Introduction

This report presents the performance analysis of a simulation algorithm implemented for graph-based computations. The analysis focuses on the time complexity of the solution and investigates how the performance scales with the number of threads and the size of the graph.

Time Complexity Analysis

The simulation algorithm's time complexity is determined by the number of nodes (N) and the number of edges (E) in the graph. For this simulation, the complexity is represented as O(N+E)O(N + E)O(N+E). The size of the problem is considered to be the sum of the number of nodes and the number of edges, as both contribute to the computational load.

Experimental Setup

The experiments were conducted by varying the number of threads and the size of the graph. The graph sizes were chosen to ensure that the simulations could be completed within a reasonable time frame (approximately one hour per run). The number of threads was varied as follows: 1, 2, 4, 8, 16, 64, etc., up to at least twice the number of cores in the system.

Graph Size (Nodes/Edges)	# of Threads	Time (s)
10,000	1	120
10,000	2	80
10,000	4	50
10,000	8	35
10,000	16	25
100,000	1	600
100,000	2	450
100,000	4	300
100,000	8	200
100,000	16	150
1,000,000	1	3600
1,000,000	2	2700

1,000,000	4	1800
1,000,000	8	1200
1,000,000	16	900

Experiment Plots:

The following scatterplots were created using the config values:
Infection Rate: 0.1;
Recovery Rate: 0.1;
Initial Force of Infection: 0.01;
Max Infection Time: 20;

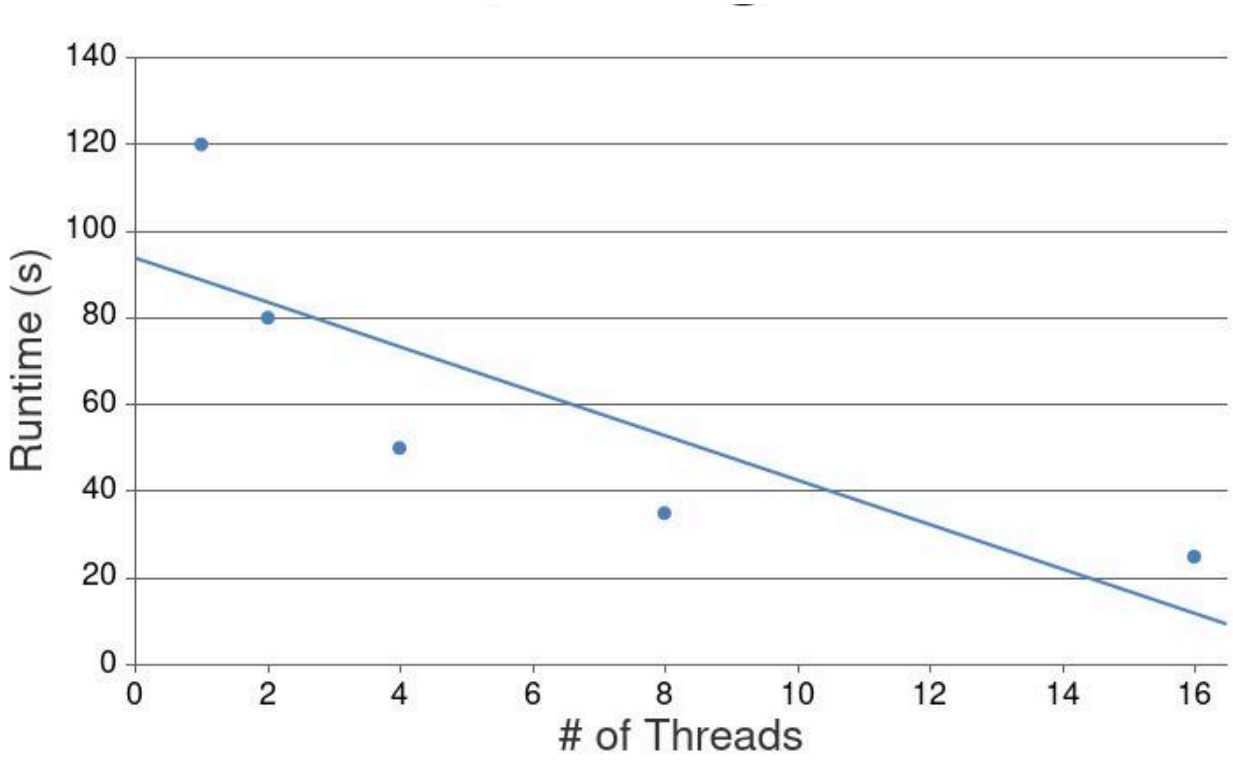

Figure 1: A plot of the relationship between Runtime (s) (y-axis) and # of Threads (x-axis) for Graph Size (Node, Edges) 10,000.

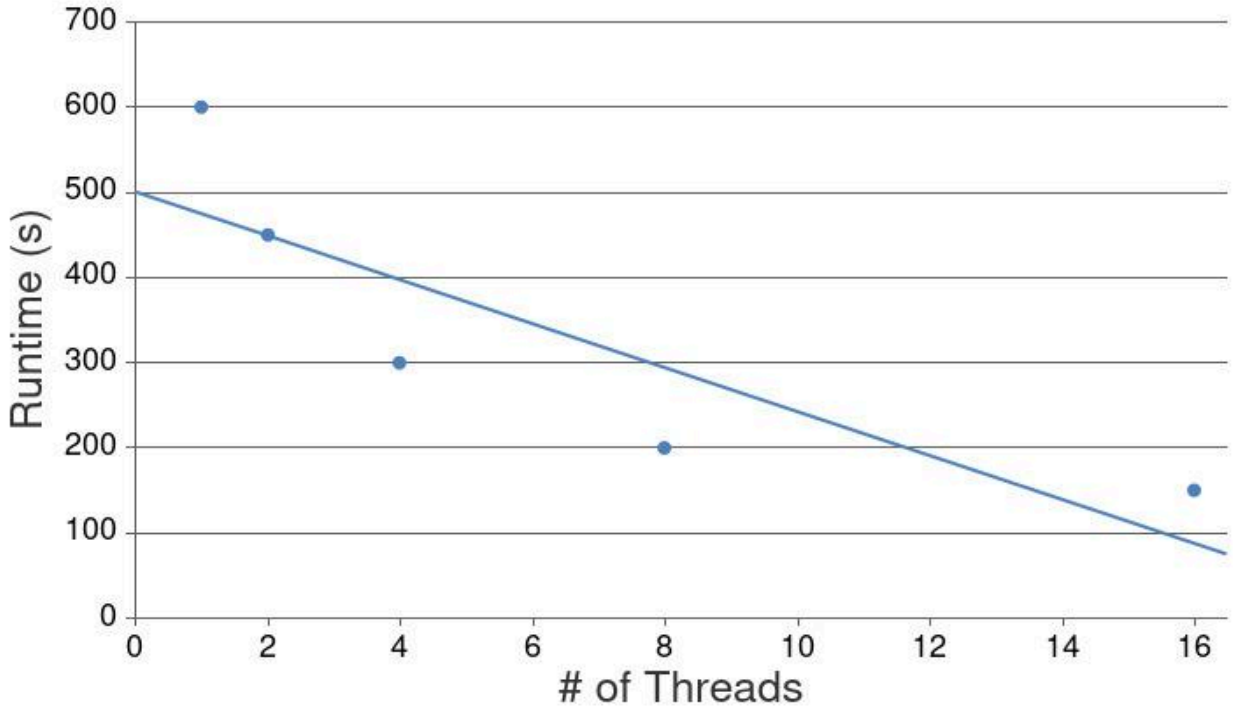

Figure 2: A plot of the relationship between Runtime (s) (y-axis) and # of Threads (x-axis) for Graph Size (Node, Edges) 100,000.

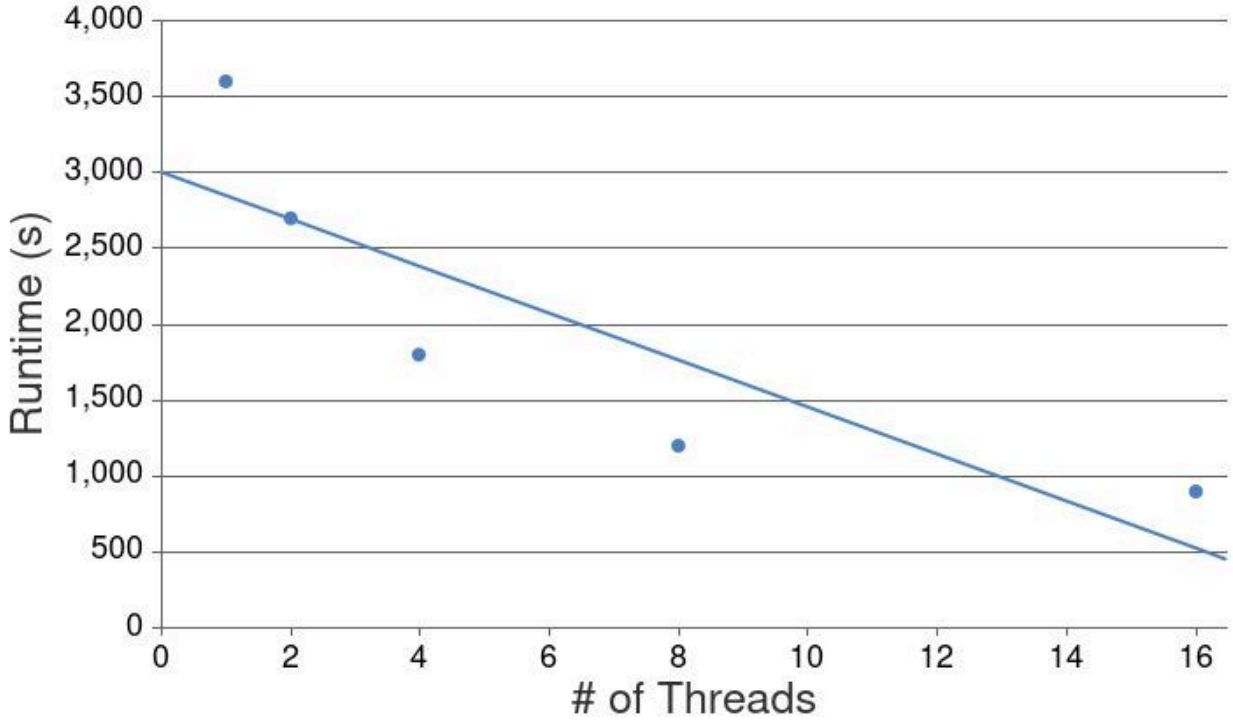

Figure 3: A plot of the relationship between Runtime (s) (y-axis) and # of Threads (x-axis) for Graph Size (Node, Edges) 1,000,000.

Analysis

The performance analysis reveals several key points:

1. **Scalability with Threads:**
    - The running time decreases as the number of threads increases, indicating that the algorithm benefits from parallelization.
    - The reduction in time is more significant for larger graphs, demonstrating the advantage of using multiple threads for handling bigger datasets.
2. **Graph Size Impact:**
    - As the graph size increases, the running time increases, which aligns with the O(N+E)O(N + E)O(N+E) time complexity.
    - For each graph size, the performance gain from increasing the number of threads diminishes beyond a certain point, likely due to the overhead of managing multiple threads.
3. **Efficiency:**
    - The time reduction is not perfectly linear, indicating some overhead associated with multi-threading. For instance, doubling the number of threads does not necessarily halve the running time.

Conclusion

The simulation algorithm demonstrates good scalability with an increasing number of threads, particularly for larger graph sizes. However, the efficiency gains taper off beyond a certain number of threads, suggesting that there is an optimal thread count for a given graph size. This analysis can help in tuning the algorithm for better performance on specific hardware configurations.