Good Bye 2021: 2022 is NEAR

A. Integer Diversity

题意:

给你一个数字序列，你可以对序列里面的任意元素进行正负号变换，求最多存在多少个不同的序列。

思路:

将所有的正负数变成正数存到flag数组里面，然后每存一次flag[i]++，对于除去0外的数flag[i]==1使sum++,flag[i]>=2使sum+=2,对于flag[0]>0就sum++,最后输出sum。

刚开始的时候我觉得这种简单题可以争取下三分钟秒了，但是WA一发后才认真看了下题目发现看错了，后面看对题目后我又急于求成，没考虑到序列中存在负数，结果又WA了，最后清醒过来改了下才AC。

代码:

```
#include<iostream>
#include<algorithm>
#include<string>
#include<string.h>
using namespace std;
typedef long long ll;
const int maxn=1e2+10;
int fun(int n){
    if(n<0)
    return -n;
    return n;
}
int flag[maxn];
int a[maxn];
int main(){
    int t;
    cin>>t;
    while(t--){
        memset(flag,0,sizeof(flag));
        int n;
        cin>>n;
        int sum=0;
        for(int i=0;i<n;i++){
            cin>>a[i];
            a[i]=fun(a[i]);
            flag[a[i]]++;
        }
        if(flag[0]>0)
        sum++;
        for(int i=1;i<=100;i++){
            if(flag[i]>=2){
                sum+=2;
            }
            if(flag[i]==1)
            sum++;
```

```
        }
        cout<<sum<<endl;
    }
}
```

B. Mirror in the String（1100）

题意：

给你一个字符串，你可以在字符串任意一处放一面镜子得到一个镜像的字符串，即类似于s1s2...sksksk-1...s1这样子，求字典数最小的字符串。

思路：

刚开始写的时候，我觉得只要是字典数递减就行了，后面直接WA了。但是自己脑子一根筋，一直陷在这个思路中。后面找到了cbba这样的特殊样例，我又以为是在递减的基础上找对称，即当相等的时候看下以相等的字符为对称轴对称的字符比大小，然后陷进去无法自拔了。最终都没有做出来，掉大分了！后面和别人交流了一下，发现bba这样的样例能够否决掉我的想法。参考了别人的思路，实际上非递增就可以实现功能，但是有一个特例需要注意，就是bba这种，这是非递增的字符串，但是答案是bb,不是bbaabb，这个需要特判一下。

不难的题目，但是第一发容易WA，确实挺多人是WA了，但是后面又找到规律就AC了。因此这道题对于我来说是一种思维壁垒，得积累下。

代码：

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
        int n;
        cin>>n;
        string s;
        cin>>s;
        int k=0;
        if(n==1||n>1&&s[0]==s[1])
        {
            cout<<s[0]<<s[0]<<endl;
            continue;
        }
        for(int i=1;i<n;i++){
            if(s[i-1]<s[i])
            break;
            else
            k=i;
        }
        for(int i=0;i<=k;i++)
        cout<<s[i];
        for(int i=k;i>=0;i--)
        cout<<s[i];
        cout<<endl;
    }
}
```

C. Representative Edges（1500）

题意：

给你一个序列，你可以将其中的数变成任意一个数，可以是double型的。让你进行最少的次数将其变成一个等差数列。

思路：

数据量非常小，可以使用O（N^3）算法。怎么切入呢？我们可以从等差数列是怎么确定的切入。

我们知道什么？假设存在等差数列，那我们知道任意两个数，与其中相隔的多少个数。这个就可以来确定一个等差数列，既然数据量不大，那我们就可以进行遍历，用i,j表示这选中的数，得到d=(a[j]-a[i])/(j-i),注意，可以是小数，所以记得用double类型。后面就进行得出这个等差数列，比较下需要改变多少个数，遍历O(N^2)次就得到答案了。

做到这里我都是没什么问题的，但是我却WA了。为什么捏？对于double来说，它是有精度限制的，比如就算是5.0/3.0*3.0与5.0是不相等的。怎么解决呢？用`fabs(a[k] - b[k]) >= 1e-9`这样的语句来处理(精度视题目而定)，fabs()返回的是绝对值。这样就能解决精度问题了。这在计算几何里面用的多。

代码：

```
#include<iostream>
#include<string.h>
#include <math.h>
using namespace std;
const int maxn=100;
double a[maxn],b[maxn];
const long double eps = 1e-9;
double min(double a,double b){
    return a>b?b:a;
}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin>>t;
    while(t--){
        int n;
        cin>>n;
        for(int i=0;i<n;i++)
            cin>>a[i];
        if(n==1)
        {
            cout<<0<<endl;
            continue;
        }
        int ans=100;
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                double d=(a[j]-a[i])/double(j-i);
                b[0]=a[i]-d*i;
                for(int k=1;k<n;k++){
                    b[k]=b[k-1]+d;
```

```
                }
                int sum=0;
                for(int k=0;k<n;k++)
                    if(fabs(a[k] - b[k]) >= 1e-9)  //不能用a[k]!=b[k]
                       sum++;
                ans=min(ans,sum);
                memset(b,0,sizeof(b));
            }
        }
        cout<<ans<<endl;
    }
}
```