HW 19: Sorting

Over the past two labs, you have been exposed to different sorting algorithms. Each of these sorting algorithms has their own advantages and flaws. Except perhaps for bubble sort, which is bad news bears.

1 Description

You are given an array of unsorted integers, and your job is to sort them from smallest to largest in an efficient manner. However, you cannot simply sort the array with a single algorithm, because a single sorting algorithm won't work optimally in all cases.

There are four variables that determine the sorting scenario:

1. $N =$ The total number of elements in the array.
2. $R =$ The total number of possible integer values. You can assume that the integers will be uniformly distributed between 0 and $R-1$.
3. $L =$ The degree to which the array is already sorted from smallest to largest. The two possible values are *randomized* and *almost sorted*. For an *almost sorted* array, there are on average 20 elements that are out of order.
4. $K =$ the number of elements we are interested in sorting. If $K = N$, sort the entire list. If $K = 1$, find the smallest element, and place it in the first index. If $K = 4$, find the four smallest elements, and sort them in increasing order.

Below are the possible values for each variable:

$$
\begin{aligned}
N &= \{2^{20}\} \\
R &= \{2, 1024, 2^{30}\} \\
L &= \{\text{Random}, \text{Almost sorted}\} \\
K &= \{1, 32, 2^{20}\}
\end{aligned}
$$

Your algorithm should account for any combination of values for N, R, L, K. In total, there are $1 \times 3 \times 2 \times 3 = 18$ possibilities.

2 Your Task

There are two main components to this homework:

1. Implement the following sorting algorithms in `MySortingAlgorithms.java`.
   (a) Insertion Sort
   (b) Selection Sort
   (c) Merge Sort
   (d) Radix Sort

   All of these classes implement the `Sorting` interface. You are allowed to modify the array passed into the `sort` method. Note that when $K < N$, you need to return an array of size K that contains the K smallest elements of the original array in sorted ascending order.

2. Implement the method `chooseSortingAlgorithm(int[] ary, int k)` in `MyAlgorithmSelector`. Given an array `ary` with size *N* and a query to find the smallest *K* elements, determine the proper sorting algorithm from part 1. Since *R* and *L* are not given, they will need to be inferred from `ary`.

3 Notes

1. `System.arraycopy` will be useful for copying entire arrays.
2. There is a sorting algorithm `JavaSort` in `MySortingAlgorithms`. You may use this for reference, but **do not** use `JavaSort` or Java's built-in `Arrays.sort` method for your your `MyAlgorithmSelector`.
3. All the files are located in ~cs61bl/code/lab26.

4 Submission

Submit `MyAlgorithmSelector.java` and `MySortingAlgorithms.java` as `hw19`