CS 61B Fall 2020

More Sorting

Exam Prep Discussion 13: November 16, 2020

1 Sorted Runtimes

We want to sort an array of N **unique** numbers in ascending order. Determine the best case and worst case runtimes of the following sorts:

(a) Once the runs in merge sort are of $size <= N/100$, we perform insertion sort on them.

Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

(b) We can only swap adjacent elements in selection sort.

Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

(c) We use a linear time median finding algorithm to select the pivot in quicksort.

Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

(d) We implement heapsort with a min-heap instead of a max-heap. You may modify heapsort but must maintain constant space complexity.

Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

(e) We run an optimal sorting algorithm of our choosing knowing:

- There are at most N inversions

  Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

- There is exactly 1 inversion

  Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

- There are exactly $(N^2 - N)/2$ inversions

  Best Case: $\Theta($ $)$, Worst Case: $\Theta($ $)$

2 Shuffled Exams

For this problem, we will be working with `Exam` and `Student` objects, both of which have only one attribute: `sid`, which is a number like any student ID.

Gradescope thought it was ready for the midterm. It had meticulously created two arrays, one of `Exams` and the other of `Students`, and ordered both on `sid` such that the ith `Exam` in the `Exams` array has the same `sid` as the ith `Student` in the `Students` array. Note the arrays are not necessarily sorted by `sid`. However, Gradescope crashed, and the `Students` array was shuffled, but the `Exams` array somehow remained untouched. Time is precious, so you must design a O(N) time algorithm to reorder the Students array appropriately **without** changing the `Exams` array! For partial credit, you may reorder **both** the `Students` and `Exams` arrays such that ith `Exam` in the `Exams` array has the same `sid` as the ith `Student` in the `Students` array.
Hint: Use radix sort.

.

3 Bears and Beds

The hot new Cal startup AirBearsnBeds has hired you to create an algorithm to help them place their customers in the best possible homes to improve their experience. They are currently in their alpha stage so their only customers (for now) are bears. Now, a little known fact about bears is that they are very, very picky about their bed sizes: they do not like their beds too big or too little - they like them just right. Bears are also sensitive creatures who don't like being compared to other bears, but they are perfectly fine with trying out beds.

The Problem:
Given a list of Bears with unique but unknown sizes and a list of Beds with corresponding but also unknown sizes (not necessarily in the same order), return a list of Bears and a list of Beds such that that the ith Bear in your returned list of Bears is the same size as the ith Bed in your returned list of Beds. Bears can only be compared to Beds and we can get feedback on if the Bed is too large, too small, or just right. In addition, Beds can only be compared to Bears and we can get feedback if the Bear is too large for it, too small for it, or just right for it.

The Constraints:
Your algorithm should run in $O(N \log N)$ time on average. It may be helpful to figure out the naive $O(N^2)$ solution first and then work from there.