Homework 2

# Problem 1

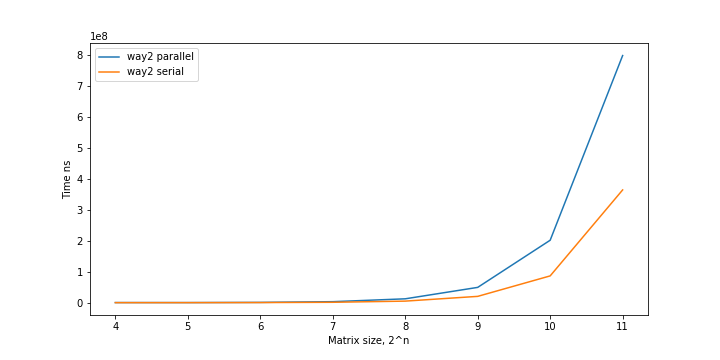
## /Users/zhuochen/OneDrive/17Spring/ECE5720/homework/ECE5320/HW2/way1.png

## Results

The graph above described the performance of multithreading verses single thread. When the size of the matrix is small, which is less than 2^9, single thread outperformed multithread with very small advantage. However, when the size exceeded 2^9, time cost by single thread is growing exponentially with the size, while multithread is still close to linear.

When the data size is small, multithread method’s performance is delayed by the overhead caused by the scheduling and some other factors. However, when the task size achieved a certain size, parallel method will have oblivious advantage. Also, the performance decreasing of single thread when the data size achieved a large value should also be caused by the data cannot fit in the core cache. When the task is parallelized, the data could be able to fit in that core. It is expected to see that when the matrix size achieved a certain large value, the thread data cannot fit in the core cache, the performance will decrease exponentially like single thread.

## Problem 2



## Result

The result of the second way is plotted in the graph above. The result shows that the OpenMP paralleled algorithm is slower than the serial version. The performance of way 1 and way 2 both starting decreasing around 2^8 and 2^9, which may indicate that the core cache can only fit the matrix (could be part of the matrix) with size around this value.

The parallel method was expected to outperform the serial method as it has more computing resource. However, the OpenMP version is slower than original merge sort. Some possible explanation for this could be:

1. The data size is not big enough to overcome the overhead leads by parallel method.
2. The sorting method for this problem is merge sort, OpenMP may not optimized for this method so the performance is not good.
3. The method is poorly implemented.