Week-4

1. Implement Strassen's Matrix Multiplication to optimize matrix operations in machine learning, where training large neural networks requires frequent multiplication of weight matrices with input data; compare the performance of Strassen's algorithm with naive multiplication by testing on square matrices of different sizes (64×64, 128×128, 256×256, 512×512), record execution times, and plot a graph of matrix size versus execution time to illustrate how Strassen's O(n^2.81) algorithm scales better than the naive O(n^3) approach for computationally heavy ML workloads.

[5m]

2. An e-commerce company needs to sort millions of daily transactions quickly for reporting. Quick Sort is fast on average but degrades to O(n^2) in worst cases, while Merge Sort guarantees O(nlogn) but with extra memory overhead. A Hybrid Sorting algorithm applies Quick Sort normally but switches to Merge Sort when recursion depth exceeds log^2n, achieving both speed and reliability.

Tasks: [5m]

1. Implement Quick Sort, Merge Sort, and Hybrid Sort.
2. Run experiments on arrays of sizes 1000, 5000, 10000, 50000.
3. Compare execution times of all three algorithms.
4. Plot input size vs execution time using gnuplot.

Expected Result:

- Quick Sort: fastest on random data, slow on nearly sorted data.
- Merge Sort: consistent but slower due to overhead.
- Hybrid Sort: combines Quick Sort's speed with Merge Sort's stability, performing best overall.