Leetcode solution of histogram

Saturday, October 30, 2021 3:58 PM

For any bar `i` the maximum rectangle is of width `r - l - 1` where r - is the last coordinate of the bar to the **right** with height `h[r] >= h[i]` and l - is the last coordinate of the bar to the **left** which height `h[l] >= h[i]`
So if for any `i` coordinate we know his utmost higher (or of the same height) neighbors to the right and to the left, we can easily find the largest rectangle:

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

```
intmaxArea = 0;
for(inti = 0; i < height.length; i++) {
    maxArea = Math.max(maxArea, height[i] * (lessFromRight[i] - lessFromLeft[i] - 1));
}
```

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

The main trick is how to effectively calculate `lessFromRight` and `lessFromLeft` arrays. The trivial solution is to use **O(n^2)** solution and for each `i` element first find his left/right heighbour in the second inner loop just iterating back or forward:

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

```
for(inti = 1; i < height.length; i++) {
    intp = i - 1;
    while(p >= 0&& height[p] >= height[i]) {
        p--;
    }
    lessFromLeft[i] = p;
}
```

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

The only line change shifts this algorithm from **O(n^2)** to **O(n)** complexity: we don't need to rescan each item to the left - we can reuse results of previous calculations and "jump" through indices in quick manner:

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

```
while(p >= 0&& height[p] >= height[i]) {
    p = lessFromLeft[p];
}
```

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

Here is the whole solution:

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>

```
publicstaticintlargestRectangleArea(int[] height){
   if(height == null|| height.length == 0) {
      return0;
   }
   int[] lessFromLeft = newint[height.length]; // idx of the first bar the left that is lower than currentint[]
lessFromRight = newint[height.length]; // idx of the first bar the right that is lower than
currentlessFromRight[height.length - 1] = height.length;
   lessFromLeft[0] = -1;
for(inti = 1; i < height.length; i++) {
      intp = i - 1;
while(p >= 0&& height[p] >= height[i]) {
         p = lessFromLeft[p];
      }
      lessFromLeft[i] = p;
   }
for(inti = height.length - 2; i >= 0; i--) {
      intp = i + 1;
while(p < height.length && height[p] >= height[i]) {
         p = lessFromRight[p];
      }
      lessFromRight[i] = p;
   }
intmaxArea = 0;
   for(inti = 0; i < height.length; i++) {
      maxArea = Math.max(maxArea, height[i] * (lessFromRight[i] - lessFromLeft[i] - 1));
   }
returnmaxArea;
}
```

From <https://leetcode.com/problems/largest-rectangle-in-histogram/discuss/28902/5ms-O(n)-Java-solution-explained-(beats-96)>