

解题思路一：

边遍历边统计：

一次遍历，同时遍历两个数组，因为长度一样，所以遍历时一起比较，如果相同，取ans=max(ans,++f),如果不同f置为0，ans初值也是0

class Solution {

public:

int temperatureTrend(vector<int>& temperatureA, vector<int>& temperatureB) {

int ans = 0, f = 0;

for (int i = 0; i < temperatureA.size() - 1; ++i) {

//获取趋势

int x = temperatureA[i + 1] - temperatureA[i];

int y = temperatureB[i + 1] - temperatureB[i];

//比较趋势

if ((x == 0 && y == 0) || x \* y > 0) {

ans = max(ans, ++f);

}

else {

f = 0;

}

}

return ans;

}

};

解题思路二：

多次遍历不统计，最后比较：遍历数组得到天气变化趋势，遍历两个天气变化趋势，设置一个计数单位，当天气相同时，count++，不同时置为0，相同时取max(nmax,count),nmax初值设置为0

#include <iostream>

#include <vector>

using namespace std;

class Solution {

public:

int temperatureTrend(vector<int>& temperatureA, vector<int>& temperatureB) {

int len = temperatureA.size() - 1;

//获取天气数组

vector<int> ans1 = getans(temperatureA);

vector<int> ans2 = getans(temperatureB);

//获取最大天数

int nmax = 0, count = 0;

for (int k = 0; k < len; k++) {

if (ans1[k] == ans2[k]) {

count++;

nmax = max(nmax, count);

}

else {

count = 0;

}

}

return nmax;

}

private:

vector<int> getans(vector<int>& temperature) {

int len = temperature.size() - 1;

vector<int> ans(len, 0);

for (int i = 0; i < len; i++) {

if (temperature[i] < temperature[i + 1]) {

ans[i] = 1;

}

else if (temperature[i] > temperature[i + 1]) {

ans[i] = -1;

}

else {

ans[i] = 0;

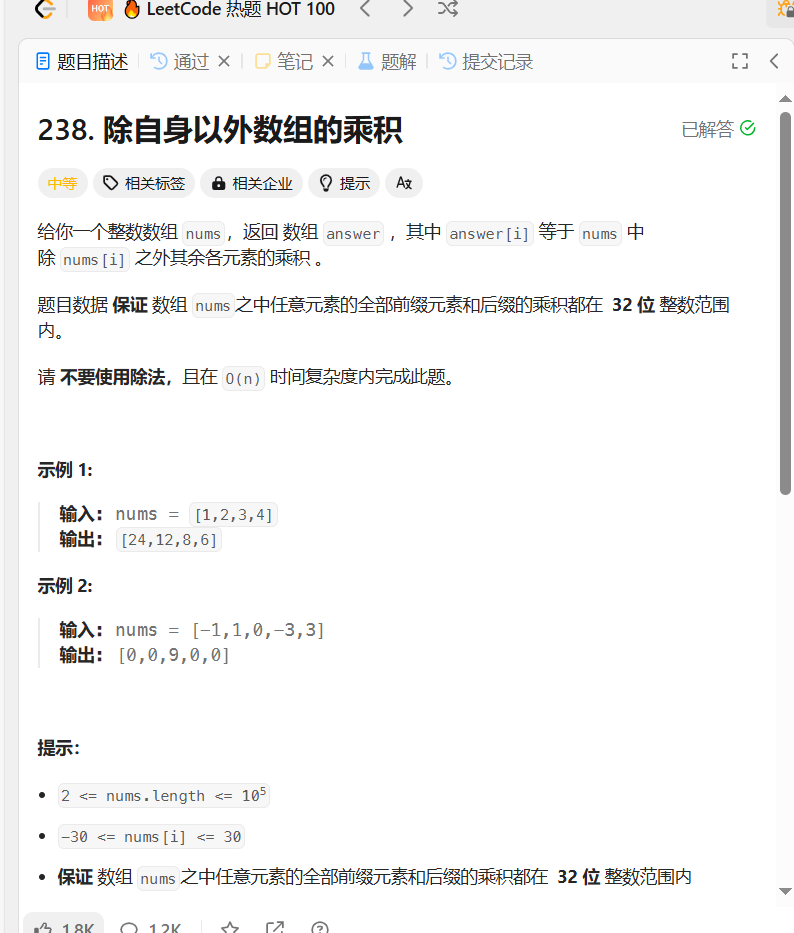
}

}

return ans;

}

};

题目：

解题思路一：

由于题目中要求不能用除法，且不能超过o(n)，所以我们不能暴力，那么我们可以考虑前缀和后缀，即把数字的左边视为前缀，右边视为后缀，存储左右两边的乘积，然后相乘就可以得到我们的结果

#include <vector>

#include <iostream>

using namespace std;

vector<int> productExceptSelf(const vector<int>& nums) {

int n = nums.size();

if (n == 0) {

return {};

}

vector<int> left\_products(n, 1);

vector<int> right\_products(n, 1);

vector<int> answer(n, 1);

// 计算每个元素左边的乘积

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

left\_products[i] = left\_products[i - 1] \* nums[i - 1];

}

// 计算每个元素右边的乘积

for (int i = n - 2; i >= 0; --i) {

right\_products[i] = right\_products[i + 1] \* nums[i + 1];

}

// 计算结果

for (int i = 0; i < n; ++i) {

answer[i] = left\_products[i] \* right\_products[i];

}

return answer;

}

解题思路二：

使用o(1)空间，我们设置一个输出数组ans，先让ans等于左边的乘积，定义一个变量right\_product，初值为1，来接我们每次右边的值，那么当我们左边乘积结束时，到右边时，ans[i]就变成了ans[i]\*=right\_product,每次更新后同时更新right\_product\*=nums[i]即可

class Solution {

public:

    vector<int> productExceptSelf(vector<int>& nums) {

        int n = nums.size();

    if (n == 0) {

        return {};

    }

    vector<int> answer(n, 1);

    // 计算每个元素左边的乘积

    for (int i = 1; i < n; ++i) {

        answer[i] = answer[i - 1] \* nums[i - 1];

    }

    // 计算每个元素右边的乘积并更新结果

    int right\_product = 1;

    for (int i = n - 1; i >= 0; --i) {

        answer[i] \*= right\_product;

        right\_product \*= nums[i];

    }

    return answer;

    }

};