Sorting Competition

Group 5 (Matt & Kyle) & 18 (Outside Source)

Results

Group 5 - (Matt & Kyle)

- 12th place (class)
- Times: 13471, 13470, 13470 ms
- Correctness: Yes.

Group 18 - (Outside Source)

- 1st place (overall)
- Times: 85, 108, 100 ms
- Correctness: Yes.

Group 5 - Algorithm Description/Efficiency

Description: Didn’t change much; used an integer version of product of prime factors to save space when input was shorter. Used the default Arrays.sort() method (timsort).

Did not precompute any values and simply called product of prime factors when comparing any two input values.

Source of complexity: Arrays.sort() (Sorting objects with comparator -> Merge sort - O(n lg n))., productOfPrimeFactors() - O(sqrt(n)). Worst Case = O(n lg n). Note that the constant factor to productOfPrimeFactors is quite large since it is being called at each comparison.

Group 18 - Algorithm Description/Efficiency

Description: Group 18 takes a precomputed list of 1000 prime longs, and uses this list to roughly sort the original list by doing a binary search with each computed product of prime factors. They then put the list of prime products into a treemap, where the key-value pair = (product, original array index). They then iterate through the key value pairs and place them into a new hashtable, where the key-value pair = (product, sorted array index). This new pairing is used to place the original elements (roughly) in their correct place in the sorted array. From here, they sort a subsection of the partially sorted array with Arrays.sort() for each element added. Note that when computing product of primes, they used the Pollard Rho algorithm. This algorithm uses RNG to increase the performance of computing the primes.

Group 18 - Algorithm Description/Efficiency Cont.

Sources of Complexity:

Calculating Product of Primes:

rho() - O(sqrt(n)), getprime() - O(n), productofprimefactors() - O(n) -> Worst case O(n).

Sorting the Partially Sorted Array:

Arrays.sort() - Worst case O(n lg n), expected case (since array is sorted) O(n).

Overall Runtime: Worst case O(n + n lg n), expected case O(n + n) -> O(2n) -> O(n) (when the array is partially sorted).

Additional notes: The data structures used (hashtables) allowed for O(1) operations when looking up values.