Group Assignment 3

CS325 Summer 2019

Due: Saturday, August 10 at 11:59PM

You are encouraged to work in groups of up to three students. Only one member of each group should submit the group's work to Canvas, including the ***project report as a pdf*** *and the* **code** *you wrote. The report should have* ***all member's names included****. You may use any language you choose for implementation.* ***No questions about this assignment will be answered after the due date above.*** I'm trying to keep this one short for you, so you have more time to study for the final.

Counting the number of Inversions in an Array

Consider the following problem: Given an array of n integers $A[0, 1, \ldots, n-1]$, find the total number of elements in the array such that for some pair of indices i and j, $i < j$; we have $a_j < a_i$. In other words, how many elements are out of order in the array (assume array should be in ascending order). You can think of this as how far from sorted the array is. For example if

$$A = [2, 4, 1, 3, 5]$$

then we would return a count of 3 inversions, for elements (2, 1), (4,1), and (4,3). You need not return the indices i, j, or the pairs of elements, just the total count.

You can solve this problem by enumerating over all $\Theta(n^2)$ choices for i and j and computing the count this way. However, in this assignment, you will develop a divide-and-conquer algorithm that is asymptotically faster and runs in $\Theta(nlogn)$ time. (Hint: modify the "merge" portion of merge-sort).

Method 1 Implement the $\Theta(n^2)$ brute-force approach to solving this problem, where we check all indices i, j where $i < j$ and count the number of elements out of order. This should be a simple enumeration (iterative) algorithm.

Method 2 Implement the $\Theta(nlogn)$ divide-and-conquer approach. You should start by implementing Merge Sort (check pseudocode on slides), then modify the non-recursive portion (merge) to keep track of the number of inversions. Your primary function should be recursive, but you may implement a helper function for $merge()$ as described in the lecture slides that forms larger sorted arrays when combining the solutions to the recursive calls. Again, this is where you will count the number of inversions (out of order elements).

Tasks

1. Write pseudocode for each of the two methods above.
2. Write the recurrence relation for the divide-and-conquer approach.
3. Implement both methods and provide the code in the language of your choice. Here are three test cases: $A = [1, 9, 6, 4, 5]$ should return 5. $A = [2, 4, 1, 3, 5]$ should return 3. $A = [1, 2, 3, 4, 5, 6]$ should return 0.

4. Randomly generate arrays of sizes $100, 200, \ldots, 1000, 2000, \ldots, 9000$ and plot the experimental runtimes for both algorithms on a single plot just like with GA1. Please run each array size 10 times with each method and average the results for each method - again, just like in GA1. Please put input size on the x-axis and runtime on the y-axis.

5. **Your pdf report should include pseudocode for each of the two methods, the recurrence relation, and the plot described above.**