Factual

1. What happens when a Java file is compiled? HINT: Think about `.java` and `.class` files.

2. Why can a block of code run faster the second time it executes as compared to the first time?

3. What's the optimal way to sort an array of integers? Assume the integers are uniformly randomly distributed.

⌄ Problem 1

When a Java file is compiled, it is transformed from its human-readable `.java` source code format into byte code in a `.class` file format that can be executed by the Java Virtual Machine (JVM).

⌄ Problem 2   #

If some segment of code is called many times, the interpreter studies and re-implements your code based on what is observed while the code is running. For example, if you create many unused data structures, they might be optimized out of your code.

⌄ Problem 3

As we saw in lecture, LSD sort outperforms quicksort for an array of integers, particularly if we optimize for a better base than base 10.

Metacognitive

1. Under what conditions would LSD sort be faster than mergesort?

Problem 1

There are two main cases when LSD sort is much faster than mergesort: when there are a large number of strings (large N), and when all strings are very similar to each other.

When N is really large, we see the asymptotic behavior of LSD sort beat the asymptotic behavior of merge sort, which is slower with a runtime of $N \log N$.

When the strings are very similar, then each comparison in merge sort is slow, roughly linear time in the length of the string. Instead of comparing each letter in each string at every merge, LSD only examines the letters of each string once.

Last updated 1 year ago