CS 445 Lab 11: Advanced Sorts

Introduction

In this lab, you will implement a base case improvement for quicksort. Your TA will overview these sorting algorithms and the optimization.

Exercise

After the TA's lesson, complete the following steps:

1. Download the provided code and read it over.

2. Run `SortTiming` to determine the size at which insertion sort becomes less efficient than quicksort. Recommended parameters are 1000 trials, ranging from size 5 to 100 with step size 1.

   Hint: Use output redirection to save the output of this program to a text file with a `csv` extension. This text file can then be opened in a spreadsheet program, such as Excel, enabling you to plot graphs for your results. Use a command such as:

   ```
   java cs445.lab11.SortTiming > data.csv
   ```

3. Implement the method `timeQuickSort2()` based on `timeQuickSort`. For the sort itself, you should use the quicksort wrapper that allows you to specify the base case size, and then call insertion sort to "smooth out" the resulting array. Use a base case for which insertion sort is clearly faster than quicksort.

4. Uncomment the call to `timeQuickSort2()` in `main()`, and re-run the experiment for a larger range of sizes. Compare the runtime of quicksort vs. your improved version of quicksort. Suggested parameters are 1000 trials, ranging from size 10 to 1500 with step size 10. Draw a plot of these results to evaluate your optimization.

Conclusion

In this lab, you wrote an optimization for quicksort, increasing the base case size and using insertion sort for the smallest subarrays. While this is not an asymptotic improvement, it can be a significant practical movement.