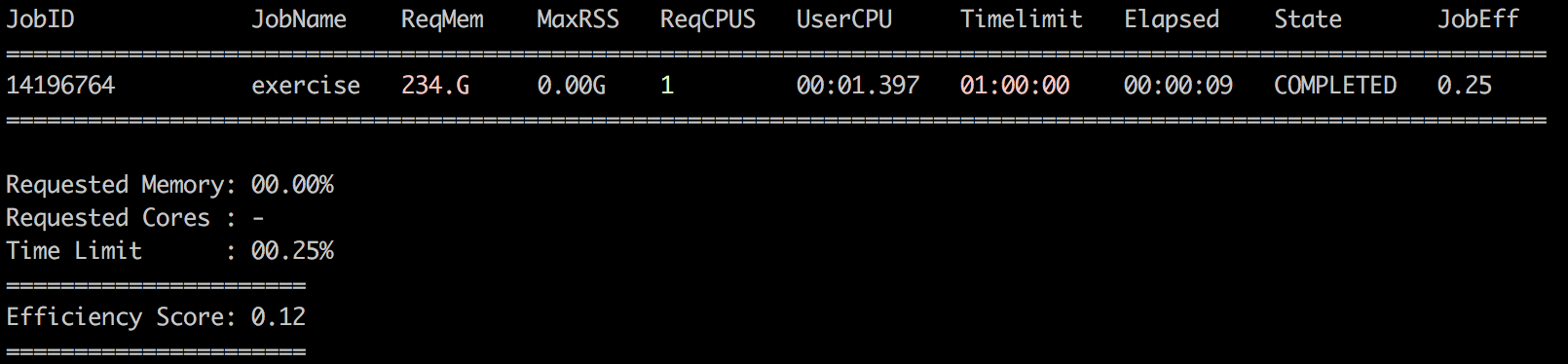
Tyler Thatcher

INF503

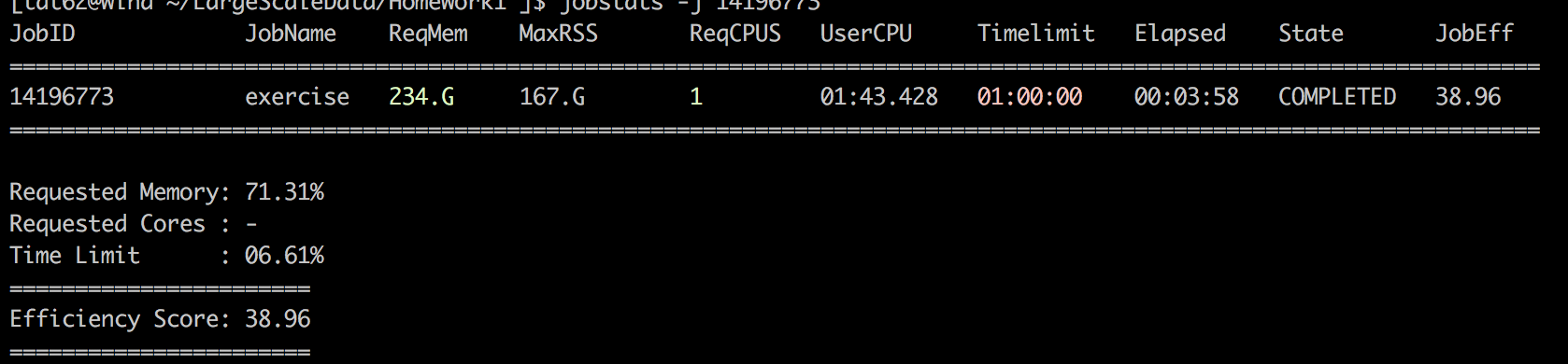
HW#1

Homework #1

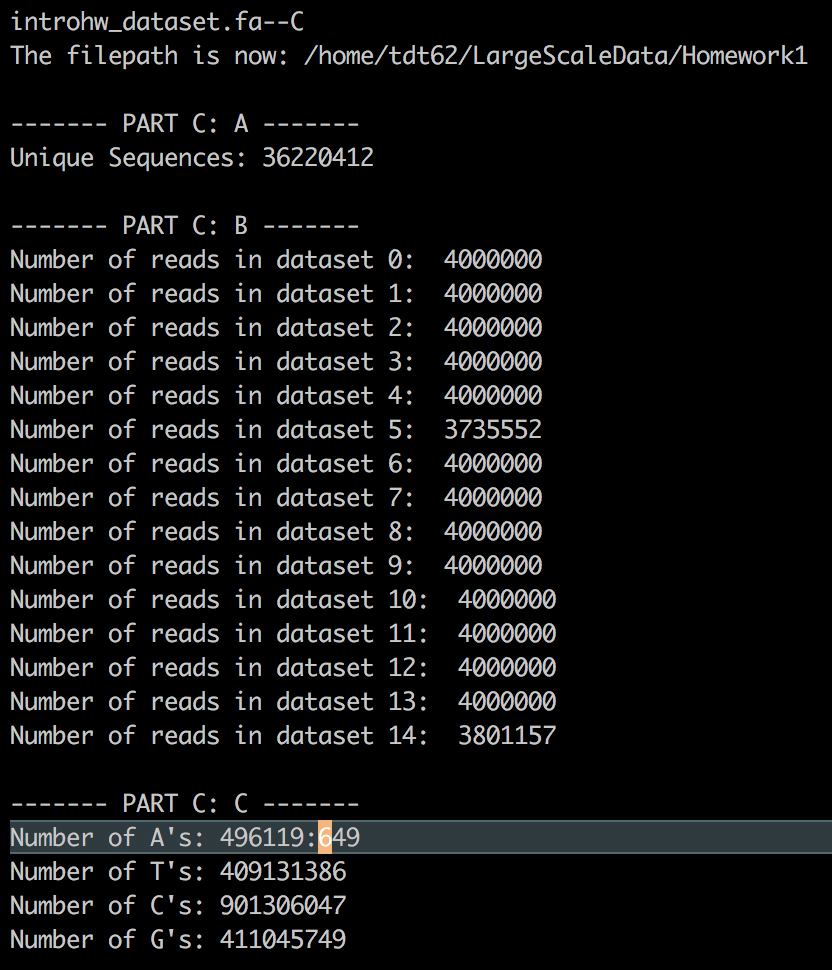
1. The screenshot below is what jobstats produced for allocating 1 million items in an array. It showed practically no memory being used, not even 1/100 of a GB. I am going to assume that the actual memory is going to be way higher. Something like 36 million / 1000 (MB). My guess is around 36000 bytes or ~360 GB of RAM. As for CPU, I would guess about 10 times the amount of usage, so 00:10.397.



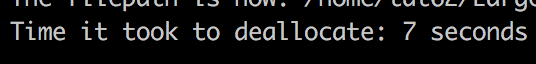
1. The screenshot below is what jobstats produced for allocating the entire text file given to us (i.e. 36 million). I expected way more RAM to be used, and I also expected a lot more of the CPU. That being said, I was not accurate at all. I thought it would scale from raw RAM/CPU time. My guess is that C++ does some sort of optimization in the background, which caused this discrepancy.



1. The screenshot below shows the stats for Part C.



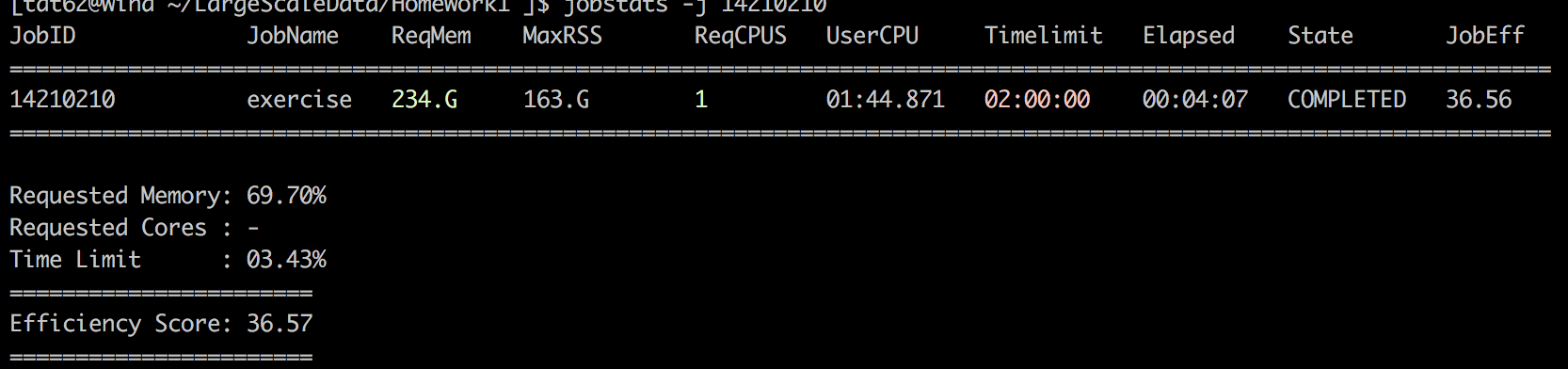
1. The total time it took to deallocate my 36 million array was 7 seconds. I assumed it would be fast because it is just deleting memory. Below is an output of my code:



1. I used insertion sort for this problem, for some reason I thought insertion was much faster. The O(N) is O(N^2) which is the same as bubble sort. For this reason, I had to reduce the 36 million to about 10000 to get it to work correctly, and in a timely manner.

Below are the screenshots of the sorted array and jobstats of the 10000 items. With the jobstats, it took 4 minutes to store and sort 10000, to do that same thing I would guess it would take around 3600 times longer which is about: which is about 240 hours. This is not doable, and the sorting method should be much faster (merge / quick sort).

* Jobstats



* Sorted Array

