Lab 2 Report

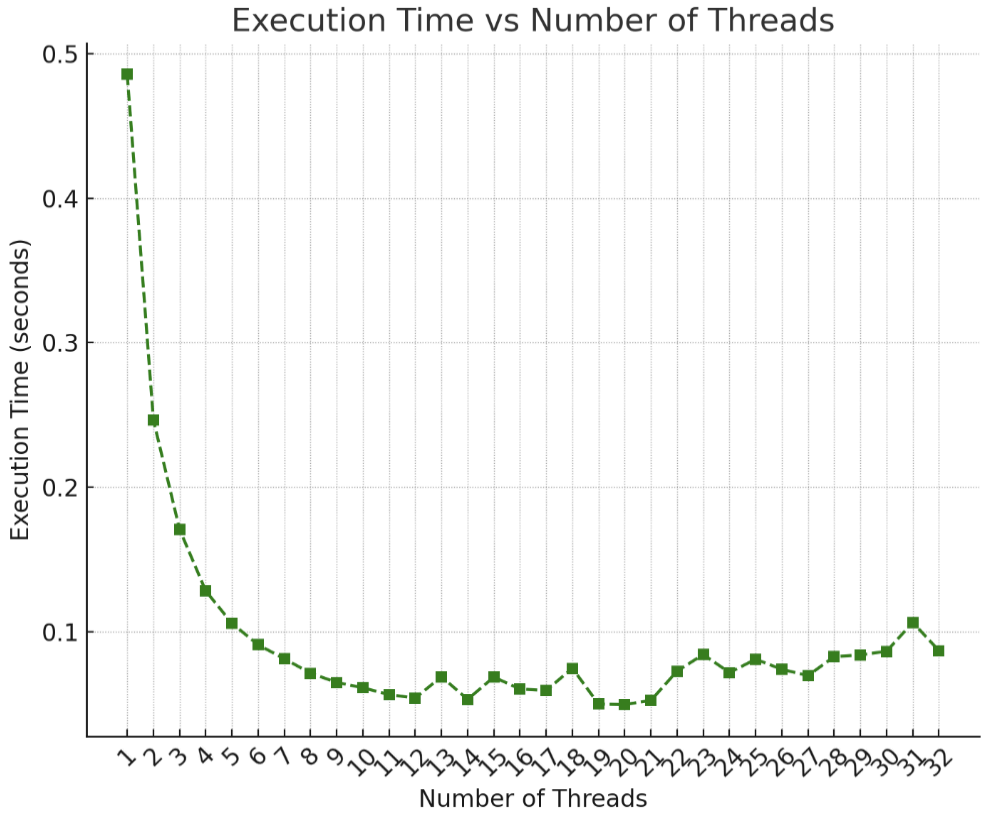
1. Does the code support up to 32 threads?
    a. Yes, the code supports up to 32 threads.

2. Did you include speedup data (timings)?
    a. Here is my data:
        i. https://docs.google.com/spreadsheets/d/1oMBrsmOyL_A33XLkFTRCbEVrHHRqDJmaJRYAdme-gj4/edit?usp=sharing
    b. Yes, I included the speedup data. The speedup data was calculated using the formula Speedup = $\frac{T_1}{T_n}$, where T1 is the best time taken with 1 thread, and Tn is the time taken with n threads. This is shown in the purple plot of the speedup graph.

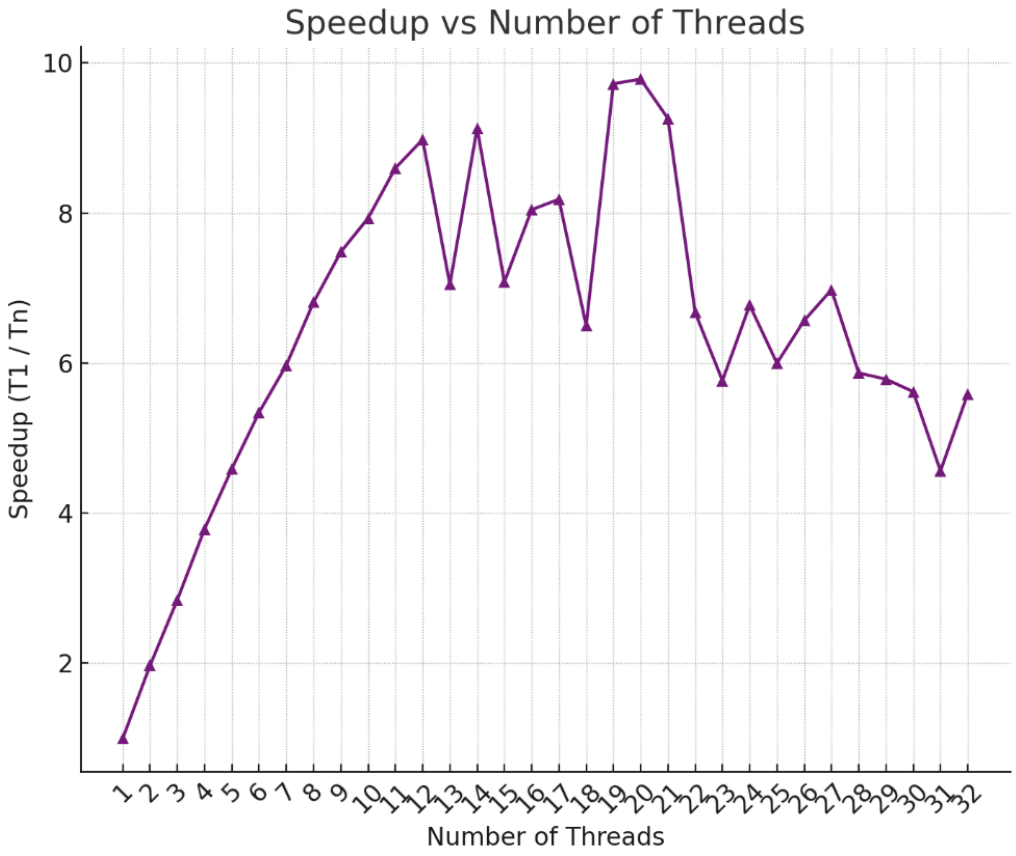
3. Did you include a graph?

Yes, I included an execution time graph and a speedup graph.

Execution Time Graph:



Speedup Graph:



4. Did you identify where you hit the Amdahl limit? And defend why you think it is there?
    a. The Amdahl limit is around **12 threads** in the Speedup graph, where speedup reaches its highest point and then starts to fluctuate or decline. Adding more threads beyond this point doesn't help much and even slows things down because of extra synchronization work, like pthread_barrier_wait calls.

5. How well does the partition scheme take memory into account?
    a. This partition scheme is memory-efficient because each thread works on its own row block, reducing memory access conflicts. However, as we go beyond the Amdahl limit, performance drops due to frequent data swaps and accessing boundary rows, which can cause cache misses and extra synchronization. Overall, this approach works well with a moderate number of threads but could be improved for higher counts to reduce these issues.