Make Product Equal One 题解

> 计算学部十大打卡活动——“龙舞编程新春会”编程打卡（1月29日）
>
> 题目为Make Product Equal One

题面翻译

题目描述

给你一个有 n 个数的数组。你可以用 x(x为任意正整数) 的代价将数组中的任意一个数增加或减少 x ，你可以重复多次此操作。现在需要你用若干次操作使得 $a_1 \cdot a_2 \cdot \ldots \cdot a_n = 1$ （数组的乘积为1）。

比如，当 $n = 3$ 和数组为 [**1,-3,0**] 时，我们最少需要花费 3 的代价：用 2 的代价把 -3 增加到 -1 ，再用 1 的代价把 0 减少到 -1 ，数组就变成了 [**1,-1,-1**] ，然后 $1 \cdot (-1) \cdot (-1) = 1$ 。

现在询问最少需要花费多少的代价使得数组的乘积为 1 。

输入格式

输入共两行。

第一行输入一个数 n ，表示数组的数字个数。

第二行输入 n 个数 a_i ，表示该数组。

输出格式

输出一个数，表示使得数组的乘积为 1 的最少的花费。

数据范围

$1 \leq n \leq 10^5$

$-10^9 \leq a_i \leq 10^9$

题目描述

You are given n numbers $a_1, a_2, \ldots, a_n$. With a cost of one coin you can perform the following operation:

Choose one of these numbers and add or subtract 1 from it.

In particular, we can apply this operation to the same number several times.

We want to make the product of all these numbers equal to 1 , in other words, we want $a_1 \cdot a_2 \ldots \cdot a_n = 1$.

For example, for $n = 3$ and numbers $[1, -3, 0]$ we can make product equal to 1 in 3 coins: add 1 to second element, add 1 to second element again, subtract 1 from third element, so that array becomes $[1, -1, -1]$. And $1 \cdot (-1) \cdot (-1) = 1$.

What is the minimum cost we will have to pay to do that?

输入格式

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of numbers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the numbers.

输出格式

Output a single number — the minimal number of coins you need to pay to make the product equal to 1 .

样例 #1

样例输入 #1

```
2
-1 1
```

样例输出 #1

```
2
```

样例 #2

样例输入 #2

```
4
0 0 0 0
```

样例输出 #2

```
4
```

样例 #3

样例输入 #3

```
5
-5 -3 5 3 0
```

样例输出 #3

```
13
```

提示

In the first example, you can change 1 to -1 or -1 to 1 in 2 coins.

In the second example, you have to apply at least 4 operations for the product not to be 0 .

In the third example, you can change -5 to -1 in 4 coins, -3 to -1 in 2 coins, 5 to 1 in 4 coins, 3 to 1 in 2 coins, 0 to 1 in 1 coin.

思路

对于题目中的“乘积为1”的要求，因为我们知道满足“乘积为1”这一要求仅当所有数都为“1”或“-1”时才有可能满足。题目中求最小操作数，我们不妨贪心求解，分类讨论。

对于任意的 x：

- **若** `x >= 1`，则让它变为“1”，`操作数 += x - 1`；
- **若** `x <= -1`，则让它变成“-1”，`操作数 += -1 - x`；
- **若** `-1 < x < 1`，**即** `x == 0`，则它既可以变成“1”，又可以变成“-1”，且都有 `操作数 += 1`。

在按照上述操作执行时，我们同时**统计“0”和变成“-1”的数字个数。**

首先判断是否有“0”出现：因为“0”是“万能数”，它既可以变成“1”也可以变成“-1”，当乘积为“-1”是可以变成“-1”使乘积为“1”，所以说只要有“0”出现，就可以直接输出操作数。

若上述判断为“否”，则继续判断“-1”的个数是否为偶数：若为偶数，则乘积已经为“1”，符合题意，直接输出。若非偶数，则说明现在所有数的乘积为“-1”，不符合题意，我们这时可以把其中的一个“1”变成“-1”，或是把其中的一个“-1”变成“1”，都有 **操作数 += 2。**

样例讲解：

样例#4：

```
//样例输入#4：
5
-5 -3 -4 3 0
```

按照上述所有操作，我们执行完第一步后应为：

```
//第一步：
-1 -1 -1 1 0(1/-1)
//统计个数
ans = 4+2+3+2+1 = 12;    //操作数（答案）
sum_1 = 3;       //-1的个数
sum0 = 1;        //0的个数
```

我们发现存在“0”，故输出12(ans)

样例#5：

```
//样例输入#5：
5
-5 -3 -4 3 4
```

按照上述所有操作，我们执行完第一步后应为：

```
//第一步：
-1 -1 -1 1 1
//统计个数
ans = 4+2+3+2+3 = 14;    //操作数（答案）
sum_1 = 3;       //-1的个数
sum0 = 0;        //0的个数
```

我们发现没有“0”的存在且“-1”的个数为奇数，现在的乘积为“-1”，我们需要把其中的一个“1”变成“-1”或是把其中的一个“-1”变成“1”。

```
//第一步：
1 -1 -1 1 1
//统计个数
ans = 14+2 = 16;     //操作数（答案）
sum_1 = 4;       //-1的个数
```

输出16(ans)。

参考代码

```
#include <cstdio>
int main(void)
{
    int n;
    scanf("%d", &n);
    long long sum_1=0, sum0=0, ans=0;    // 注意数据范围
    for (int i = 1; i <= n; i++) {
        long long x;
        scanf("%lld", &x);
        if (x >= 1) {
            ans += x - 1;
        }
        else if (x <= -1) {
            ans += - 1 - x;
            sum_1++;
        }
        else if (x == 0) {
            ans += 1;
            sum0++;
        }
    }
    if (sum0 == 0 && sum_1 % 2 == 1) {
        ans += 2;
    }
    printf("%lld", ans);
    return 0;
}
```