Maximum Strictly Increasing Cells in a Matrix (/problems/maximum-strictly-increasing-cells-in-a-matrix/)

Submission Detail

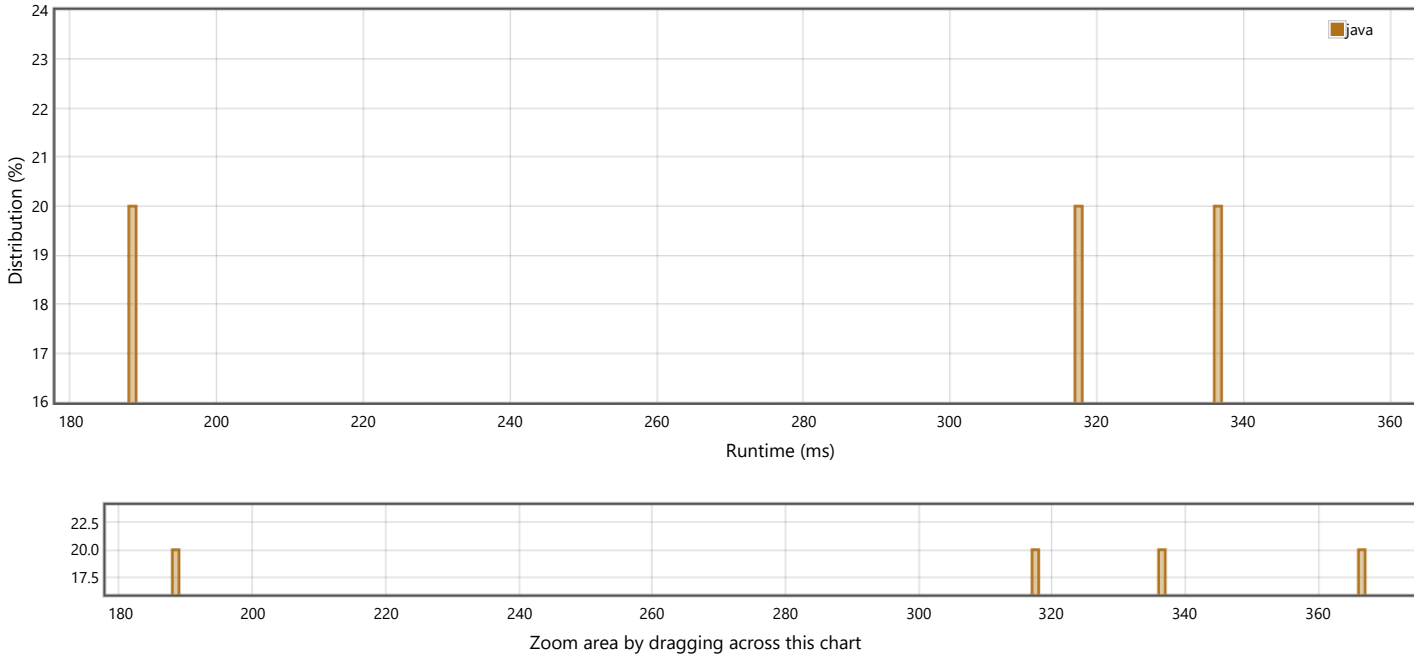**564 / 564** test cases passed.   Status: **Accepted**

Runtime: **149 ms**
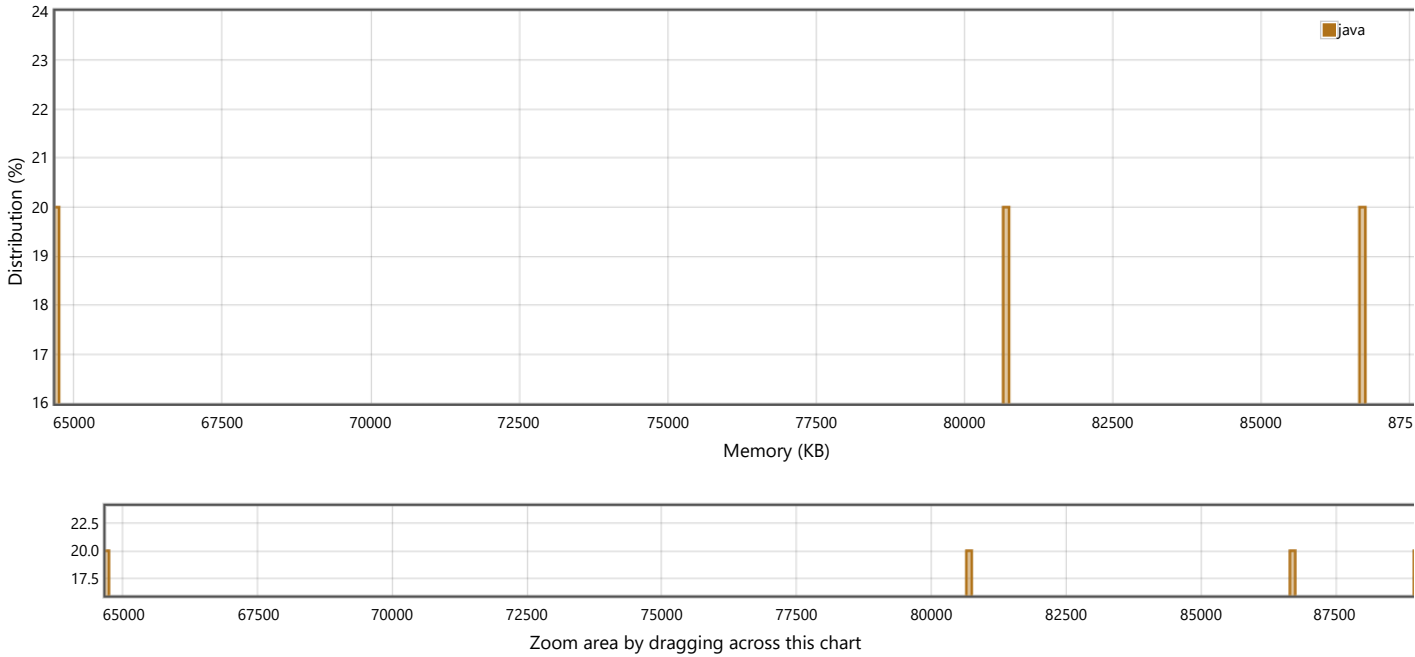Memory Usage: **81.8 MB**   Submitted: **27 minutes ago**

Accepted Solutions Runtime Distribution

Accepted Solutions Memory Distribution

Invite friends to challenge **Maximum Strictly Increasing Cells in a Matrix**

Submitted Code: 27 minutes ago

Language: java   [Edit Code]

```
1   class Solution {
```

```java
    public int maxIncreasingCells(int[][] mat) {
        int r = mat.length;
        int c = mat[0].length;
        int[] row = new int[r];
        int[] col = new int[c];
        int ans = 0;
        List<int[]> arr = new ArrayList<>();

        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                arr.add(new int[]{mat[i][j], i, j});
            }
        }

        Collections.sort(arr, (a, b) -> Integer.compare(a[0], b[0]));
        Queue<int[]> qu = new LinkedList<>();

        for (int[] itr : arr) {
            while (!qu.isEmpty()) {
                int[] cur = qu.peek();
                if (cur[0] < itr[0]) {
                    row[cur[1]] = Math.max(row[cur[1]], cur[3]);
                    col[cur[2]] = Math.max(col[cur[2]], cur[3]);
                    qu.poll();
                } else {
                    break;
                }
            }

            int cur = Math.max(row[itr[1]], col[itr[2]]) + 1;
            ans = Math.max(ans, cur);
            qu.offer(new int[]{itr[0], itr[1], itr[2], cur});
        }

        return ans;
    }
}
```