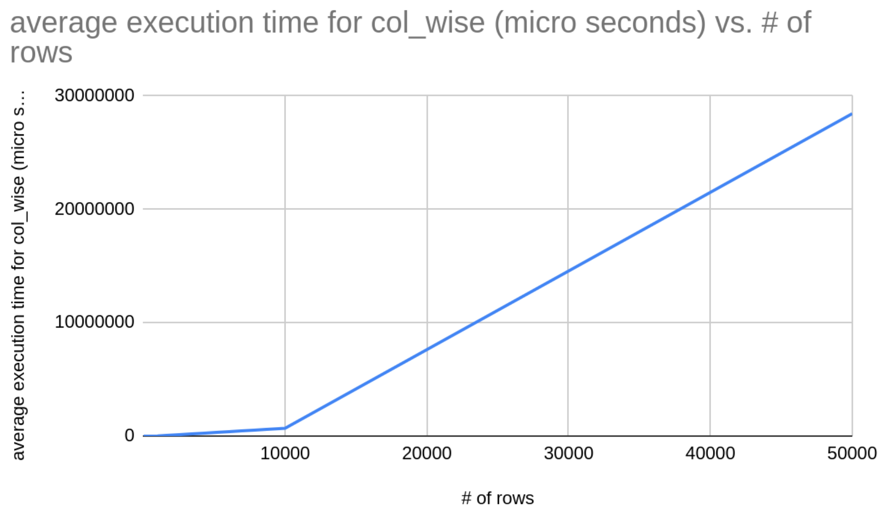
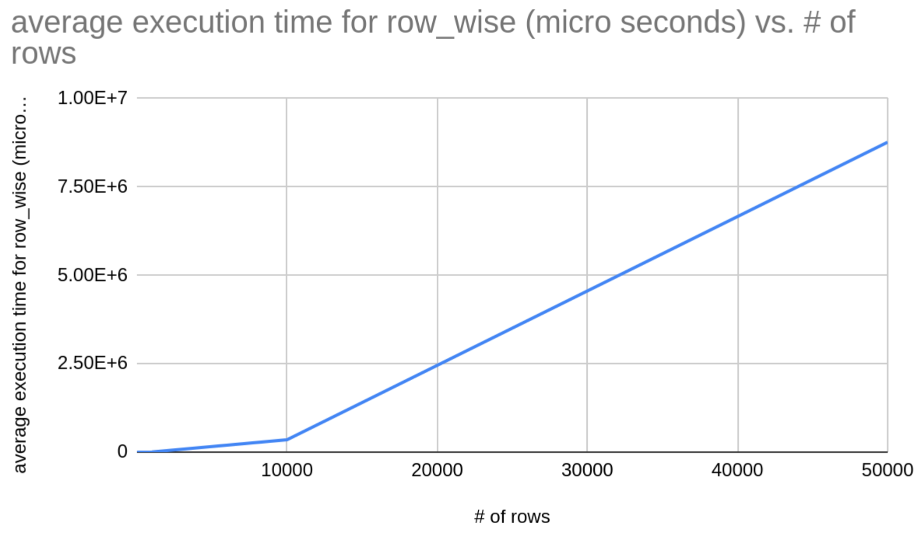
Part I

# of rows	average execution time for col_wise (micro seconds)	average execution time for row_wise (micro seconds)
10	1.1	1.6
100	37.8	44
1000	2799.5	3486.5
10000	679267.8	349216.8
50000	28435570.3	8766531.2





1. What is spatial locality?

   Spatial locality is a concept that if a program accesses a memory location, it’s likely that it will access a neighboring memory location. The addresses are also close to each other physically.

2. Based on your knowledge of spatial locality, which for-loop should perform faster and why?

   A row-wise for loop should perform faster. This is because traversing a row would allow for accessing elements with close memory addresses and they are more likely to be cached together because of spatial locality. But with column-wise traversal, the addresses are much more spread out and there would be a need to access the cache multiple times per iteration.

3. According to the plot how does the execution time behave when we keep increasing the array size (which for loop is faster)?

   For both row-wise and column-wise, the execution time increases as the number of elements in the array increases. However, column-wise execution times increased much more than row-wise at one point when the dimensions of the array were 1000x1000. The row-wise execution is faster.

4. Is the answer you gave for 2 the same as the answer given for 3?

   Yes it is the same as the answer for question 2. This is expected as the row-wise traversal will need to have less cache fetch lines while a column-wise traversal will need multiple cache fetches.

5. Why do you think the execution time behaves the way it does? Explain.

   It is because of spatial locality. As mentioned previously, because row-wise traversal would require less cache fetches than a column-wise traversal, the row-wise traversal should take less time for execution in the long run.

6. Why do you think we executed each implementation 10 times? Explain.

   There could be external factors affecting the performance time. For instance, silo crashed on me when running the execution because I was timed out, so my program just ran indefinitely. There’s also a bunch of other processes running as well, so the computer’s resources could be allocated differently than it was at the previous execution.

Part 2

1. Execute command lscpu in the terminal. What are the L1, L2 and L3 Cache sizes?

L1d cache:	768 KiB
L1i cache:	768 KiB
L2 cache:	12 MiB
L3 cache:	192 MiB

2. Execute command free -m in the terminal. What is the size of the RAM?

   515388 MB

3. Does the RAM or cache sizes have an effect on the execution time?

   Yes. The larger the RAM, the faster the execution time. If it wasn't too big, the CPU would have to keep accessing the hard drive to access data, which would slow down the performance time. With a larger cache, RAM won't have to be referred to as much so it'll increase the performance.

Part 3

	Col_wise execution		Row_wise execution	
# of rows	Cache hits	Cache misses	Cache hits	Cache misses
10	9562	4347	9549	4377
100	11,101	4397	11515	4512
1000	1,399,963	12945	230,053	6339
10000	277,334,382	116,272,711	23,005,511	128,294
50000	7,917,828,813	5,250,528,748	553,492,584	3,083,460

1. What is the command you had to use to get the cache hits and misses?

   perf stat -e cache-references,cache-misses <executable name>

   It gave me cache misses and cache references. But after searching online, it was said that cache hits could be calculated by subtracting cache misses from cache references. The e is a hardware event tag that allowed me to specify that I wanted to get the cache references and cache misses counts. I compiled the program every time before running the perf command.

2. What do the cache hits and misses indicate?

   They are used as a performance metric. They indicate how well a program is using the cache. When the cache hits, that means that the data that is needing to be retrieved was already in the cache. This is a lot faster than trying to access the data from the main memory. It's better when there are more cache hits than misses because that tells you that you are using the cache effectively. A cache miss is when the data was not already in the cache. Therefore the data has to be accessed via main memory, which is much slower than accessing the data through cache. In the table above, we can see that as the size of the dimensions increases for the column-wise traversal, the numbers of cache hits and misses are much closer to each other than for the row-wise traversal, which indicates that the row-wise traversal is using the cache more efficiently than the column-wise traversal.

3. What is the relationship between cache hits, misses and spatial locality?

   Spatial locality is a concept that, when used by a program, more than likely has the desired data in the cache. This means that there would be many more cache hits than misses. The cache also uses spatial locality because it stores neighbor memory addresses together in the cache. If there are more cache misses than hits, this implies that the data that needs to be retrieved is likely not in the cache.