Programming Assignment 1
THREE GENIUSES

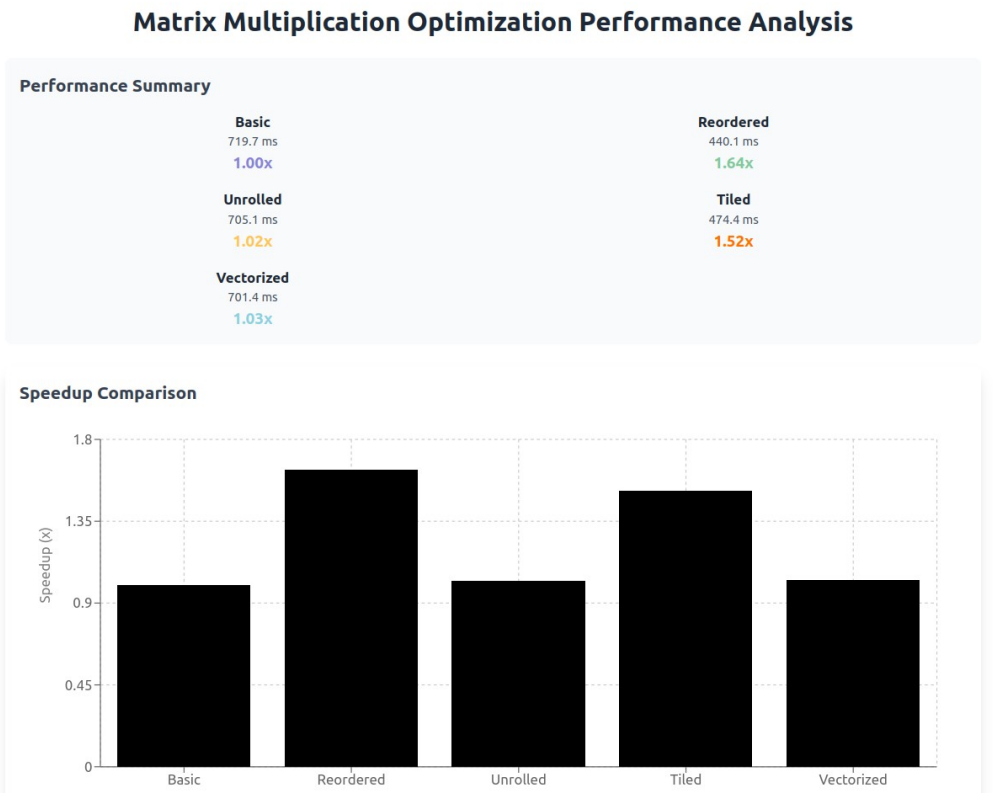
Task 1E: Confusion hi confusion hai !!

Analysis Report (techniques and results)

We tried several ways to make dense matrix multiplication faster:

- **Naive (ijk)** — The basic version, easy to understand but slow because it keeps re-reading data from memory.
- **Loop reordering (ikj)** — Changes loop order so A's values are reused more, cutting run time about in half for big matrices.
- **Loop unrolling** — Reduces loop overhead by expanding loops, but didn't help much and sometimes slowed things down; it doesn't fix the real memory bottleneck.
- **Tiling (blocking)** — Splits matrices into cache-sized tiles, keeping needed data in fast memory. Works as well or better than loop reordering at large sizes.
- **SIMD vectorization (AVX2)** — Does several operations at once, good for small matrices but less useful for large ones since memory access still dominates.

Summary: Loop reordering and tiling give the best improvements by reducing memory traffic. SIMD adds a boost when combined with these, while loop unrolling alone is rarely worth it.

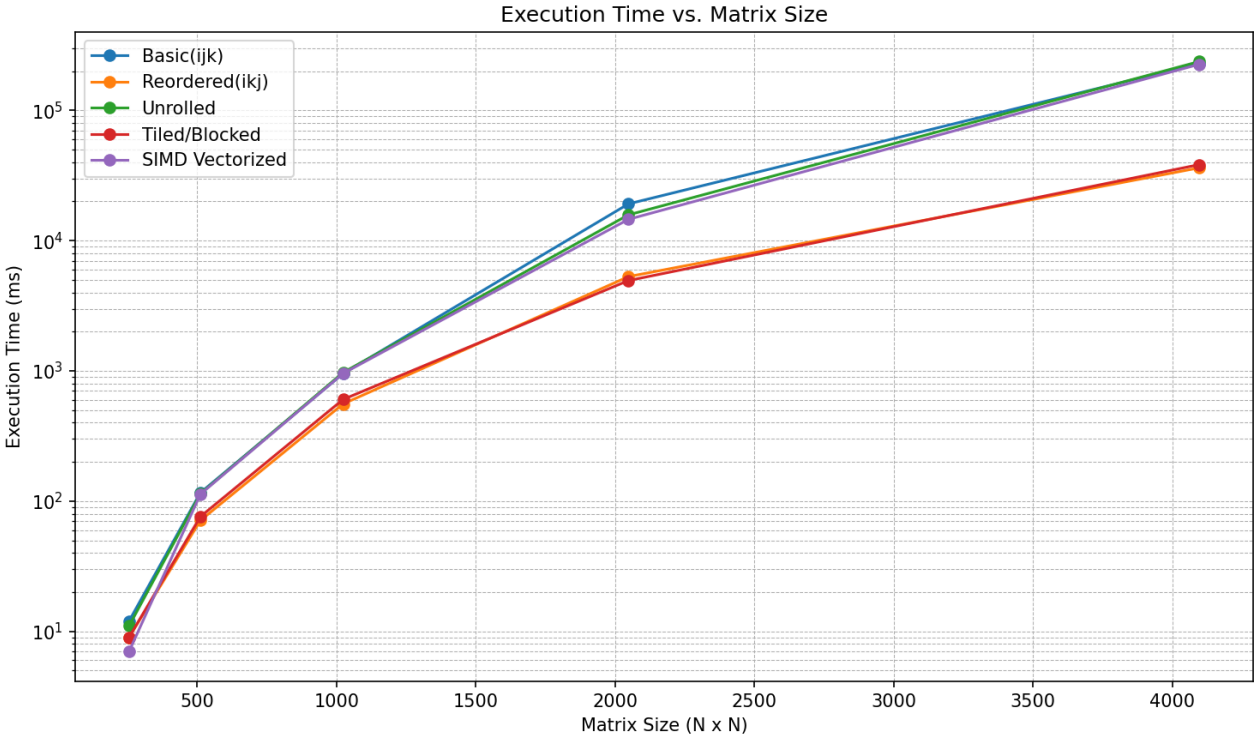


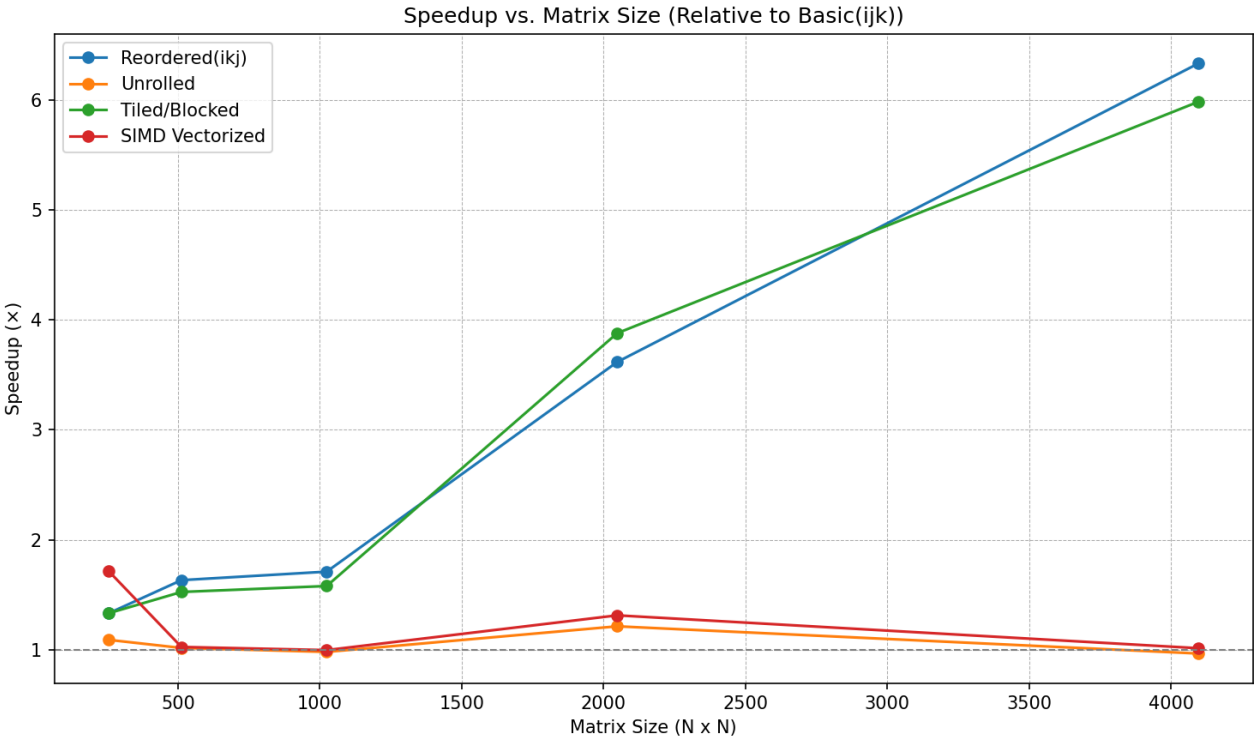
Comparative Analysis

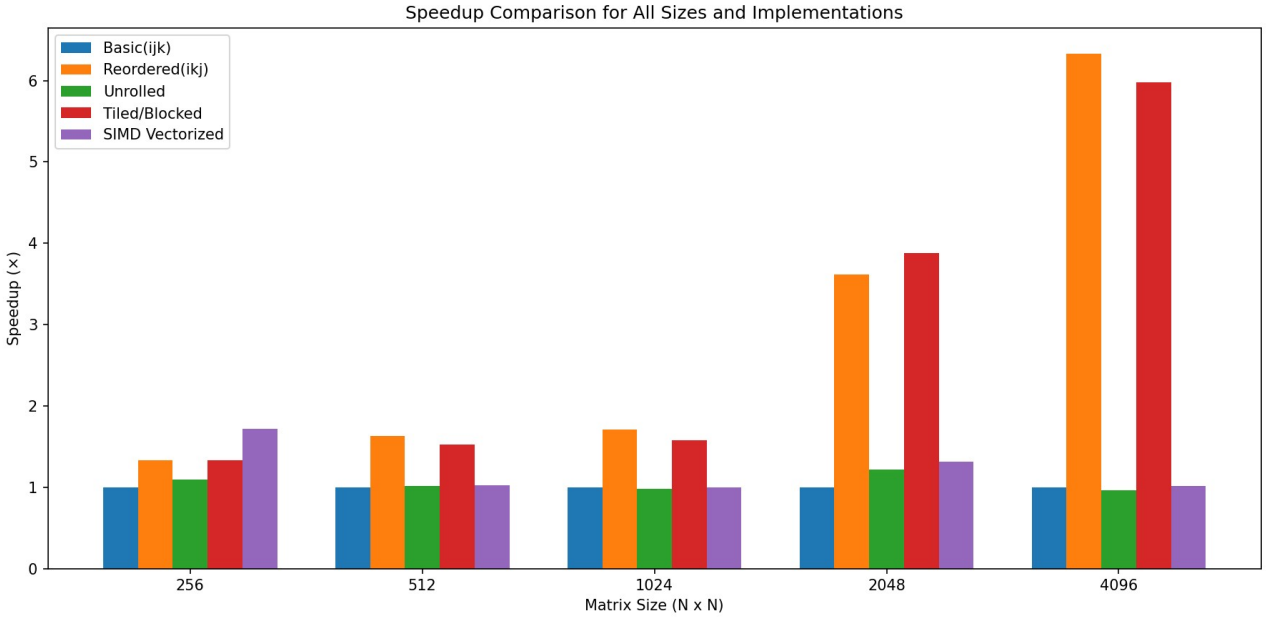
Absolute performance:
The basic (ijk) version is always slowest. For small matrices (256×256), SIMD is fastest because data fits in cache and the CPU can do several operations at once. For medium sizes (512, 1024), reordered (ikj) and tiled win by keeping data in fast memory longer. For very large sizes (2048, 4096), only reordered and tiled keep scaling; SIMD and unrolling don’t help since the CPU is waiting on memory.

Speed-up:
At 256: SIMD ≈1.7× faster, reordered/tiled ≈1.3×.
At 512–1024: reordered/tiled ≈2× faster; SIMD barely helps; unrolling can slow things down.
At 2048–4096: reordered ≈3–5× faster, tiled similar or better; SIMD/unrolled drop near the slow baseline.

Memory & cache:
The basic version wastes time re-reading B. Reordered reuses A better. Tiled reuses both A and B in cache-sized blocks. SIMD processes several numbers at once but doesn’t improve data reuse. Unrolling just reduces loop overhead. Reordered and tiled use cache efficiently and keep speed-ups at large sizes. SIMD helps only when data is already in cache; unrolling doesn’t fix the memory bottleneck.


Execution Time vs. Matrix Size
Basic(ijk)
Reordered(ikj)
Unrolled
Tiled/Blocked
SIMD Vectorized
Execution Time (ms)
10^5
10^4
10^3
10^2
10^1
500
1000
1500
2000
2500
3000
3500
4000
Matrix Size (N x N)


Speedup vs. Matrix Size (Relative to Basic(ijk))
Reordered(ikj)
Unrolled
Tiled/Blocked
SIMD Vectorized
Speedup (×)
6
5
4
3
2
1
500
1000
1500
2000
2500
3000
3500
4000
Matrix Size (N x N)


Speedup Comparison for All Sizes and Implementations
Basic(ijk)
Reordered(ikj)
Unrolled
Tiled/Blocked
SIMD Vectorized
Speedup (x)
0
1
2
3
4
5
6
256
512
1024
2048
4096
Matrix Size (N x N)