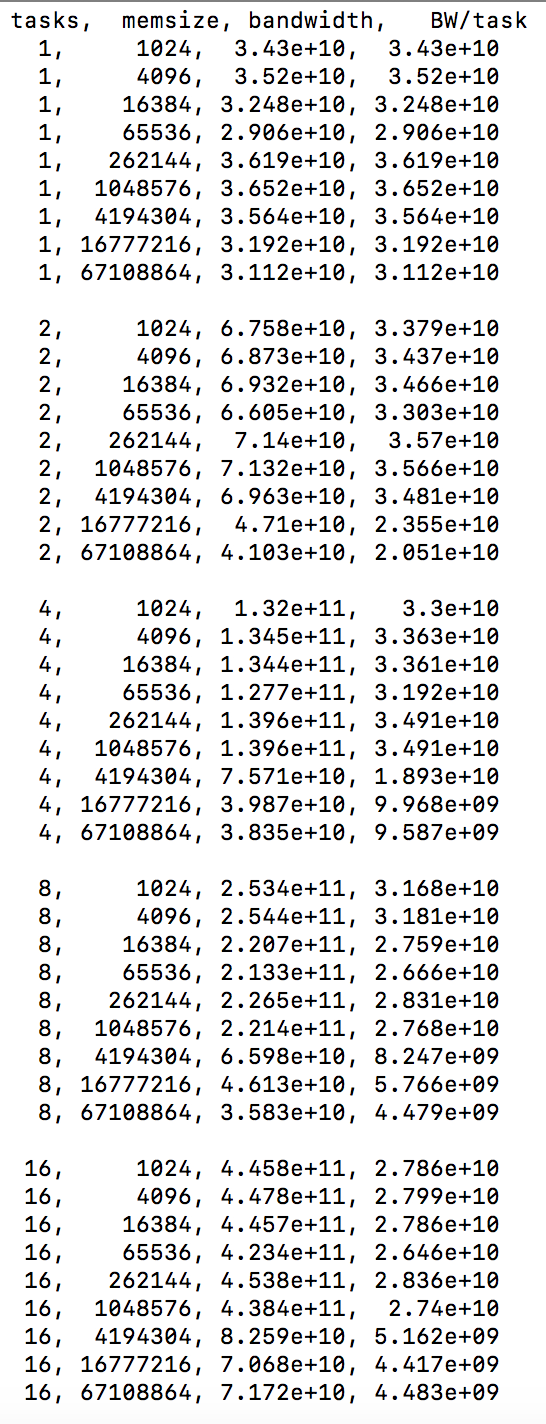
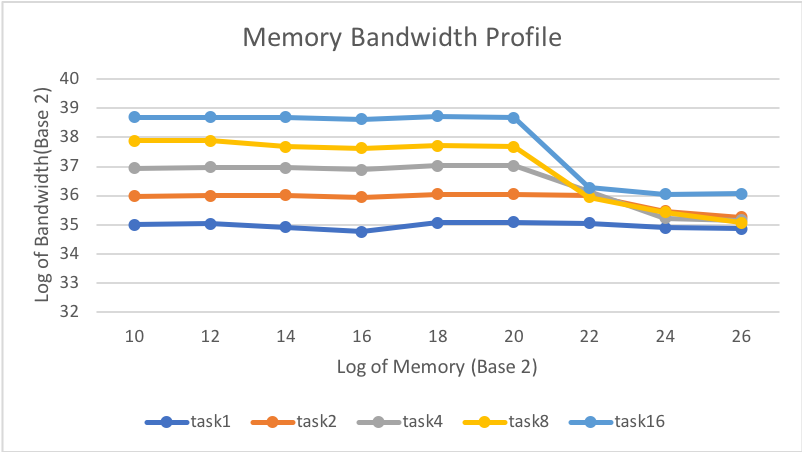
EC527 Assignment 1 Chen-Yu Chang U93093024

Part 1.





1a.

With only 1 task, memory of 1048576 bytes has the highest bandwidth.

1b.

With only 1 task, memory of 65536 bytes has the lowest bandwidth.

1c.

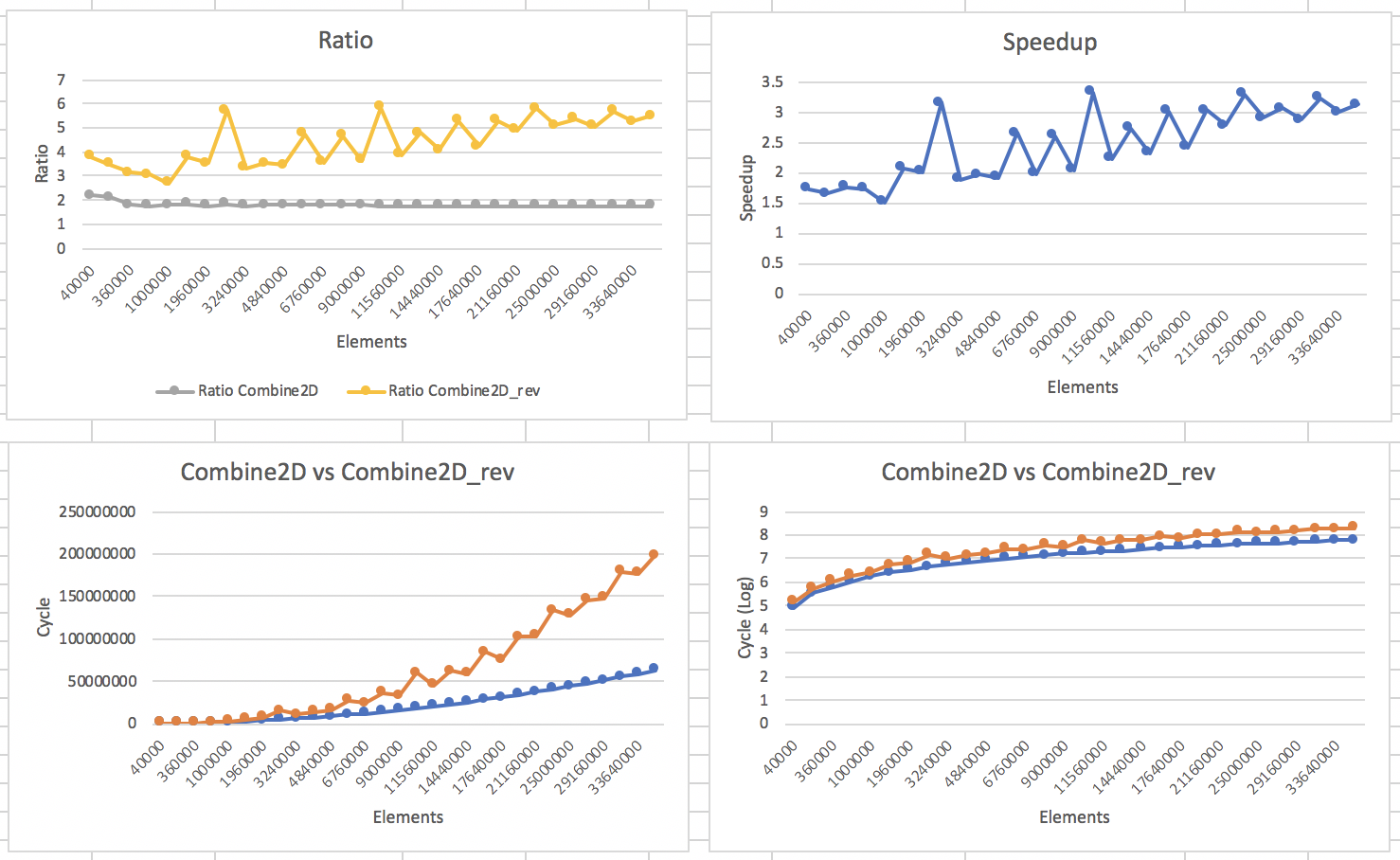
I got a highest bandwidth with 16 tasks and memory of 4096 bytes.

1d.

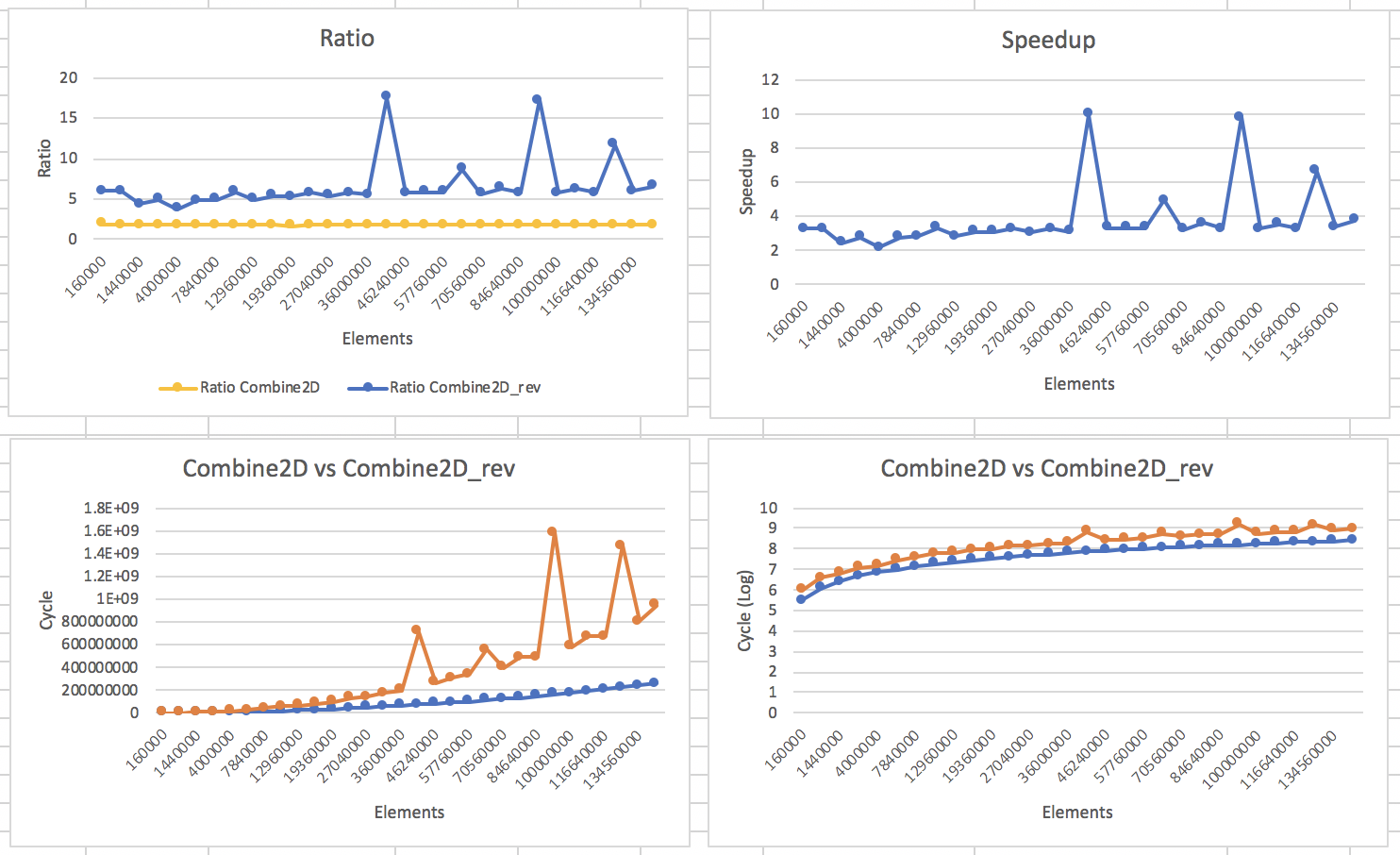
The closest bandwidth per task I got from my data is when running 2 tasks at the same time on the machine.

Part 2.

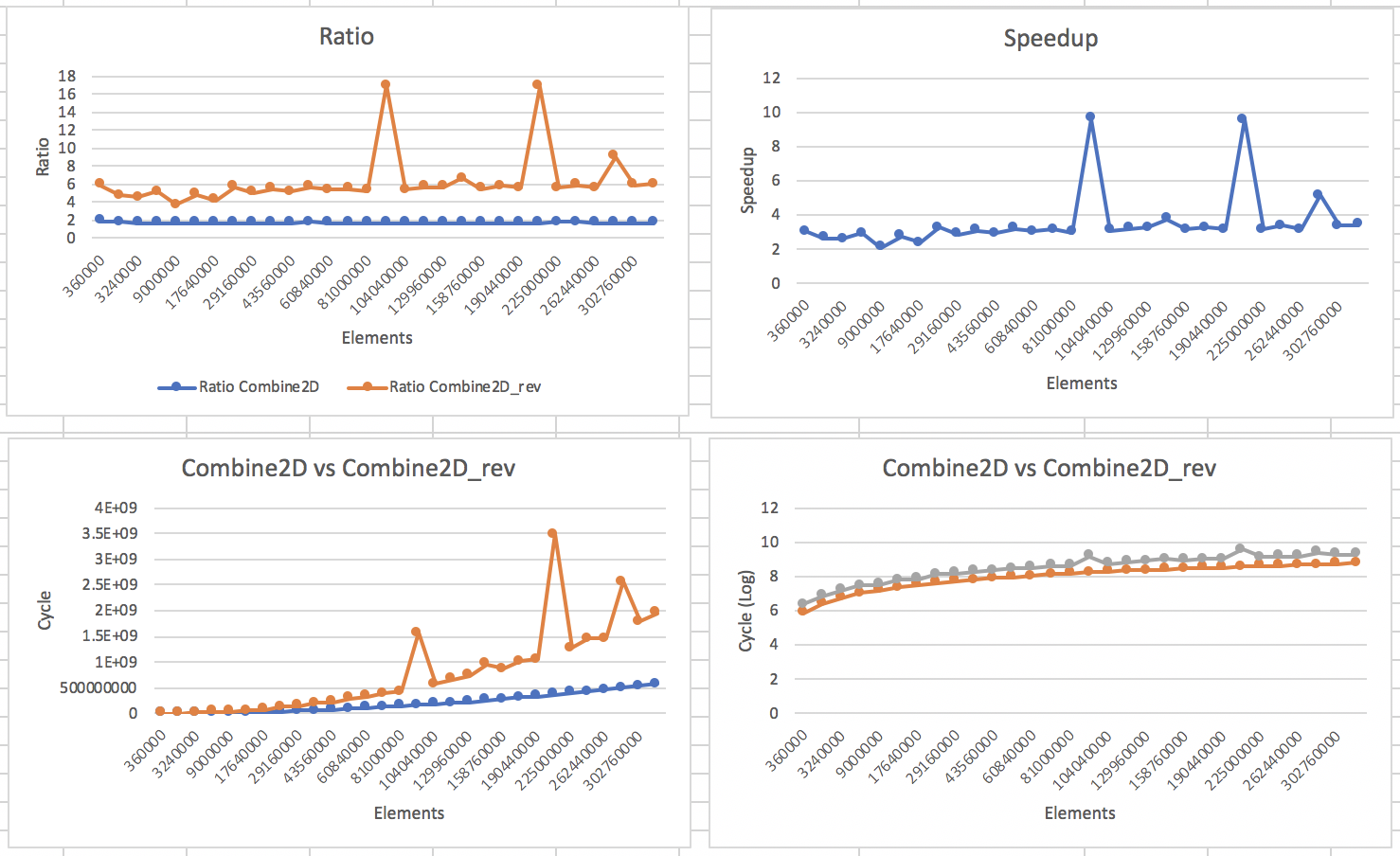
When BASE=0, ITERS=30, DELTA=200



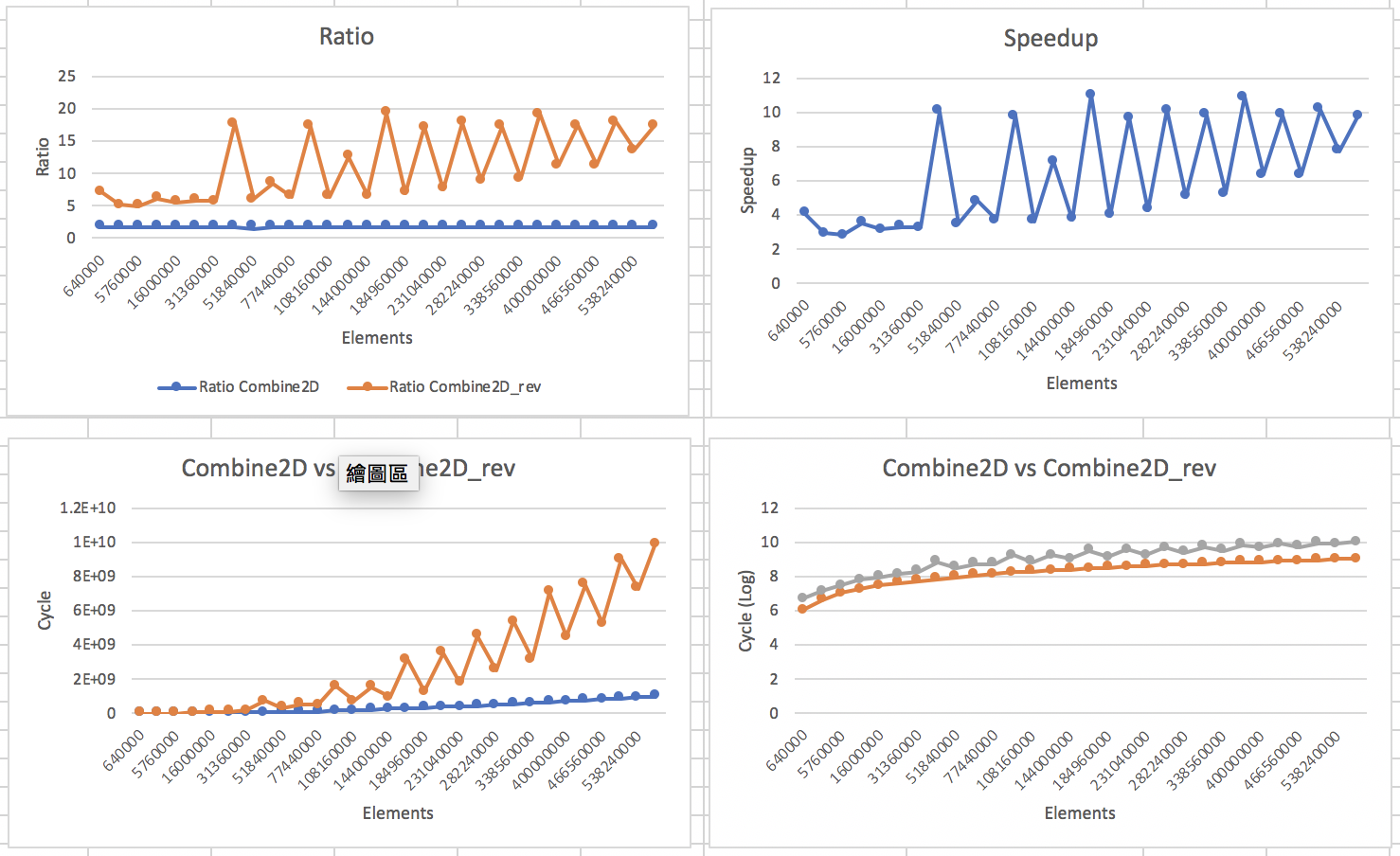
When BASE=0, ITERS=30, DELTA=400



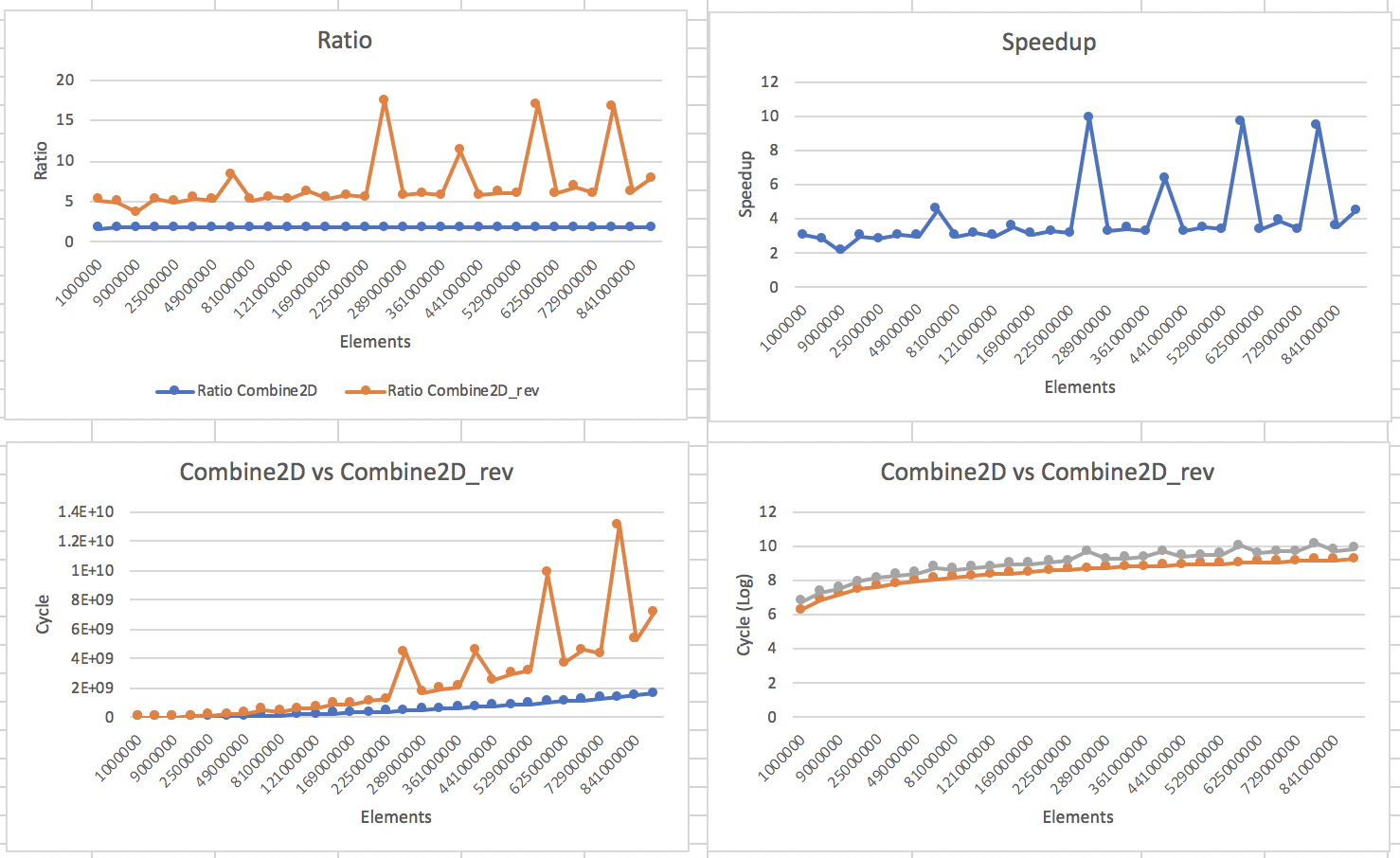
When BASE=0, ITERS=30, DELTA=600



When BASE=0, ITERS=30, DELTA=800



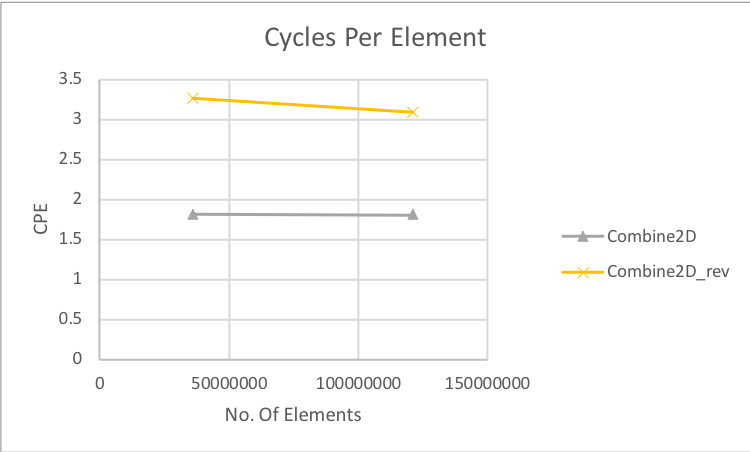
When BASE=0, ITERS=30, DELTA=1000



2a.

There is an execution problem after I assumed delta to be about 2000 due to failure of allocating storage. Overall, I found that Combine2D is a faster function by about 1.6 times. I found that the special spikes seem to appear for Combine2D\_rev when the row number of elements surpasses 6000 owing to some locality problems.

2b.

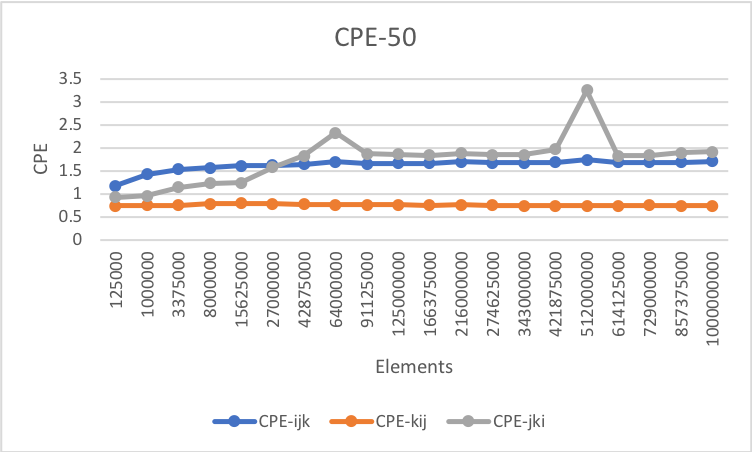


Cycle per element for Combine2D is 2.9734 when the width of the element is 11000, while Combine2D\_rev is 5.1945. Therefore, Combine2D\_rev has a higher CPE than Combine2D because Combine2D\_rev needs to change variable every iteration whereas this makes it more complex to operate. Overall, Combine2D is a more efficient function.

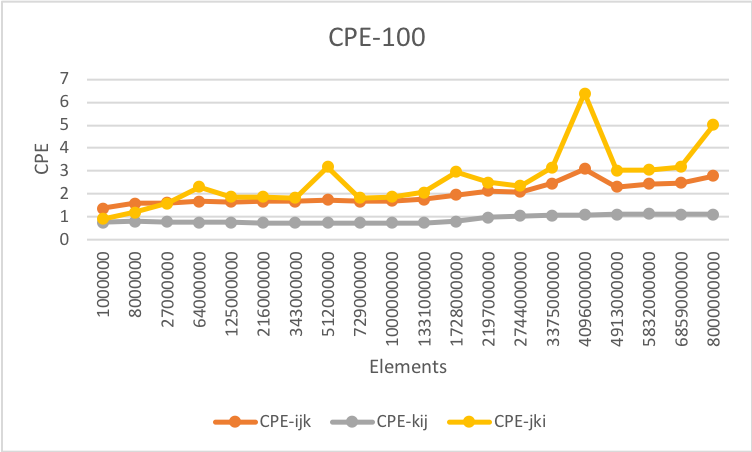
2c.

The part d graph may improve for a short time because there are spatial or temporal locality in the function while after the low tide, it will follow by a spike when new cache block is brought in. In part e graph, it will be jumpy for Combine2D\_rev since this function requires the access of memory more compared to Combine2D.

Part 3.



1. There are 0 plateaus for ijk and kij, but there are 2 plateaus for jki.
2. Jki: at point 64000000, the cycle per iteration is about 2.334 and at point 512000000, the cycle per iteration is about 3.265.
3. Jki: Transition at 64000000 and 512000000

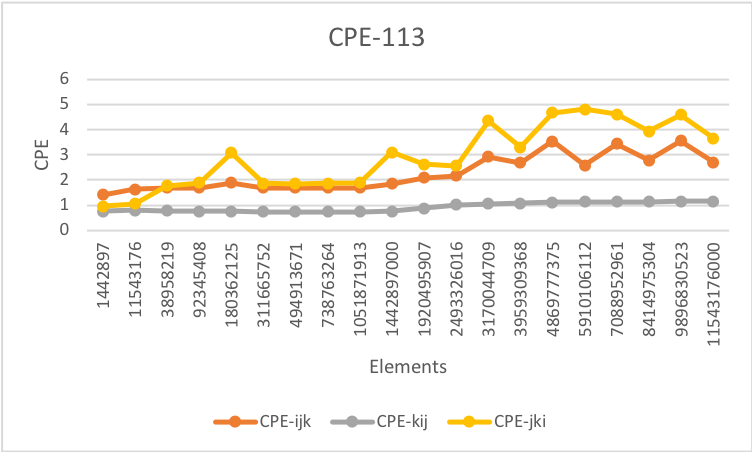


1. There are 1 plateau for ijk, no plateaus for kij, and 5 plateaus for jki.
2. Ijk: at point 4096000000, the cycle per iteration is about 3.1.

Jki: at point 64000000, the cycle per iteration is about 2.307. At point 512000000, the cycle per iteration is about 3.182. At point 1728000000, the cycle per iteration is about 2.952. At point 4096000000, the cycle per iteration is about 6.379. At point 8000000000, the cycle per iteration is about 5.

1. Ijk: transition at 4096000000

Jki: transition at 64000000, 512000000, 1728000000, 4096000000, and 8000000000



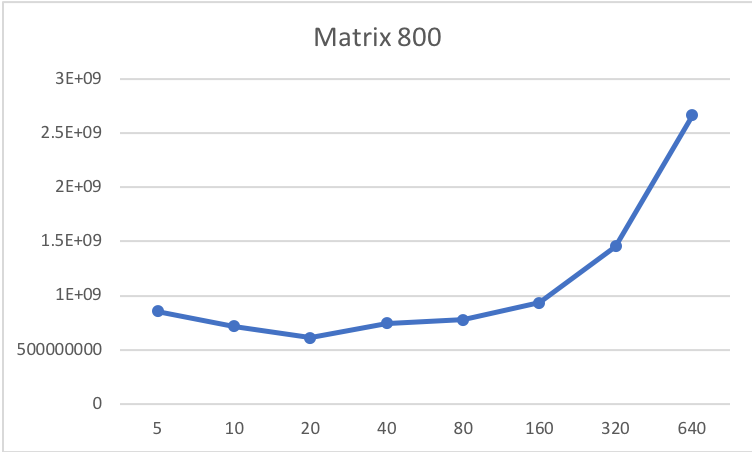
1. There are 4 plateaus for ijk, no plateaus for kij, and 5 plateaus for jki.
2. Ijk: at point 3170044709, the cycle per iteration is about 2.92. At point 4869777375, the cycle per iteration is about 3.537. At point 7088952961, the cycle per iteration is about 3.434. At point 9896830523, the cycle per iteration is about 3.566.

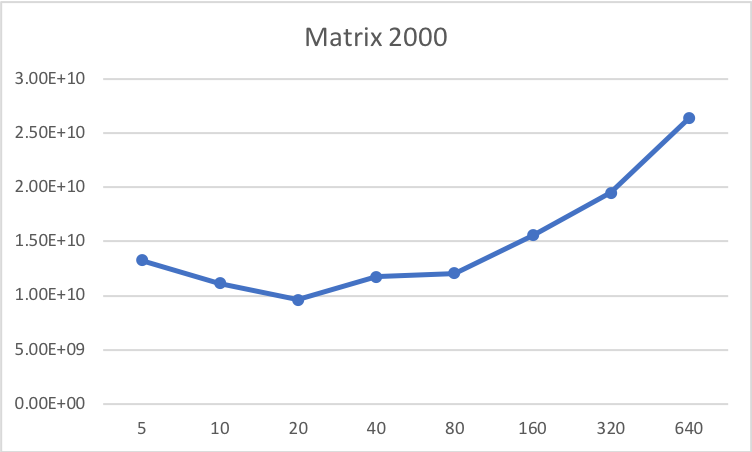
Jki: at point 180362125, the cycle per iteration is about 3.08. At point 1442897000, the cycle per iteration is about 3.106. At point 3170044709, the cycle per iteration is about 4.351. At point 4869777375, the cycle per iteration is about 4.661. At point 9896830523, the cycle per iteration is about 4.587.

1. Ijk: transition at 3170044709, 4869777375, 7088952961, and 9896830523

Jki: transition at 180362125, 1442897000, 3170044709, 4869777375, and 9896830523

Part 4.





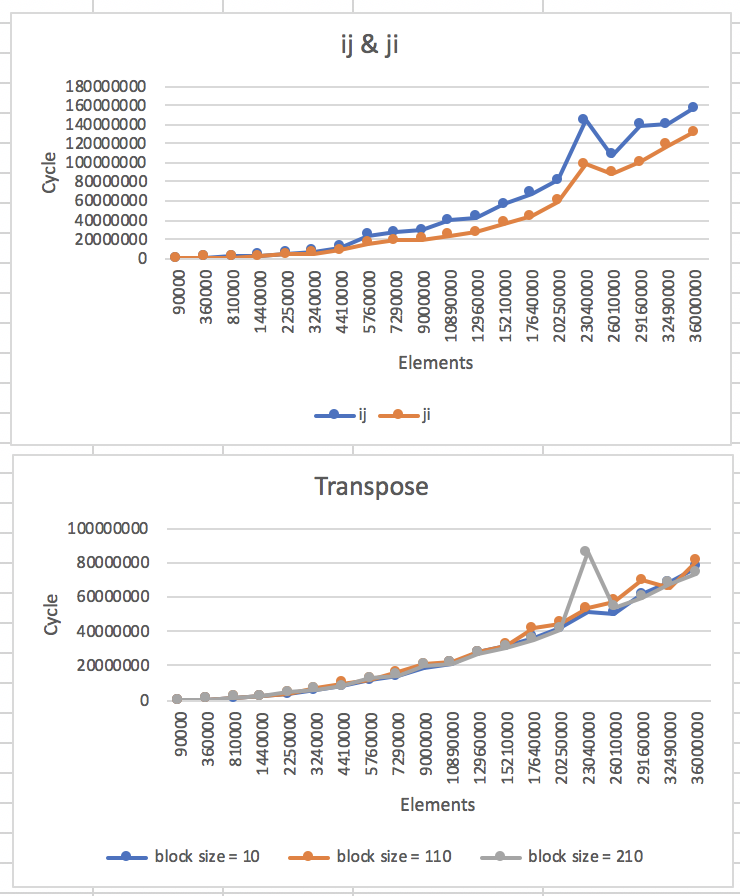
4a.

The benefit of blocking is the inner loop’s temporal locality. I tested the matrix size to be 800 and for each iteration, they have different time. According to the graph, I can see that the optimal block size is about 20.

4b.

Then, I tried to change my matrix size to be 2000. It turned out to be having similar optimal block size of 20.

Part 5.



From the graph, I can see that ij runs faster than ji. I also blocked into three different block sizes of 10, 110, and 210, while graph showed that larger block sizes are faster, comparing 10 to 110. Blocking is an efficient method because blocking can prevent from repetitively referring to the same elements every time to the cache. However, Larger blocks, like 210, are not as effective as it shows in the graph compared to 110.

Part 6.

6a. It took me about 9 hours.

6b. I spent most of the time on part 2 since it got more work and graph to work through.

6c. No.

6d. Not now.