CSSE230: Sorting Races

# Name(s):Yilun Wu

# Part 1: Data

Table 1 shows the runtimes of 6 sorts for at least 4 different types of arrays:

N=1000000 . Time is all in MS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sort | Unique | Almost Sorted | Almost Rev Sorted | Random Int |
| Merge | 11 | 13 | 26 | 479 |
| Quick Sort | 100 | 11 | 12 | 124 |
| TreeSet | 1055 | 259 | 240 | 2042 |
| My heap | 250 | 260 | 227 | 700 |
| My Quick | 53 | 51 | 76 | 162 |

# Part 2: Discussion

Include your discussion of the runtimes in Table 1, as described in the specification.

I don’t know why in my codes, the unique is much faster than random.

Merge sort is strict nlogn. But it use more memory so it is slower than quick sot.

Quick sort is nearly nlogn but it use less memory, and I use random to avoid o(N^2) in almost sorted.

The constant number of TreeSet is much bigger than other. And it need more time on handling duplicate. So it is slower than other.

The constant in heap is a little more than merge and quick sort. Each time, we need to compare two int is not next to each other in array. It is not good for cache.

My quick sort is of course slower than Java’s! No doubt! XD

I think there are lots of reasons are cache. TAT