5724. 绝对差值和--[代码处理的功底]

题目描述：

给你两个正整数数组 `nums1` 和 `nums2` ，数组的长度都是 `n` 。

数组 `nums1` 和 `nums2` 的 **绝对差值和** 定义为所有 `|nums1[i] - nums2[i]|` （`0 <= i < n`）的 **总和**（**下标从 0 开始**）。

你可以选用 `nums1` 中的 **任意一个** 元素来替换 `nums1` 中的 **至多** 一个元素，以 **最小化** 绝对差值和。

在替换数组 `nums1` 中最多一个元素 **之后** ，返回最小绝对差值和。因为答案可能很大，所以需要对 $10^9 + 7$ **取余** 后返回。

`|x|` 定义为：

- 如果 `x >= 0` ，值为 `x` ，或者
- 如果 `x <= 0` ，值为 `-x`

示例 1：

```
输入：nums1 = [1,7,5], nums2 = [2,3,5]
输出：3
解释：有两种可能的最优方案：
- 将第二个元素替换为第一个元素：[1,7,5] => [1,1,5] ，或者
- 将第二个元素替换为第三个元素：[1,7,5] => [1,5,5]
两种方案的绝对差值和都是 |1-2| + (|1-3| 或者 |5-3|) + |5-5| = 3
```

示例 2：

```
输入：nums1 = [2,4,6,8,10], nums2 = [2,4,6,8,10]
输出：0
解释：nums1 和 nums2 相等，所以不用替换元素。绝对差值和为 0
```

示例 3：

```
输入：nums1 = [1,10,4,4,2,7], nums2 = [9,3,5,1,7,4]
输出：20
解释：将第一个元素替换为第二个元素：[1,10,4,4,2,7] => [10,10,4,4,2,7]
绝对差值和为 |10-9| + |10-3| + |4-5| + |4-1| + |2-7| + |7-4| = 20
```

提示：

- `n == nums1.length`
- `n == nums2.length`
- `1 <= n <= 10^5`
- `1 <= nums1[i], nums2[i] <= 10^5`

题目分析：

考虑到大整数加法可能出现的越界问题！！！

这道题从本质上来说并没有设计算法上的东西：关键是去找交换值后可以得到最大减免值的情况；

暴力解法或许更好；优化的话可能造成很多其他方面的消耗

代码：（存在问题）-10^9 并不是表示10的9次方-正确写法int mod=1e9(double)

```
class Solution {
public:
    int minAbsoluteSumDiff(vector<int>& nums1, vector<int>& nums2) {
        /*
        //备份
        vector<int>temp1(nums1);
        vector<int>temp2(nums2);
        //备份排序
        sort(temp1.begin(),temp1.end());
        sort(temp2.begin(),temp2.end());
        */
        int mod=1e9+7;

        int n=nums1.size(),res=0;
        unordered_set<int>temp1;
        //unordered_set<int>temp2;
        for(int i=0;i<n;i++){
            temp1.insert(nums1[i]);
            //temp2.insert(nums2[i]);
        }
        //差值集合
        vector<vector<int>>difft(nums1.size(),vector<int>(2,0));
        for(int i=0;i<n;i++){
            difft[i][0]=abs(nums1[i]-nums2[i]);
            difft[i][1]=i;
            res+=difft[i][0];
            //res%=10^9 + 7;//此处有问题10^9并不是表示10的9次方
            res%=mod;
```

```
        }
        sort(difft.begin(),difft.end());//,greater<int>());//从大到小排序

        int diff_max=0;//最大可以减少的差值
        int abs_num=difft[n-1][0];//满足什么样的条件可以继续向下延申
        for(int i=n-1;i>=0;i--){
            //满足什么样的条件可以继续向下延申
            if(difft[i][0]>=abs_num){
                int temp=nums2[difft[i][1]];//nums2中该处的值
                int k=0;
                while(1){
                    if((temp1.find(temp-
k)!=temp1.end()||temp1.find(temp+k)!=temp1.end())&&k<=temp){
                        diff_max=max(difft[i][0]-k,diff_max);
                        abs_num=diff_max+1;
                        break;
                    }else if(k>temp){
                        break;
                    }
                    k++;
                }

            }else break;
        }


        res=(res-diff_max+mod)%mod;//10^9 并不是表示10的9次方
        return res;


    }
};
```

大佬代码：【暴力解法】【二分查找】

```
class Solution {
public:
    int minAbsoluteSumDiff(vector<int>& nums1, vector<int>& nums2) {
        int mod = 1e9 + 7;
        int ans = 0, n = nums1.size();
        for(int i=0;i<n;i++) ans = (ans + abs(nums1[i]-nums2[i])) % mod;
        auto tmp = nums1;
        sort(nums1.begin(),nums1.end());
        int mx = 0;
        for(int i=0;i<n;i++){
            auto p = lower_bound(nums1.begin(),nums1.end(),nums2[i]);
```

```
            if(p!=nums1.end()){
                int x = *p;
                mx = max(mx,abs(tmp[i]-nums2[i])-abs(x-nums2[i]));
            }
            if(p!=nums1.begin()){
                p--;
                int x = *p;
                mx = max(mx,abs(tmp[i]-nums2[i])-abs(x-nums2[i]));
            }
        }
        ans = (ans - mx + mod) % mod;
        return ans;
    }
};
```