December 2017 USACO Bronze/Silver Review

Mihir Patel

January 12, 2018

1 Bronze - Blocked Billboard

1.1 Problem

During long milking sessions, Bessie the cow likes to stare out the window of her barn at two huge rectangular billboards across the street advertising "Farmer Alex's Amazingly Appetizing Alfalfa" and "Farmer Greg's Great Grain". Pictures of these two cow feed products on the billboards look much tastier to Bessie than the grass from her farm.

One day, as Bessie is staring out the window, she is alarmed to see a huge rectangular truck parking across the street. The side of the truck has an advertisement for "Farmer Smith's Superb Steaks", which Bessie doesn't quite understand, but she is mostly concerned about the truck potentially blocking the view of her two favorite billboards.

Given the locations of the two billboards and the location of the truck, please calculate the total combined area of both billboards that is still visible. It is possible that the truck obscures neither, both, or only one of the billboards.

INPUT FORMAT (file billboard.in): The first line of input contains four space-separated integers: x_1 y_1 x_2 y_2, where $(x_1, y_1)(x_1, y_1)$ and $(x_2, y_2)(x_2, y_2)$ are the coordinates of the lower-left and upper-right corners of the first billboard in Bessie's 2D field of view. The next line contains four more integers, similarly specifying the lower-left and upper-right corners of the second billboard. The third and final line of input contains four integers specifying the lower-left and upper-right corners of the truck. All coordinates are in the range -1000 to +1000. The two billboards are guaranteed not to have any positive area of overlap between themselves.

1.2 Solution

We know that when any two rectangles overlap, the region formed by the intersection will also be a rectangle. As we are given the two billboards do not overlap, we can treat each individually. For each billboard, we aim to see where the truck intersects the board in each dimension. We begin our scan at the bottom left of the board and iterate to the right, marking each point valid if that x-coordinate is in between the x-coordinates of the corners of the truck. We take the largest and smallest, and subtract them to get the x-component of the intersecting rectangle. We do this again for the y-axis.

We can then say the area of the board that is unblocked is the area of the rectangle minus the area of the intersecting region. The same procedure is repeated for billboard two to get the total unblocked region.

1.3 Example

1 2 3 5
6 0 10 4
2 1 8 3

Iterating over the bounds for the first rectangle, we get a 1x1 intersection. For the second rectangle, we get 2x2 intersection. Our answer is $6 - 1 + 16 - 4 = 17$.

2 Bronze - Bovine Shuffle

2.1 Problem

Convinced that happy cows generate more milk, Farmer John has installed a giant disco ball in his barn and plans to teach his cows to dance! Looking up popular cow dances, Farmer John decides to teach his cows the "Bovine Shuffle". The Bovine Shuffle consists of his N cows ($1N100$) lining up in a row in some order, then performing three "shuffles" in a row, after which they will be lined up in some possibly different order. To make it easier for his cows to locate themselves, Farmer John marks the locations for his line of cows with positions $1 \ldots N$, so the first cow in the lineup will be in position 1, the next in position 2, and so on, up to position N.

A shuffle is described with N numbers, $a_1 \ldots a_N$, where the cow in position i moves to position a_i during the shuffle (and so, each a_i is in the range $1 \ldots N$). Every cow moves to its new location during the shuffle. Fortunately, all the a_i's are distinct, so no two cows try to move to the same position during a shuffle.

Farmer John's cows are each assigned distinct 7-digit integer ID numbers. If you are given the ordering of the cows after three shuffles, please determine their initial order.

INPUT FORMAT (file shuffle.in): The first line of input contains N, the number of cows. The next line contains the N integers $a_1 \ldots a_N$. The final line contains the order of the N cows after three shuffles, with each cow specified by its ID number.

2.2 Solution

This one can be done by brute force, where you calculate the shuffles and just test each possible combination. A faster way to doing this is to simply reverse the transformation. We can create 3 arrays, one containing the given sequence, one with the shuffle orders, and a new sequence to place numbers in. We simply loop over the sequence with shuffle orders and say $placesequence[shuffle[i]] = sequence[i]$. This can just be done three times.

2.3 Example

5
1 3 4 5 2
1234567 2222222 3333333 4444444 5555555

Looping through this procedure we get 1234567 5555555 2222222 3333333 4444444.

3 Bronze - Milk Measurement

3.1 Problem

Farmer John purchases three cows: Bessie, Elsie, and Mildred, each of whom initially produces 7 gallons of milk per day. Since the milk output of a cow is known to potentially change over time, Farmer John takes periodic measurements over the next 100 days and scribbles them down in a log book. Entries in his log look like this:

35 Bessie -2

14 Mildred +3

The first entry indicates that on day 35, Bessie's milk output was 2 gallons lower than it was when last measured. The next entry indicates that on day 14, Mildred's milk output increased by 3 gallons from when it was last measured. Farmer John has only enough time to make at most one measurement on any given day. Unfortunately, he is a bit disorganized, and doesn't necessarily write down his measurements in chronological order.

To keep his cows motivated, Farmer John proudly displays on the wall of his barn the picture of whichever cow currently has the highest milk output (if several cows tie for the highest milk output, he displays all of their pictures). Please determine the number of days on which Farmer John would have needed to change this display.

INPUT FORMAT (file measurement.in): The first line of input contains N, the number of measurements Farmer John makes. Each of the next N lines contains one measurement, in the format above, specifying a day (an integer in the range 1..100), the name of a cow, and the change in her milk output since it was last measured (a nonzero integer). Each cow's milk output will always be in the range 0..1000.

3.2 Solution

Because we have such few days, we can just loop through each day and check if any notes occurred on this day (two nested for loops). A more optimal solution would be to sort the list. At each point in this loop, we constantly update the maximum values for each cow. We can keep three booleans indicating which ones are currently on the leaderboard. At each iteration in the leaderboard, we recalculate the booleans and increase some counter if this differs from the previously stored booleans.

3.3 Example

4
7 Mildred +3
4 Elsie -1
9 Mildred -1
1 Bessie +2

We obtain 3 through our method.

4 Silver - Cow Ate Homework

4.1 Problem

In your bovine history class, you have been given a rather long homework assignment with N questions ($3 <= N <= 100,000$), each graded with an integer score in the range 0...10,000. As is often customary, your teacher plans to assign a final grade by discarding a question on which you received the lowest score and then averaging the remaining scores together. Unfortunately, your pet cow Bessie has just eaten your answers to the first K questions! (K could be as small as 1 or as large as $N2$).

After copious explanation, your teacher finally believes your story, and agrees to grade the remaining non-eaten part of the assignment the same way as before – by removing the lowest-scoring question (or one such question, in the event of a tie) and averaging the rest.

Please output all values of K which would have earned you the maximum possible score according to this grading scheme, in sorted order.

INPUT FORMAT (file homework.in): The first line of input contains N, and the next line contains the scores on the N homework questions.

4.2 Solution

We seek to loop over all values of K and at each step remove minimum / get average, keeping track of highest average and the corresponding K. The trick is that we must go backwards, as if we remove the minimum while increasing K, it becomes $O(n)$ to find new minimum and recompute sum. By going backwards, we merely check if the new number added is a new minimum.

4.3 Example

5
3 1 9 2 7

We implement our solution to get 2.

5 Silver - Milk Measurement

5.1 Problem

Each of Farmer John's cows initially produces G gallons of milk per day ($1 <= G <= 109$). Since the milk output of a cow is known to potentially change over time, Farmer John decides to take periodic measurements of milk output and write these down in a log book. Entries in his log look like this:

35 1234 -2
14 2345 +3

The first entry indicates that on day 35, cow 1234's milk output was 2 gallons lower than it was when last measured. The next entry indicates that on day 14, cow 2345's milk output increased by 3 gallons from when it was last measured. Farmer John has only enough time to make at most one measurement on any given day. Unfortunately, he is a bit disorganized, and doesn't necessarily write down his measurements in chronological order.

To keep his cows motivated, Farmer John proudly displays on the wall of his barn the picture of whichever cow currently has the highest milk output (if several cows tie for the highest milk

output, he displays all of their pictures). Please determine the number of days on which Farmer John would have needed to change this display.

Note that Farmer John has a very large herd of cows, so although some of them are noted in his log book as changing their milk production, there are always plenty of other cows around whose milk output level remains at G gallons.

INPUT FORMAT (file measurement.in): The first line of input contains the number of measurements N that Farmer John makes ($1 <= N <= 100,000$), followed by G. Each of the next N lines contains one measurement, in the format above, specifying a day (an integer in the range $1 \ldots 106$), the integer ID of a cow (in the range $1 \ldots 109$), and the change in her milk output since it was last measured (a nonzero integer). Each cow's milk output will always be in the range $0 \ldots 109$.

5.2 Solution

We first sort all values chronologically. We then update the value of each cow that was changed by looping through time. We also store current list of cows at top. We say if a cow's value decreased and it was removed, a counter keep track of changes is increased. We say if a value increased above current max, we create a new list of cows at the top starting with this one and checking all cow's whose values changed that day, and we increment the counter. To check if a cow is currently at the top, we can use hashing for $O(1)$ time.

5.3 Example

4 10
7 3 +3
4 2 -1
9 3 -1
1 1 +2

We implement our solution to get 3.

6 Silver - Bovine Shuffle

6.1 Problem

Convinced that happy cows generate more milk, Farmer John has installed a giant disco ball in his barn and plans to teach his cows to dance!

Looking up popular cow dances, Farmer John decides to teach his cows the "Bovine Shuffle". The Bovine Shuffle consists of his N cows ($1 <= N <= 100,000$) lining up in a row in some order, then performing successive "shuffles", each of which potentially re-orders the cows. To make it easier for his cows to locate themselves, Farmer John marks the locations for his line of cows with positions $1 \ldots N$, so the first cow in the lineup will be in position 1, the next in position 2, and so on, up to position N.

A shuffle is described with N numbers, $a_1 \ldots a_N$, where a cow in position i moves to position a_i during the shuffle (and so, each a_i is in the range $1 \ldots N$). Every cow moves to its new location during the shuffle. Unfortunately, all the a_i's are not necessarily distinct, so multiple cows might try to move to the same position during a shuffle, after which they will move together for all remaining shuffles.

Farmer John notices that some positions in his lineup contain cows in them no matter how many shuffles take place. Please help him count the number of such positions.

INPUT FORMAT (file shuffle.in): The first line of input contains N, the number of cows. The next line contains the N integers $a_1 \ldots a_N$.

6.2 Solution

We create a list of length N. We then loop through the shuffle array, incrementing the value for a location when it is pointed to. After, we create a queue and populate it with all locations that have no pointers to them. For obvious reasons, a cow will never be able to re-enter this square so we consider it dead. We go through the queue. Whenever we pop an element, we look at where it's pointing to and decrement the counter for that location in the original list because a dead square cannot feed another square. If the location we just decremented the value for becomes a dead square, we add it to the queue. Our solution is the total number of locations minus the number of locations that were processed through the queue.

6.3 Example

4
3 2 1 3

We implement our solution to get 3.