Kick Start 2019 - Round C

Analysis: Circuit Board

Test set 1 (Visible)

Restating the problem, we are asked to find the largest sub-rectangle in a grid of integers such that the difference between the maximum and minimum integers in each row of the sub-rectangle is not more than **K**.

For test set 1, we have **K** = 0. This means we have to find the largest sub-rectangle in which all integers are equal in each of the rows.

To do this, we first define $P_{i,j}$ as the maximum number of contigious cells on the right of the cell i, j which have the same value as cell i, j.

We immediately observe that $P_{i, j} = 1 + P_{i, j + 1}$ if the integers in j and (j+1)th columns of row i are equal, else $P_{i,j}$ is 1.

Once we have these values, our problem simply reduces to the well known problem of finding the area of the largest rectangle in a histogram. For every column c, we first divide it into segments where all integers are equal. For any such segment say from row r_1 to r_2, we take $P_{i, c}$ for i = r_1 to r_2 as the heights of histogram. It is clear now that any rectangle in this histogram represents a rectangle where in all integers are equal in each row of the rectangle.

Calculating the largest rectangle in the histogram of width M can be done in O(M) time using stacks. The idea is to find for every bar of the histogram, the closest bars to the left and right of the current bar which are shorter than the current bar. To find the closest shorter bar on the left (we can modify the same procedure to also find the closest shorter bar on the right), we traverse our histogram from left to right. We initially have an empty stack which we update as follows:
If the current bar's height is greater than the top element of the stack, then it's the required bar for our current bar and we push the current bar's height into the stack.
Otherwise we keep on popping the stack till we find a number which is lesser than the current bar's height (or the stack becomes empty, in which case there's no bar to the left which is shorter than the current bar). This top element now corresponds to our required bar. We then push the current bar's height into our stack. Since we push or pop at most once for every bar, the complexity of this procedure is linear as described above.

And hence, we'll have to spend O(**R**) time per column with a total run time complexity of O(**RC**).

Test set 2 (Hidden)

For this case, **K** >= 0, so we re-define $P_{i,j}$ as the maximum length such that the difference between the maximum number and the minimum number present in all cells of row i with column >= j and < j + $P_{i,j}$ is not more than **K**.

To calculate $P_{i,j}$ we binary search for the first cell in the ith row and to the right of jth column such that the condition that the difference between the maximum and minimum number present in these cells is not more than **K** is violated. We can do this quickly if we can answer range minimum and range maximum queries quickly. Those can be dealt with a lot of different data-structures.

Once we have this, our problem again reduces to the problem of finding the area of the largest rectangle in a histogram. For, every column c, we take $P_{i, c}$ for i = 1 to **R** as the heights of histogram. It is clear now that any rectangle in this histogram represents a rectangle where in no row maximum and minimum elements differ by more than **K**.

If we use a data-structure like Sparse Table to answer the range minimum and maximum queries then we can find all the P-values in O(**RC**log(**C**)) time. The last part of finding the largest area rectangle in histogram should be done once for each column, and hence takes O(**R**) time for each case resulting in O(**RC**) time in total. Combining both, we have a final complexity of O(**RC**log(**C**)). We can further reduce this to a complexity of O(**RC**) by replacing the Sparse Table data-structure with a two pointers variation using two deques, that approach is left as an exercise for the reader.