Coursework 2: Adversarial Search

To evaluate how Minimax scales with increasing game size, we measured the execution time for computing the first move from the initial game state for several (m, n, k) configurations. We then compare these results between setups with and without **α-β** pruning.

We recorded:

- **Execution time** for computing the optimal move
- **Number of states (nodes) visited** during search
- **Effectiveness of α-β pruning** in reducing computation

Results Overview

The following experiments were conducted:

Table 1. Execution times for action selection and number of visited states (nodes) with and without pruning for increasing values of m, n, and k.

m × n × k	**Without Pruning**		**With Pruning**	
	Time Taken (s)	Nodes Visited	Time Taken (s)	Nodes Visited
3×3×3	3.257	549,946	0.1266	21,731
3×4×3	4751.2218	276,911,233	0.5891	84,467
4×3×3	2,688.37	276,911,233	2.0565	284,487
4×4×3	-	-	49.632	5,958,619

As the board grows, the branching factor and depth of the game tree both increase. This makes Minimax expensive. α-β pruning reduces the number of evaluated states significantly, especially for larger boards, at the same time substantially reducing execution times (Table 1).

1. **Observation 3×3×3**
    - Minimax without pruning is fast because the search tree is small.
    - Pruning makes only a small improvement (tree is already small).

2. **3×4×3 and 4×3×3**
    - The tree grows substantially.
    - α-β pruning cuts off branches early, reducing time & nodes by a big factor.

3. **4×4×3**
    - Minimax **without pruning does not finish in reasonable time** because of exponential explosion.
    - α-β pruning is able to compute a result, but still takes noticeably longer than smaller board.

4. **4x4x4** and higher values of m, n, and k
    - Minimax **both with and without pruning does not finish in reasonable time** because of the exponential explosion.
    - α-β pruning is not able to compute a result in **reasonable time**, execution takes significantly longer than for other configurations.

Key Findings

- Minimax execution time grows **exponentially** with board size.
- α-β pruning dramatically reduces:
    - number of states visited
    - execution time
- For large boards (e.g., 4×4×3), Minimax without pruning becomes **computationally infeasible**, illustrating its poor scaling.

Discussion

1) Effect on Time as m, n, k Increase

The results clearly show that the time required for Minimax action selection **grows significantly** as the board size increases (Table 1), as expected from the theoretical complexity of Minimax.

- On the 3×3×3 board, the non-pruned Minimax completes in **~3.26 seconds**, while the pruned version is almost instantaneous at **~0.13 seconds**.

- For 3×4×3, the non-pruned Minimax time explodes to **~4751 seconds (~1.3 hours)** due to the exponential growth of the game tree. The pruned version completes in **~0.59 seconds**, demonstrating the huge efficiency gain.
- The non-pruned Minimax for 4×4×3 is infeasible because of the combinatorial explosion as both m and n increase.

2) Effect on Number of States Visited as m, n, k Increase

The number of states visited follows the same trend as execution time. We notice that the number of nodes visited grows exponentially with board size and k, but pruning effectively reduces redundant exploration (Table 1):

- 3×3×3 without pruning visits **~550,000 nodes**, whereas 3×4×3 without pruning visits **~277 million nodes**, illustrating how small increases in board size cause **explosive growth** in the search tree.
- Pruning drastically reduces the number of nodes visited: 3×3×3 with pruning visits only **~22,000 nodes**, and 3×4×3 with pruning visits **~84,500 nodes**, a reduction of several orders of magnitude.
- 4×3×3 with pruning visits **~284,000 nodes**, showing that geometry affects node counts even with the same total number of cells.

3) Effect of α–β Pruning

α-β pruning has a dramatic effect on both execution time and number of states visited. As the board grows, the branching factor and depth of the game tree both increase. This makes Minimax expensive. α-β pruning reduces the number of evaluated states significantly, especially for larger boards.

- Across all tested boards, pruning reduces **execution time** by **one to four orders of magnitude**.
- It also reduces the **number of nodes visited** by a similarly large factor.
- The effect becomes more pronounced as the board size increases: for 3×4×3, pruning reduces runtime from **~4751 s to 0.59 s** and nodes visited from **~277 million to ~84,000**.
- Without pruning, computing Minimax on a 4×4×3 board is **practically infeasible**, as the number of states grows exponentially and the runtime takes hours or more. However, α–β pruning allows the computation to finish in a **reasonable time (~50 s).**