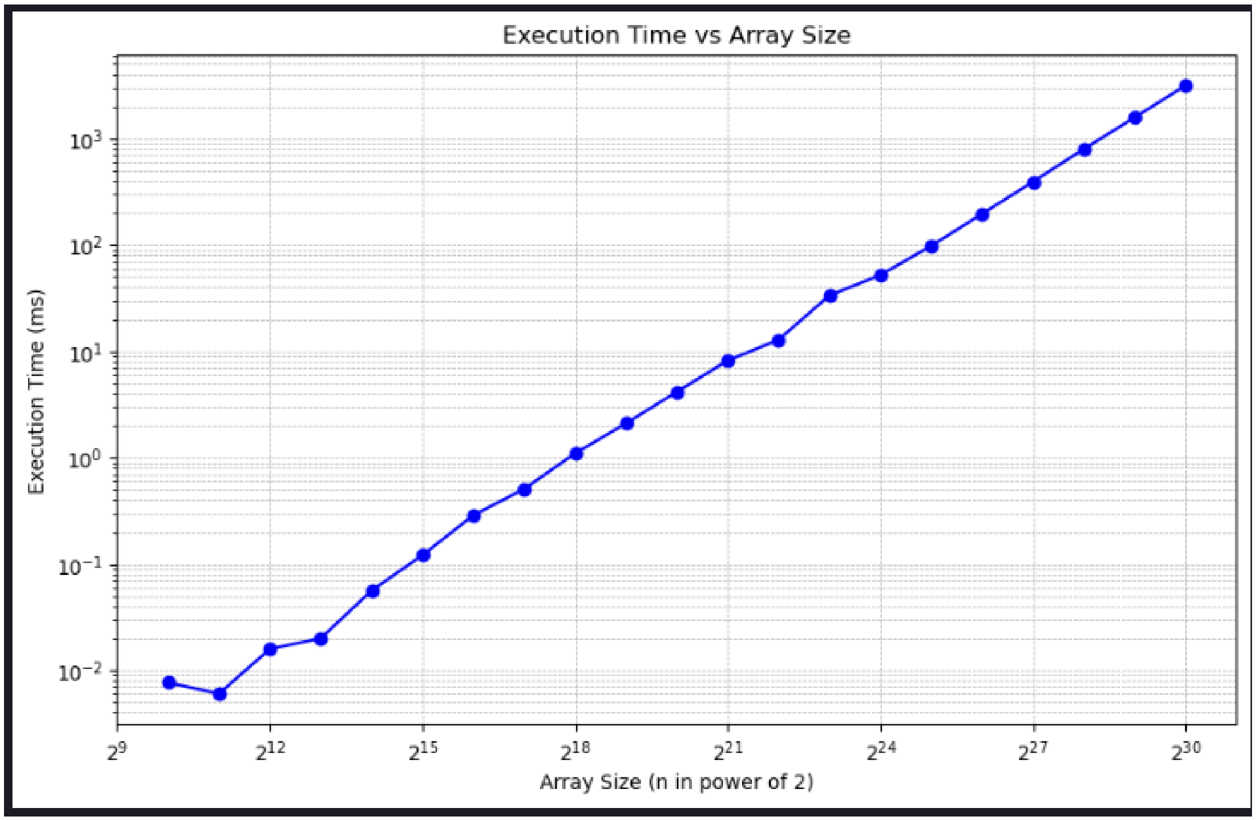
HW02

https://github.com/yosunlu/repo759/tree/main/HW02

Task 1:



Task 3:

With the performance order `2 < 1 ≈ 4 < 3`:

- `mmul2 (i, k, j)` is the fastest because the middle loop (`k`) allows accessing `B` in a row-wise manner, which improves cache locality, minimizing cache misses and making the computation efficient.

- **`mmul1 (i, j, k)`** and **`mmul4`** perform similarly. In both, accessing `B` column-wise (`j`) leads to inefficient, non-contiguous memory access, resulting in more cache misses. The overhead of `std::vector` in `mmul4` roughly cancels out the impact of the memory access pattern, leading to performance similar to `mmul1`.
- **`mmul3 (j, k, i)`** is the slowest because accessing `C` column-wise (`j`) for every `k` iteration causes inefficient memory usage. This poor cache locality results in more cache misses, thus significantly degrading performance compared to the other implementations.