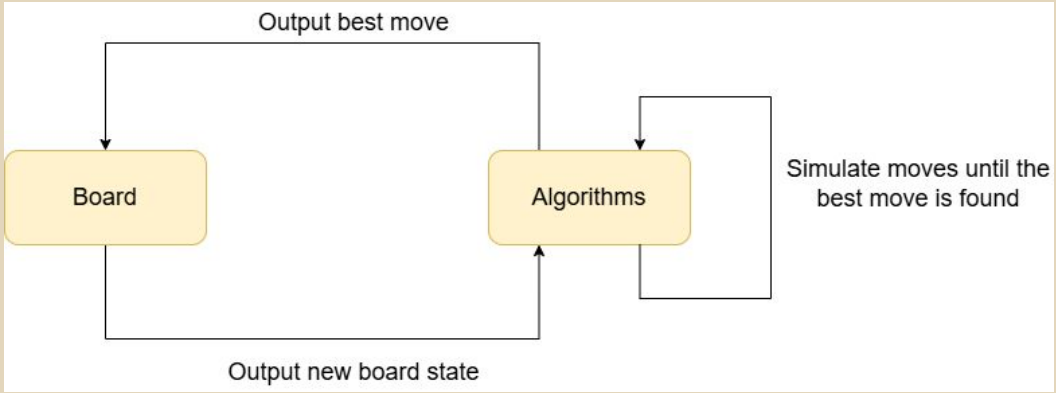
- Minimax + Alpha-Beta pruning: an optimization of Minimax that cuts off branches of the tree that cannot possibly affect the final decision.



Architecture

Board:
- Board
- Moves
- Random Spawning

Algorithms:
- Minimax w/ Alpha-Beta pruning
- Minimax w/o Alpha-Beta pruning
- Evaluation functions
- Depth Limit (7)



Evaluation Method

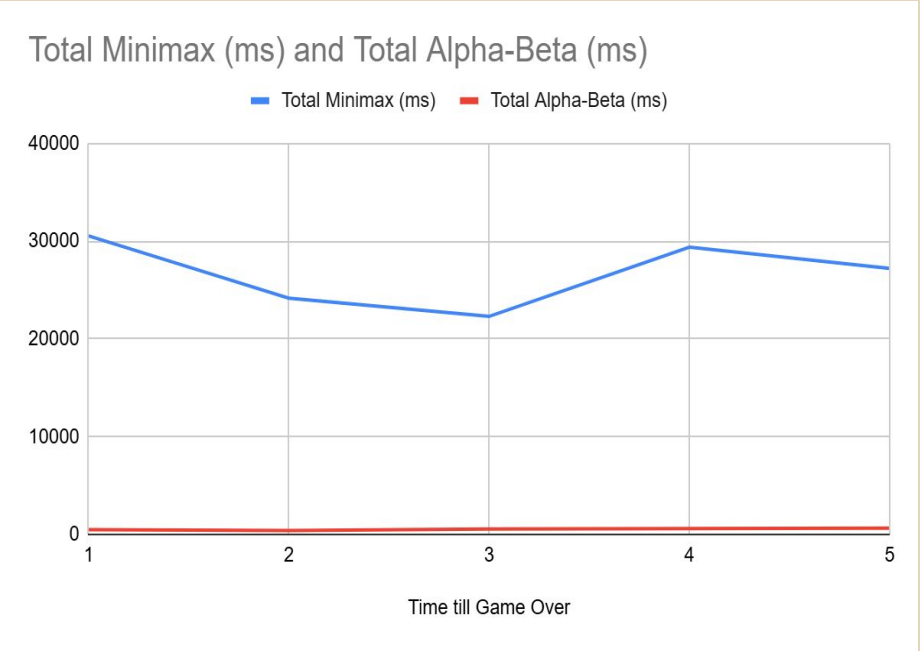
We will be evaluating time in order to compare how long minimax and minimax + alpha-beta takes to complete 10 moves. Then take the average and find out which algorithm was the most efficient.

We also decided to record the scores that the algorithms achieve in the game. We believe that they should achieve extremely similar results, as they should have the exact same performance aside from how fast they are.

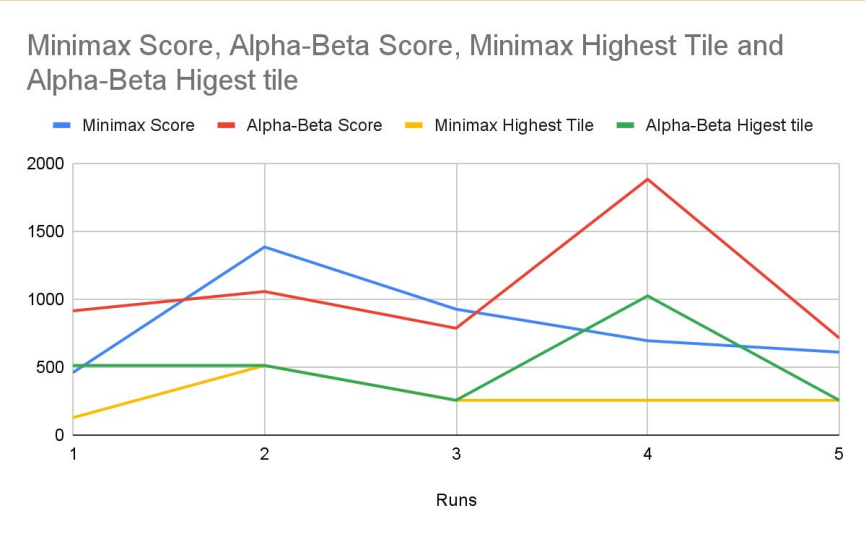
Results: Initial 10 moves

Key Findings:

- Minimax is **significantly slower** during early gameplay.
- Alpha-Beta pruning drastically reduces the computation time per move.
- Example from your data:
    - **Minimax average (first 10 moves): ~11,338 ms per move**
    - **Alpha-Beta average (first 10 moves): ~118 ms per move**
- Alpha-Beta is roughly **100× faster** on average.



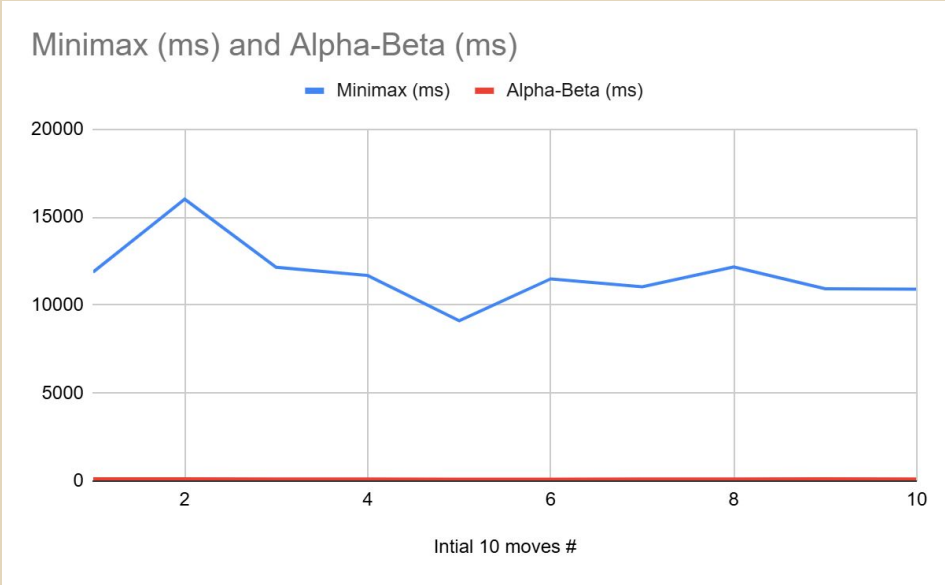
Results: Total Time & Game Outcomes



Total computation until game over:

- **Minimax average:** ~26,900 ms
- **Alpha-Beta average:** ~536 ms

Alpha-Beta consistently ran the game **60–75× faster**.



Game performance (5 runs):

- **Minimax avg score:** 814
- **Alpha-Beta avg score:** 1070
- Alpha-Beta outperformed Minimax in **3 out of 5 runs**.

Highest tile reached:

- Minimax typical: **256–512**
- Alpha-Beta typical: **256–1024**

Conclusions

Success in the Game:

- Neither algorithm reliably solves the 2048 game; reaching the 2048 tile is rare.
- Limited search depth and a simple evaluation function prevent the AI from planning long-term strategies.
- The AI tends to prioritize immediate merges over strategic positioning, leading to board lock situations.

Insights from Timing and Gameplay:

- Timing analysis shows Alpha-Beta pruning is consistently faster than Minimax for the same search depth.
- Performance scales poorly for deeper searches; even with pruning, very deep searches are computationally expensive.

Takeaways:

- Both algorithms demonstrate the importance of search efficiency (Alpha-Beta pruning).
- Game-solving AI requires advanced heuristics beyond basic Minimax evaluation for long-term success.

Limitations

Search depth restriction: Both Minimax and Alpha-Beta are limited by fixed search depth, preventing long-term planning.

Simple evaluation function: The AI uses the sum of tiles, which does not account for board strategy (e.g., corner placement, monotonicity).

Inability to consistently solve 2048: The AI often fails to reach 2048 because it prioritizes immediate merges over strategic positioning.

Computational cost: Even with Alpha-Beta pruning, deeper searches are slow and resource-intensive.

Expectimax: Minimax is inefficient for this kind of game. For future projects, it would be much more effective to use an algorithm like expectimax, which takes randomness into account.

THANK YOU!

Or else.