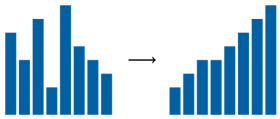
3 Improving Quick Sort

Problem Introduction

The goal in this problem is to redesign a given implementation of the randomized quick sort algorithm so that it works fast even on sequences containing many equal elements.



Problem Description

Task. To force the given implementation of the quick sort algorithm to efficiently process sequences with few unique elements, your goal is replace a 2-way partition with a 3-way partition. That is, your new partition procedure should partition the array into three parts: $< x$ part, $= x$ part, and $> x$ part.

Input Format. The first line of the input contains an integer n. The next line contains a sequence of n integers $a_0, a_1, \ldots, a_{n-1}$.

Constraints. $1 \le n \le 10^5$; $1 \le a_i \le 10^9$ for all $0 \le i < n$.

Output Format. Output this sequence sorted in non-decreasing order.

Sample 1.

Input:

```
5
2 3 9 2 2
```

Output:

```
2 2 2 3 9
```

Starter Files

In the starter files, you are given an implementation of the randomized quick sort algorithm using a 2-way partition procedure. This procedure partitions the given array into two parts with respect to a pivot x: $\le x$ part and $> x$ part. As discussed in the video lectures, such an implementation has $\Theta(n^2)$ running time on sequences containing a single unique element. Indeed, the partition procedure in this case splits the array into two parts, one of which is empty and the other one contains $n - 1$ elements. It spends cn time on this. The overall running time is then

$$cn + c(n-1) + c(n-2) + \ldots = \Theta(n^2)\,.$$

What To Do

Implement a 3-way partition procedure and then replace a call to the 2-way partition procedure by a call to the 3-way partition procedure.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.