Kick Start 2021 - Round A

Analysis: Rabbit House

Barbara is given a 2D grid with $\mathbf{R}$ rows and $\mathbf{C}$ columns. Her goal is to create a grid such that no two adjacent cells have an absolute difference in height greater than 1. Moreover, she is only allowed to increase the height of cells.

We notice that the cell initially with the largest height (H) never requires increasing. Furthermore, Barbara can update its neighbor cells to have a height of $H - 1$ (unless they already have a height of H). Afterwards, she can repeat the process for the cell with the next largest height, until she visits all the cells.

One thing to be careful of in this problem is that the final result can be larger than the limits of a 32-bit integer. Using 64-bit integers avoids WAs due to overflow.

Test set 1

For this test set, Barbara sees that $1 \leq \mathbf{R}, \mathbf{C} \leq 50$. Therefore, she performs a linear scan over the grid to find the cell with the largest height. Then, she updates the height of its neighbors, and increments the result to account for the increase in height.

Finally, she marks this cell as visited, which can be done via a secondary 2D grid of booleans. She repeats the above process until all cells have been visited, and return the result.

The linear scan can be done in $O(\mathbf{R} \cdot \mathbf{C})$, and Barbara visits each cell exactly once, so she performs the linear scan $O(\mathbf{R} \cdot \mathbf{C})$ times. The overall time complexity is therefore $O((\mathbf{R} \cdot \mathbf{C})^2)$. This fits within the limits of the small test set.
The additional space complexity is $O(\mathbf{R} \cdot \mathbf{C})$ due to the secondary 2D grid of booleans.

Test set 2

For this test set, $1 \leq \mathbf{R}, \mathbf{C} \leq 300$. A time complexity of $O((\mathbf{R} \cdot \mathbf{C})^2)$ will not satisfy the time limits.

Barbara still needs to visit the unvisited cell with the largest height in each iteration. However, this can be done in $O(\log(\mathbf{R} \cdot \mathbf{C}))$ using a priority queue. In each iteration, she pops the cell with the largest height, updates the height of its neighbors in the priority queue, and increments the result.

The time complexity of the above solution is $O(\mathbf{R} \cdot \mathbf{C} \cdot \log(\mathbf{R} \cdot \mathbf{C}))$. The additional space complexity is still $O(\mathbf{R} \cdot \mathbf{C})$, since initially the priority queue contains all the cells.

Note that, depending on the implementation, updating a non-top element in the priority queue might not be an $O(1)$ operation. In such cases, one trick would be to insert a new element into the queue with the new height, and update the height in the grid. On processing any element, check whether the height in the queue corresponds to the height in the grid, and ignore the element otherwise. This does not change the worst-case time and space complexity, since the maximum number of times a cell can be added to the queue is 4.

Further Improvements

While not necessary to pass the time limits, the time complexity can be reduced further.

One approach is to replace the priority queue with a list of buckets, each bucket containing a set of cells with a given height. With this approach, Barbara can iterate over the buckets in decreasing order of height in order to visit each cell, then apply the same algorithm as above: update the neighbors (by placing them in their new buckets according to their new height), and remove the visited cells from the list.

By using a hashset for each bucket, insertion and removal becomes an $O(1)$ operation. She can also use a vector. However, the trick from Test Set 2 is needed to maintain an $O(1)$ insertion/removal in that case. Iterating over the buckets is linear in the number of buckets, and she will visit at most $O(\mathbf{R} \cdot \mathbf{C})$ cells.

Let $G = \max(\mathbf{G_{i,j}})$. Since no cell has an initial height larger than G, and she will never increase the height of the cell with the largest height, she observes that the number of buckets is at most G. This leads to a time complexity of $O(\mathbf{R} \cdot \mathbf{C} + G)$.

Barbara can improve this further. Notice that all cells in the final grid will have a height of at least $G - \mathbf{R} - \mathbf{C} + 2$. She achieves this value by decreasing the height by 1 with each step away from the highest cell. The maximum number of steps occurs when the highest cell is in a corner of the grid: the opposite corner is $\mathbf{R} + \mathbf{C} - 2$ steps away.

Now, she can first increase the height of all cells to $G - \mathbf{R} - \mathbf{C} + 2$, then apply the same bucketing approach. Except, now the number of buckets is $\mathbf{R} + \mathbf{C} - 2$, leading to a time complexity of $O(\mathbf{R} \cdot \mathbf{C} + \mathbf{R} + \mathbf{C}) = O(\mathbf{R} \cdot \mathbf{C})$.