E2-F problem

题目描述

小水獭在夏令营机试过程中遇到了一个难题，想要紧急求助你，你能帮帮它嘛?

具体来说，给定一个长度为 n 的数组 a1,a2,...,an，你需要求解其中所有子数组的极差的和。

形式化地说，定义 S(l,r)（1≤l≤r≤n）表示 al,al+1,...,ar 这些元素所组成的集合，maxS表示集合 S 中的最大值，minS 表示集合 S 中的最小值。请你求解：

$$\sum_{i=1}^{n}\sum_{j=i}^{n}[\max S(i,j)-\min S(i,j)]$$

题解思路

- 首先如果使用暴力求解，双重循环遍历数组，第二层循环每次记录最大最小值，时间复杂度是O(n^2)，超时无法通过。
- 转变一下思路，题目要求全部子数组的极差，可以分成极大值和极小值两部分：

$$\sum_{i=1}^{n}\sum_{j=1}^{n}[\max S(i,j)-\min S(i,j)]=\sum_{i=1}^{n}\sum_{j=1}^{n}\max S(i,j)-\sum_{i=1}^{n}\sum_{j=1}^{n}\min S(i,j)$$

- 而对于全部子数组的最大值和，可以转换为计算原数组中每个元素作为子数组最大值的次数×该元素值，再求和。
- 为了保证子数组最大元素唯一，定义逻辑大于：若ai > aj，或ai = aj且i > j，则ai逻辑大于aj。同理定义逻辑小于。
- 对于元素ai，以计算其作为子数组逻辑最大值的次数为例，需要找到在它之前出现的最近的逻辑大于它的元素al，以及在它之后出现的最近的逻辑大于它的元素ar。则元素ai作为子数组逻辑最大值的次数为 (i - l) × (r - i)。
- 若遍历数组每个元素，再向两侧遍历查找，时间复杂度依旧为O(n^2)，超时无法通过评测。
- 使用单调栈的方法存储每个元素之前或之后出现的第一个比自身逻辑大或逻辑小的元素，一共需要遍历四次数组，并用四个数组minLeft[n]、minRight[n]、maxLeft[n]、maxRight[n]来存储坐标。
- 维护单调栈的方法：【设数组为A[n]，以存储ai元素前出现的第一个逻辑大于自身的元素坐标为例子（即填写maxLeft[n]）】
    - 初始化单调栈为空，用于存储元素下标。从左到右遍历数组，遍历到元素ai时，若栈为空，则进行下一步操作，否则对单调栈栈顶进行判断，若A[top] <= A[i] (即逻辑小于)，进行出栈操作。重复此判断、出栈操作，直到栈空或A[top] > A[i]。【注：top为单调栈栈顶元素值】
    - 若此时栈为空，则直接令maxLeft[i] = -1，否则令maxLeft[i] = top
- 维护单调栈的方法：【设数组为A[n]，以存储ai元素后出现的第一个逻辑小于自身的元素坐标为例子（即填写minRight[n]）】
    - 初始化单调栈为空，用于存储元素下标。从右到左遍历数组，遍历到元素ai时，若栈为空，则进行下一步操作，否则对单调栈栈顶进行判断，若A[top] > A[i] (即逻辑大于)，进行出栈操

作。重复此判断、出栈操作，直到栈空或A[top] <= A[i]。【注：top为单调栈栈顶元素值】
    - 若此时栈为空，则直接令minRight[i] = n，否则令minRight[i] = top
- 最后遍历数组，对于ai分别计算其作为子数组最大、最小值为极差做出的贡献：
    - `sumMax += static_cast<__int128>(maxRight[i] - i) * (i - maxLeft[i]) * A[i];`
    - `sumMin += static_cast<__int128>(minRight[i] - i) * (i - minLeft[i]) * A[i];`
- 输出sumMax - sumMin即可。

代码

```
#include <iostream>
#include <iostream>
#include<bits/stdc++.h>

using namespace std;

#define MAX 1000005
int box[MAX] = {0};

void print(__int128 x){
    if(x < 10){
        printf("%d", (int)x);
        return;
    }
    print(x / 10);
    printf("%d", (int)(x % 10));
}

int main() {
    int t;
    scanf("%d", &t);
    while (t--) {
        int n;
        scanf("%d", &n);
        for (int i = 0; i < n; i++) {
            scanf("%d", &box[i]);
        }
        vector<int> minLeft(n), minRight(n), maxLeft(n), maxRight(n);
        stack<int> minStack, maxStack;
        __int128 ans = 0;
        for (int i = 0; i < n; i++) {
            while (!minStack.empty() && box[minStack.top()] > box[i]) {
                minStack.pop();
            }
            if (minStack.empty()) minLeft[i] = -1;
            else minLeft[i] = minStack.top();
            minStack.push(i);

            while (!maxStack.empty() && box[maxStack.top()] <= box[i]) {
                maxStack.pop();
            }
            if (maxStack.empty()) maxLeft[i] = -1;
            else maxLeft[i] = maxStack.top();
            maxStack.push(i);
        }
```

```
        minStack = stack<int>();
        maxStack = stack<int>();
        for (int i = n - 1; i >= 0; i--) {
            while (!minStack.empty() && box[minStack.top()] >= box[i]) {
                minStack.pop();
            }
            if (minStack.empty()) minRight[i] = n;
            else minRight[i] = minStack.top();
            minStack.push(i);

            while (!maxStack.empty() && box[maxStack.top()] < box[i]) {
                maxStack.pop();
            }
            if (maxStack.empty()) maxRight[i] = n;
            else maxRight[i] = maxStack.top();
            maxStack.push(i);
        }
        __int128 sumMax = 0, sumMin = 0;
        for (int i = 0; i < n; i++) {
            sumMax += static_cast<__int128>(maxRight[i] - i) * (i - maxLeft[i])
* box[i];
            sumMin += static_cast<__int128>(minRight[i] - i) * (i - minLeft[i])
* box[i];
        }
        ans = sumMax - sumMin;
        print(ans);
        printf("\n");
    }
    return 0;
}
```