1. 题目
2. 题干

请根据每日 气温 列表，重新生成一个列表。对应位置的输出为：要想观测到更高的气温，至少需要等待的天数。如果气温在这之后都不会升高，请在该位置用 0 来代替。

例如，给定一个列表 temperatures = [73, 74, 75, 71, 69, 72, 76, 73]，你的输出应该是 [1, 1, 4, 2, 1, 1, 0, 0]。

提示：气温 列表长度的范围是 [1, 30000]。每个气温的值的均为华氏度，都是在 [30, 100] 范围内的整数。

1. 题解
2. 思路

这个题目用暴力解法可以很容易实现，直接遍历每个数，查找大于它的第一个数即可。可是这样的时间复杂度非常高，我们需要找个更加的做法。这里使用单调栈的做法，从名字就可以看出来，栈中的数字的排序是单调的。本题中寻找的是每个数后面更大的那个数，因此每次入栈的一定是小于等于当前栈顶的数。遍历每个数，如果当前栈非空且当前的数大于栈顶的数，说明该数一定是第一个大于栈顶元素的数，直接求出二者的下标差。若当前的数小于或等于栈顶元素，则入栈，这样可以保持单调栈。

1. 代码

Java：

暴力解法：

class Solution {

    public int[] dailyTemperatures(int[] T) {

        if(T == null || T.length == 0){

            return new int[0];

        }

        //暴力解法

        int n = T.length;

        for(int i = 0; i < n; ++i){

            if(i == n-1){

                T[i] = 0;

                break;

            }

            for(int j = i+1; j < n; ++j){

                if(T[i] < T[j]){

                    T[i] = j-i;

                    break;

                }

                if(j==n-1 && T[i] >= T[j]){

                    T[i] = 0;

                }

            }

        }

        return T;

    }

}

单调栈：

class Solution {

    public int[] dailyTemperatures(int[] T) {

        if(T == null || T.length == 0){

            return new int[0];

        }

        // 单调栈

        int n = T.length;

        int[] out = new int[n];

        LinkedList<Integer> stack = new LinkedList<>();

        for(int i = 0; i < n; ++i){

            while(!stack.isEmpty() && T[i] > T[stack.getLast()]){

                int t = stack.removeLast();

                out[t] = i- t;

            }

            stack.addLast(i);

        }

        return out;

    }

}