My graph matches my runtime analysis expectations. In this assignment, structuring and implementing the different prefix scan algorithms went smoothly. The explanation of time complexity for each method provided clear insights into the trade-offs between single-threaded and parallel approaches. Leveraging CUDA for parallel computing was effective in demonstrating speedup with naive and recursive doubling techniques. Integrating CUDA events for timing helped capture accurate performance metrics, and adding loops to handle various input sizes made the comparisons more comprehensive.

The main challenge was managing shared memory and thread synchronization in the recursive doubling implementation. Ensuring correctness while avoiding race conditions required careful use of synchronization primitives like `__syncthreads()`. Additionally, optimizing the thread and block configurations to handle different array sizes without exceeding hardware limitations was a bit tricky.

If approached again, I would incorporate error checking for CUDA operations to ensure robust code execution and use libraries like Thrust to compare custom implementations against highly optimized GPU routines. Finally, profiling tools like NVIDIA Nsight could be used to gain deeper insights into performance bottlenecks and optimize memory usage further.