Lab 13: Linear Sorting (It's possible!)

This week in lab you're going to be writing Counting sort and Radix sort.

Counting Sort

CountingSort.java

Counting sort is a special sort of sort where we have the restriction that the sortable elements have to be able to be converted into numbers.

In this sort, we simply count the number of occurrences of each value in the array and then go through this counts array in order and fill in the sorted array with the number of counts each value has.

However, standard implementations are unable to handle negative numbers. Look at and try running `CountingSortTester` and you'll see that the provided `naiveCountingSort` cannot handle an array with negative numbers.

Fill in the `betterCountingSort` method so that it still does a counting based sort, but also handles negative numbers gracefully.

For fun (optional): Add a test to `CountingSortTester` that causes your `betterCountingSort` to fail.

Radix Sort

In this part of lab you'll write an implementation of radix sort for ASCII Strings. Normally, if we just had decimal numbers, we would say that we would have a radix of 10 (R = 10) since there are 10 possible digits at each index, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]. It is important to note that the time Radix Sort takes does depend on the length of the longest value it has to sort. We consider running Radix Sort to be linear time for integers in Java because the number of digits allowed for an integer is limited (10 digits max) which means we will at most have to do 10N iterations.

For our purposes in this lab, we are going to be sorting ASCII Strings which have 256 possible characters (numbered 0-255) and are of variable length. In JAVA, you can get the ASCII code for a character by casting the `char` as an `int` (`int i = (int)'a'`) and get the character from the ASCII code by casting the other way (`char a = (char)97`). You may implement either MSD (most significant digit) or LSD (least significant digit), but one is significantly easier (think about how you would sort words in your mind).

Since we have 256 characters to use, we have a radix of 256 (R = 256). Write the method 'sort' in `RadixSort.java` that will sort the list of ASCII Strings that is passed in and return the sorted list. Make sure the method is NON-destructive (so the original list cannot change). Feel free to add any helper methods you want (you can also use your counting sort implementation). Here is a great tool for seeing how Radix sort works visually.

Keep in mind that Radix Sort on Strings runs in `O(N*M)` time where `N` is the number of Strings and `M` is the length of the longest String. HINT: Remember ASCII codes start from 0, not 1.

Extra for experts (optional): Compare the runtime of your Radix sort compared to `Arrays.sort` . Which is faster for short arrays? Long arrays? Do the values in the array matter?

Submission

Submit zip file with `CountingSort.java` , `CountingSortTester.java` , `RadixSort.java` , and `MagicWord13.java` .

Note the MagicWord has been provided for free, since we will not cover counting sort untli the day of the lab.