**Discussion Report**

**Rohith Jathan**

1.

There are several factors that contribute to the AI exceeding the time limit in Tic-Tac-Toe. Despite its simplicity and limited search space, the game can present computational challenges, particularly when algorithms like iterative deepening with Alpha-Beta pruning are employed. As the game progresses and the search depth increases, the number of nodes evaluated grows exponentially, potentially surpassing the allocated time limit. Moreover, the method used for time measurement (time.perf\_counter() in Python) may introduce small inaccuracies that accumulate over multiple iterations, causing the AI to exceed the intended time limit. Furthermore, depending on the game state and the effectiveness of pruning strategies, the AI may initiate deeper searches than anticipated in order to identify optimal moves. This inclination can lead to the AI surpassing the time limit, especially if pruning conditions are not strictly enforced or if there are inefficiencies in move ordering.

2.

To address these challenges, we can implement more precise time management techniques within the AI algorithms. Instead of relying solely on time.perf\_counter(), using time.time() for time measurement can provide wall-clock time that aligns better with user expectations. Introducing a heuristic-based threshold for iterative deepening offers another solution. Rather than allowing iterative deepening to continue indefinitely until the time limit is reached, setting a maximum depth or heuristic cutoff based on board complexity or heuristic evaluation can prevent unnecessary deepening. Enhancing move ordering strategies within Alpha-Beta pruning by prioritizing moves likely to result in early pruning—such as capturing immediate threats or securing advantageous positions—significantly reduces the search space, thereby improving adherence to the time limit. Adopting a fixed time budget per move instead of a cumulative time limit for the entire move computation ensures that each move computation remains within a predictable time frame, thereby enhancing responsiveness during gameplay. Lastly, conducting thorough profiling of AI algorithms using Python's profiling tools (such as cProfile, timeit, etc.) allows for identification and optimization of computational hotspots.