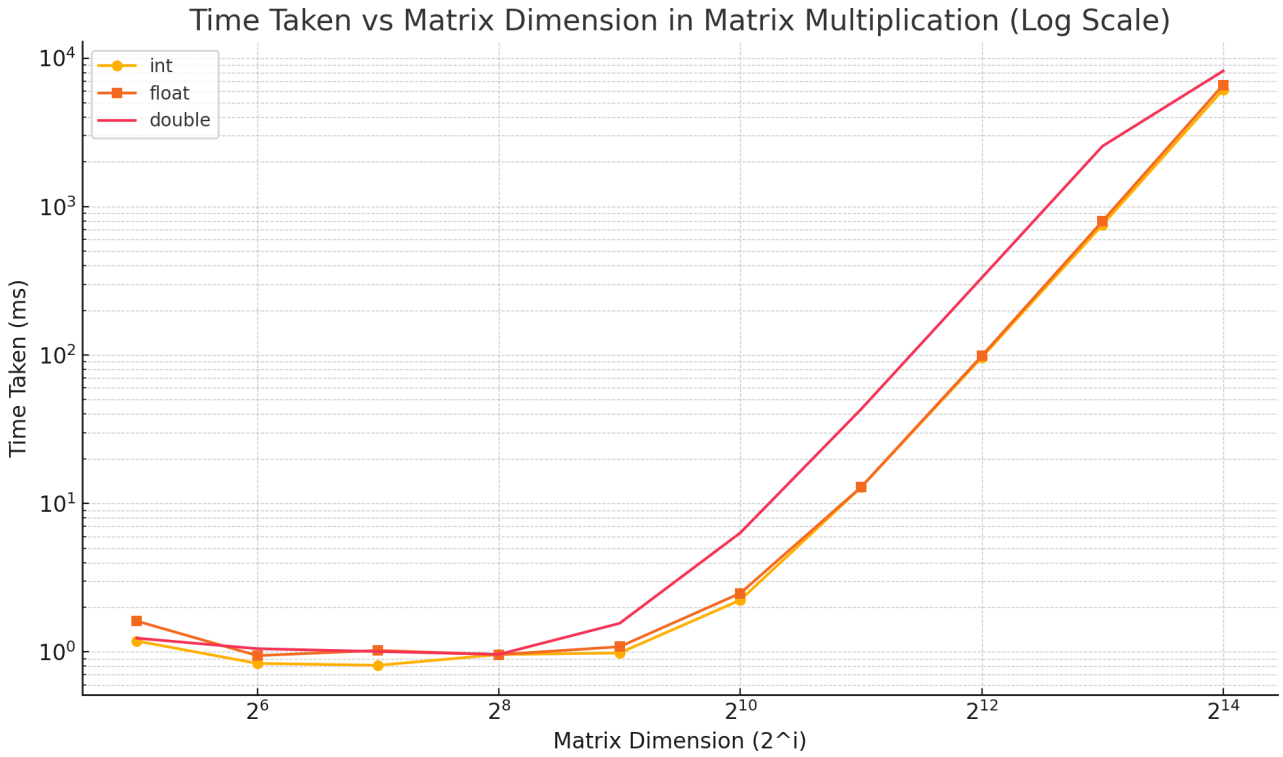
Assignment 7

https://github.com/yosunlu/repo759/tree/main/HW07

Task1

b) On Euler using Slurm, run task1 for each value n = 2^5, 2^6, · · · , 2^14 and generate a plot of the time taken by your algorithm as a function of n.



c) What is the best performing value of block dim when n = 2^14?

The best performing `block_dim` for n=2^14 in my implementation is **16**. Although a `block_dim` of 32 generally matches the GPU warp size, a smaller block size reduces shared memory usage, increases occupancy, and achieves better performance in this case.

d) Does the performance change depending on the type of the data (i.e. int, float, double)? Why do you think that is?

`int`: Fast for small matrices but slower for large ones due to simpler computations but lack of optimizations.

`float`: Best performance due to efficient single-precision support on GPUs.

`double`: Slowest due to higher computational and memory demands with limited optimization compared to `float`.

e) Present the best runtime for n = 2^14 from your HW06 matrix multiplication task. Explain why one of them (tiled vs. naive) performs better than the other.

- **Best runtime:**
  - **Tiled:** 6600 ms (float).
  - **Naive:** 12000 ms (float).

Tiled is better because it reduces global memory accesses by reusing data in shared memory, improving memory efficiency and throughput.

f) Present the runtime for n = 2^14 from HW02 (serial implementation mmul1) (or state that it goes beyond 10 minutes). Compare the performance between CPU and GPU implementations and explain why one of them is better. It is preferred that you use your own implementation for this comparison, but for those who dropped HW02, simply predict the better approach and explain why it is superior.

- **CPU Runtime:** > 10 min
- **GPU Runtime (tiled):** 6600 ms.

Why GPU is better: GPUs exploit massive parallelism, higher memory bandwidth, and shared memory optimizations, making them significantly faster than sequential CPU computations.

Task2

b) On Euler using Slurm, run task2 for each value n = 2^10 , 211 , · · · , 2^30 and generate a plot named task2.pdf with the time taken by your algorithm as a

function of N when threads per block = 1024. Overlay another plot which plots the same relationship with a different choice of threads per block.