One question of your choice (either #1 or #2) is due by **5:00PM Tuesday, April 2** and will be graded on completion only (1 point). Your full answers to all questions are due by **5:00PM Friday, April 5**. Each solution will be graded for correctness and clarity (4 points per question). Read the course information page for further info about this 4-point scale.

At the top of your write-up for each problem, please estimate the amount of time you spent on the problem, list your collaborators, and briefly describe the nature of your collaboration. Also, provide a list of other sources you used, including links for online resources. Thanks!

1. Guess what? You have been hired at the US Internal Revenue Service (IRS) as a data analyst. You have a dataset consisting of a single column of numbers, unsorted. This column of numbers is the gross adjusted income, rounded to the nearest 1000 dollars, from every US individual tax return filed for the 2023 tax year. So for example, there might be a 42 in the column representing an entry-level marketing associate who made $41,850 last year, a 5450 representing the $5.45 million made by Minnesota Twins pitcher Pablo López, and a 63100 for Apple's CEO Tim Cook.

   Your bosses want you to do a bunch of modeling experiments to imagine the impact of various proposed tax policies. Based on the many, many feature requests handed to you from up above, you have decided that there's one computational operation you need to make really fast: given a salary range (e.g., 50-75, representing $50,000 - $75,000), how many people (or, more accurately, how many tax returns) are there in that range? We're going to call this count the *range-count* for the specified range.

   Of course, the brute force approach to this problem is to scan through the entire column of numbers and count how many of them are in the specified range. The problem is that there are about 260 million tax returns, and you need to perform this operation a *lot*, with a ton of different randomized salary ranges, to complete your project. So an $O(n)$ algorithm just won't cut it.

   More technically, here's the situation. You have a list L containing n integer incomes in the range $[0, m]$. (Here, *income* means *the gross adjusted income from one of the tax returns, expressed as an integer number of thousands of US dollars.*) Your goal is to devise an algorithm to compute the range-count for any income range in $O(1)$ time.

   You may ***preprocess*** L as long as you can do it in $O(n + m)$ time. Preprocessing is a one-time action done before you run your range-count algorithm. You will then run range-count a zillion times (where *zillion* is a technical term whose definition is *I dunno for sure, but a lot*).

   Here's the range-count problem:

   Input: two integers $x \in [0, m]$ and $y \in [0, m]$ where $x \leq y$.
   Output: the number of incomes in the range $[x, y]$.

   For example, suppose $L = \langle 30, 25, 90, 80, 23, 35, 75, 70 \rangle$. Given $x = 25$ and $y = 75$, range-count would be 5.

   In your solution, make sure you clearly describe both your pre-processing step and how you determine the number of incomes in the specified range. You must also provide a proof (i.e., a well-reasoned and clearly articulated argument) that your approach is correct and meets all the specifications.

2. Back in the day, the Meanderer 2 spacecraft was launched, fully equipped with the best computer hardware you could buy in 1977. The Meanderer has three still-working cameras: Cameras B, C, and D.

   It has always been true that the Meanderer's cameras are capable of capturing a lot more image data than the available network bandwidth can transmit. The cameras have dutifully dumped their images into Meanderer's RAM for 47 years, but the transmitter has time to send only a tiny number of those images back to the Earth-bound scientists before the image memory fills up and the cameras go back to the beginning and start overwriting their old images.

   Recently the Meanderer's team (now including some grandchildren of the original team) had a bright idea. What if we upload some modern image processing algorithms to help Meanderer make better choices about which images to transmit? We might get more useful data that way.

   A long technical story involving 70s-era caches, I/O buses, and operating systems leads us here: we need the image pointers stored in our array A to be arranged so that all the Camera B images come first, then all the Camera C images, and finally all the Camera D images. We don't care about the order within each single-camera block. But the algorithm that rearranges the randomly distributed B's, C's, and D's has to be very fast.

   The Meanderer team has formalized this problem like so. You are given an array of n image pointers $A[1...n]$. Each image pointer $A[k]$ has a type $t(A[k]) \in \{\text{B, C, D}\}$. You are asked to rearrange the array A so that all of the B's come first, then all the C's, and then all the D's. In your solution, you are only allowed two very specific operations on the array:

   (i) for a particular j, query $t(A[j])$; and

   (ii) for two particular indices j and k, swap $A[j]$ and $A[k]$.

   For example, you cannot copy elements from the array to an auxiliary array or even create another array (there's just not enough RAM out there in the Oort Cloud). If you wish, you may use a small number of integer variables to store indices into the array.

   Give an algorithm that runs in time $O(n)$ to perform your assigned task. As always, you must also provide a proof that your approach is correct and obtains the specified runtime.

Submission Logistics

Write up your solutions to each question in this problem set using LaTeX. Convert the LaTeXinto a PDF file and upload the PDF to Moodle.

Please use the class LaTeX template as the basis for your write-up. See the class LaTeX resources page for more LaTeX help.

Have fun!