贪心

一、AcWing 1055. 股票买卖 II

【题目描述】

给定一个长度为N的数组，数组中的第i个数字表示一个给定股票在第i天的价格。

设计一个算法来计算你所能获取的最大利润。你可以尽可能地完成更多的交易（多次买卖一支股票）。

注意：你不能同时参与多笔交易（你必须在再次购买前出售掉之前的股票）。

【输入格式】

第一行包含整数N，表示数组长度。

第二行包含N个不大于10000的正整数，表示完整的数组。

【输出格式】

输出一个整数，表示最大利润。

【数据范围】

$1 \leq N \leq 10^5$

【输入样例1】

```
6
7 1 5 3 6 4
```

【输出样例1】

```
7
```

【输入样例2】

```
5
1 2 3 4 5
```

【输出样例2】

```
4
```

【输入样例3】

```
5
7 6 4 3 1
```

【输出样例3】

```
0
```

【分析】

任何时间跨度大于一天的交易都可以拆分成多个时间跨度为一天的交易，那么只要后一天的价格比前一天高，那么我们就在前一天买，后一天卖，因此利润就加上$a[i+1]-a[i]$，枚举所有跨度为一天的交易，如果利润大于0就进行交易，最后的利润即为最大值。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 100010;
int a[N];
int n, res;

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i];
    for (int i = 0; i < n - 1; i++)
        if (a[i + 1] - a[i] > 0) res += a[i + 1] - a[i];
    cout << res << endl;
    return 0;
}
```

二、AcWing 104. 货仓选址

【题目描述】

在一条数轴上有N家商店，它们的坐标分别为$A_1 \sim A_N$。

现在需要在数轴上建立一家货仓，每天清晨，从货仓到每家商店都要运送一车商品。

为了提高效率，求把货仓建在何处，可以使得货仓到每家商店的距离之和最小。

【输入格式】

第一行输入整数N。

第二行N个整数$A_1 \sim A_N$。

【输出格式】

输出一个整数，表示距离之和的最小值。

【数据范围】

$1 \leq N \leq 100000$

$0 \leq A_i \leq 40000$

【输入样例】

```
4
6 2 9 1
```

【输出样例】

```
12
```

【分析】

货仓的坐标为各商店坐标的中位数~

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 100010;
int a[N];
int n, res;

```

```cpp
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> a[i];
    nth_element(a, a + n / 2, a + n);
    for (int i = 0; i < n; i++) res += abs(a[i] - a[n / 2]);
    cout << res << endl;
    return 0;
}
```

三、AcWing 122. 糖果传递

【题目描述】

有n个小朋友坐成一圈，每人有$a[i]$个糖果。

每人只能给左右两人传递糖果。

每人每次传递一个糖果代价为1。

求使所有人获得均等糖果的最小代价。

【输入格式】

第一行输入一个正整数n，表示小朋友的个数。

接下来n行，每行一个整数$a[i]$，表示第i个小朋友初始得到的糖果的颗数。

【输出格式】

输出一个整数，表示最小代价。

【数据范围】

$1 \le n \le 1000000$

$0 \le a[i] \le 2 \times 10^9$

数据保证一定有解

【输入样例】

```
4
1
2
5
4
```

【输出样例】

```
4
```

【分析】

设1号给2号x_1个糖果，2号给3号x_2个，以此类推，设最后每人的糖果都为b个，$b=(a_1+\cdots+a_n)/n$

则本题要求的为最小化$|x_1|+|x_2|+\cdots+|x_n|$

则1号的糖果为$a_1-x_1+x_n=b$，2号的糖果为$a_2-x_2+x_1=b\ldots$

转换后方程为$x_1=x_n-(b-a_1)$，$x_2=x_n-(2b-a_1-a_2),\ldots,x_{n-1}=x_n-((n-1)b-a_1-a_2-\cdots-a_{n-1})$，$x_n=x_n-(nb-a_1-\cdots-a_n)$即为$x_n=x_n$，则该式可移除。

将方程组带入上式得$|x_n-(b-a_1)|+|x_n-(2b-a_1-a_2)|+\cdots+|x_n-0|$

记为$|x-c_0|+|x-c_1|+\cdots+|x-c_{n-1}|$，要使该式最小，则$x$应取$c$的中位数。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
const int N = 1000010;
LL s[N], c[N];
int n;

int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) { scanf("%d", &s[i]); s[i] += s[i - 1];
}
    LL b = s[n] / n;//平均值
    int k = 0;
    for (int i = 1; i <= n; i++) c[k++] = i * b - s[i];
    nth_element(c, c + k / 2, c + k);
    LL res = 0;

    for (int i = 0; i < k; i++) res += abs(c[i] - c[k / 2]);
```

```
    printf("%lld\n", res);
    return 0;
}
```

四、AcWing 112. 雷达设备

【题目描述】

假设海岸是一条无限长的直线，陆地位于海岸的一侧，海洋位于另外一侧。

每个小岛都位于海洋一侧的某个点上。

雷达装置均位于海岸线上，且雷达的监测范围为d，当小岛与某雷达的距离不超过d时，该小岛可以被雷达覆盖。

我们使用笛卡尔坐标系，定义海岸线为x轴，海的一侧在x轴上方，陆地一侧在x轴下方。

现在给出每个小岛的具体坐标以及雷达的检测范围，请你求出能够使所有小岛都被雷达覆盖所需的最小雷达数目。

【输入格式】

第一行输入两个整数n和d，分别代表小岛数目和雷达检测范围。

接下来n行，每行输入两个整数，分别代表小岛的x,y轴坐标。

同一行数据之间用空格隔开。

【输出格式】

输出一个整数，代表所需的最小雷达数目，若没有解决方案则所需数目输出-1。

【数据范围】

$1 \leq n \leq 1000$

$-1000 \leq x,y \leq 1000$

【输入样例】

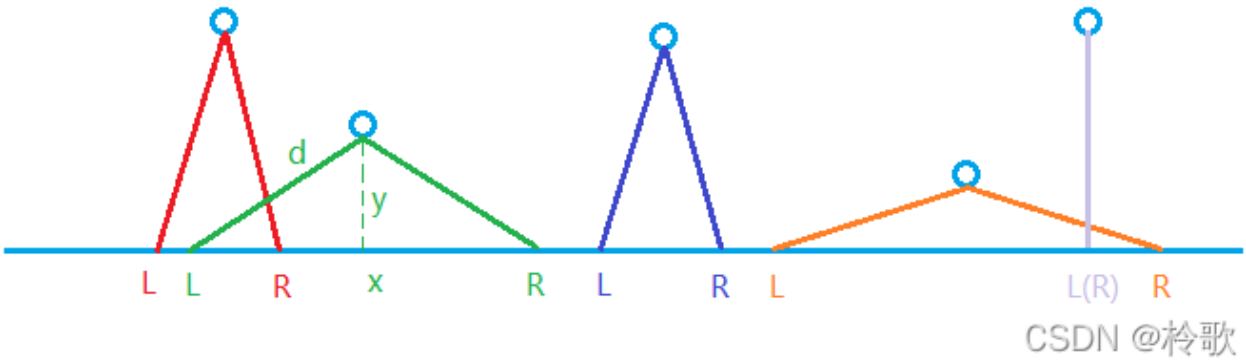
```
3 2
1 2
-3 1
2 1
```

【输出样例】

```
2
```

【分析】

每个小岛在海岸线上都有一个区间$[l,r]$，当把雷达放在这个区间内时小岛就能被覆盖，易知区间的两个端点为：$l=x-(d^2-y^2), r=x+(d^2-y^2)$，示意图如下：



那么问题就转化成：给定若干区间，最少选择多少个点，使得每个区间上至少包含一个选中的点。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

const int N = 1010;
int n, d;
bool st;

struct Edge
{
    double l, r;
    bool operator< (const Edge& t) const
    {
        return r < t.r;
    }
}e[N];

int main()

{
```

```
    cin >> n >> d;
    for (int i = 0; i < n; i++)
    {
        int x, y;
        cin >> x >> y;
        if (y > d) st = true;
        else
        {
            double t = sqrt(d * d - y * y);
            e[i].l = x - t, e[i].r = x + t;
        }
    }
    if (st) { cout << -1 << endl; return 0; }
    sort(e, e + n);
    int cnt = 0;
    double ed = -1e18;
    for (int i = 0; i < n; i++)
        if (ed < e[i].l) { cnt++, ed = e[i].r; }
    cout << cnt << endl;
    return 0;
}
```

五、AcWing 1235. 付账问题

【题目描述】

几个人一起出去吃饭是常有的事。

但在结帐的时候，常常会出现一些争执。

现在有n个人出去吃饭，他们总共消费了S元。

其中第i个人带了a_i元。

幸运的是，所有人带的钱的总数是足够付账的，但现在问题来了：每个人分别要出多少钱呢？

为了公平起见，我们希望在总付钱量恰好为S的前提下，最后每个人付的钱的标准差最小。

这里我们约定，每个人支付的钱数可以是任意非负实数，即可以不是1分钱的整数倍。

你需要输出最小的标准差是多少。

标准差的介绍：标准差是多个数与它们平均数差值的平方平均数，一般用于刻画这些数之间的“偏差有多大”。

形式化地说，设第i个人付的钱为b_i元，那么标准差为：

$s=\sqrt{\frac{(b_1-S/n)^2+(b_2-S/n)^2+\cdots+(b_n-S/n)^2}{n}}$。

【输入格式】

第一行包含两个整数n,S；

第二行包含n个非负整数$a_1,\ldots,a_n$。

【输出格式】

输出最小的标准差，四舍五入保留4位小数。

【数据范围】

$1\le n\le 5\times 10^5$

$0\le a_i\le 10^9$

$0\le S\le 10^{15}$

【输入样例1】

```
5 2333
666 666 666 666 666
```

【输出样例1】

```
0.0000
```

【输入样例2】

```
10 30
2 1 4 7 4 8 3 6 4 7
```

【输出样例2】

```
0.7928
```

【分析】

设$x_1+x_2+\cdots+x_n=c$为一个定值，则：$\frac{x_1^2+x_2^2+\cdots+x_n^2}{n}\ge(\frac{x_1+x_2+\cdots+x_n}{n})^2$，且$x_1=x_2=\cdots=x_n=c/n$时取到最小值。

令$b_i - S/n = x_i$，则$x_1 + x_2 + \cdots + x_n = 0$，因此$\sqrt{\frac{(b_1-S/n)^2+(b_2-S/n)^2+\cdots+(b_n-S/n)^2}{n}} \geq 0$。要使式子最小化，应使$b_i$最接近$S/n$。

对于第i个同学，有两种情况：

- $a_i \geq S/n$：那么该同学出的钱即为平均数，即$b_i = S/n$；
- $a_i < S/n$：那么该同学有多少钱就出多少钱，即$b_i = a_i$。

我们记$avg = S/n$，然后按钱数从小到大枚举每个同学，假设当前总共还需要出S'元，还剩下n'个同学需要出钱，那么当前同学i理论需要出的钱为$cur = S'/n'$，如果该同学的钱a_i不足cur，那么就出a_i元，然后计算一下$t+ = (a_i - avg)^2$，最后要更新一下$S' - = a_i, n' - -$。$\sqrt{t/n}$即为答案。

注意本题使用`double`的精度不够，需要使用`long double`。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

const int N = 500010;
int n, a[N];
long double s;

int main()
{
    scanf("%d%llf", &n, &s);
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);
    sort(a, a + n);
    long double t = 0, avg = s / n;
    for (int i = 0; i < n; i++)
    {
        double cur = s / (n - i);//当前每个同学需要均摊的钱数
        if (a[i] < cur) cur = a[i];
        t += (cur - avg) * (cur - avg);
        s -= cur;
    }
    printf("%.4llf\n", sqrt(t / n));
    return 0;

}
```

六、AcWing 1239. 乘积最大

【题目描述】

给定N个整数$A_1, A_2, \ldots, A_N$。

请你从中选出K个数，使其乘积最大。

请你求出最大的乘积，由于乘积可能超出整型范围，你只需输出乘积除以10^9+9的余数。

注意，如果$X<0$，我们定义X除以10^9+9的余数是$(-X)$除以10^9+9的余数，即：$0-((0-x)\%(10^9+9))$。

【输入格式】

第一行包含两个整数N和K。

以下N行每行一个整数A_i。

【输出格式】

输出一个整数，表示答案。

【数据范围】

$1 \leq K \leq N \leq 10^5$

$-10^5 \leq A_i \leq 10^5$

【输入样例1】

```
5 3
-100000
-10000
2
100000
10000
```

【输出样例1】

```
999100009
```

【输入样例2】

```
5 3
-100000
-100000
-2
-100000
-100000
```

【输出样例2】

```
-999999829
```

【分析】

首先我们先将整个序列从小到大排好序，然后分情况讨论：

- $k = n$：则答案就是整个序列的乘积；
- $k \neq n$：需要分别讨论k为奇数与k为偶数的情况：

（1）k为偶数：则最终结果一定非负，我们成对枚举负数或正数，假设当前l指向序列的最左边，r指向最右边，那么我们需要选$a[l] * a[l+1]$与$a[r] * a[r-1]$中最大的，然后将相应的指针向后移两位，不断重复此操作直到选出k个数为止；

（2）k为奇数：首先我们先把最大的数选了，即$a[r]$，然后判断$a[r]$是否为负数，如果为负数，说明整个序列都是负数，那么最后的结果一定为负，因此我们成对选择的时候需要选择乘积更小的；否则最后的结果一定还是非负，那么选法同（1）。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
const int N = 100010, MOD = 1e9 + 9;
LL a[N];
int n, k;

int main()
{
    cin >> n >> k;

    for (int i = 0; i < n; i++) cin >> a[i];
```

```
    sort(a, a + n);
    LL res = 1;
    int l = 0, r = n - 1, sign = 1;//如果序列全为负数则sign为-1
    if (k % 2)//k为奇数就先把最大的数选了
    {
        res = a[r--];
        k--;
        if (res < 0) sign = -1;
    }
    while (k)//k为偶数的情况
    {
        LL x = a[l] * a[l + 1], y = a[r] * a[r - 1];
        if (x * sign > y * sign)
        {
            res = x % MOD * res % MOD;//注意要先取余数
            l += 2;
        }
        else
        {
            res = y % MOD * res % MOD;
            r -= 2;
        }
        k -= 2;
    }
    cout << res << endl;
    return 0;
}
```

七、AcWing 1247. 后缀表达式

【题目描述】

给定N个加号、M个减号以及$N+M+1$个整数$A_1,A_2,\ldots,A_{N+M+1}$，小明想知道在所有由这N个加号、M个减号以及$N+M+1$个整数凑出的合法的后缀表达式中，结果最大的是哪一个？

请你输出这个最大的结果。

例如使用$123+-$，则`23+1-`这个后缀表达式结果是4，是最大的。

【输入格式】

第一行包含两个整数N和M。

第二行包含$N+M+1$个整数$A_1,A_2,\ldots,A_{N+M+1}$。

【输出格式】

输出一个整数，代表答案。

【数据范围】

$0 \leq N, M \leq 10^5$

$-10^9 \leq A_i \leq 10^9$

【输入样例】

```
1 1
1 2 3
```

【输出样例】

```
4
```

【分析】

后缀表达式是可以改变运算的顺序的，也就是可以给式子加括号，假设我们有N个加号，M个减号，那么可以有如下式子：

- $b - a_1 - a_2 - a_3 - a_4 + a_5 + a_6$
- $b - (a_1 - a_2 - a_3 - a_4) + a_5 + a_6$
- $b - a_1 - a_2 - a_3 - (a_4 + a_5 + a_6)$

因此我们可以凑出$1 \sim N + M$中的任意数量的负号，同理加号也是如此，那么我们分以下两种情况讨论：

1. $M = 0$，即全都为加号，那么答案就是所有数之和；
2. $M \neq 0$，那么我们由于至少需要加一个数减一个数，那么就用序列中的最大值$maxv$减去最小值$minv$，然后剩余的数中既可以加也可以减，那么就加上正数减去负数，即加上剩余所有数的绝对值就为答案。

【代码】

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

```

```cpp
typedef long long LL;
const int N = 200010;
LL a[N];
int n, m;

int main()
{
    cin >> n >> m;
    int cnt = n + m + 1;
    for (int i = 0; i < cnt; i++) cin >> a[i];
    sort(a, a + cnt);
    LL res = 0;
    if (!m)//特判负号为0的情况
        for (int i = 0; i < cnt; i++) res += a[i];
    else
    {
        res = a[cnt - 1] - a[0];//加上一个最大的数再减去一个最小的数
        for (int i = 1; i < cnt - 1; i++) res += abs(a[i]);
    }
    cout << res << endl;
    return 0;
}
```

八、AcWing 1248. 灵能传输

【题目描述】

在游戏《星际争霸II》中，高阶圣堂武士作为星灵的重要AOE单位，在游戏的中后期发挥着重要的作用，其技能“灵能风暴”可以消耗大量的灵能对一片区域内的敌军造成毁灭性的伤害。

经常用于对抗人类的生化部队和虫族的刺蛇飞龙等低血量单位。

你控制着n名高阶圣堂武士，方便起见标为$1,2,\ldots,n$。

每名高阶圣堂武士需要一定的灵能来战斗，每个人有一个灵能值a_i表示其拥有的灵能的多少（a_i非负表示这名高阶圣堂武士比在最佳状态下多余了a_i点灵能，a_i为负则表示这名高阶圣堂武士还需要$-a_i$点灵能才能到达最佳战斗状态）。

现在系统赋予了你的高阶圣堂武士一个能力：传递灵能。每次你可以选择一个$i\in[2,n-1]$，若$a_i\geq 0$则其两旁的高阶圣堂武士，也就是$i-1,i+1$这两名高阶圣堂武士会从i这名高阶圣堂武士这里各抽取a_i点灵能；若$a_i<0$则其两旁的高阶圣堂武士，也就是$i-1,i+1$这两名高阶圣堂武士会给i这名高阶圣堂武士$-a_i$点灵能。

形式化来讲就是$a_{i-1}+=a_i,a_{i+1}+=a_i,a_i-=2a_i$。

灵能是非常高效的作战工具，同时也非常危险且不稳定，一位高阶圣堂武士拥有的灵能过多或者过少都不好，定义一组高阶圣堂武士的不稳定度为$max_{i=1}^{n}|a_i|$，请你通过不限次数的传递灵能操作使得你控制的这一组高阶圣堂武士的不稳定度最小。

【输入格式】

本题包含多组询问。输入的第一行包含一个正整数T表示询问组数。

接下来依次输入每一组询问。

每组询问的第一行包含一个正整数n，表示高阶圣堂武士的数量。

接下来一行包含n个数$a_1,a_2,\ldots,a_n$。

【输出格式】

输出T行。

每行一个整数依次表示每组询问的答案。

【数据范围】

$1\leq T\leq 3$

$3\leq n\leq 300000$

$|a_i|\leq 10^9$

【输入样例1】

```
3
3
5 -2 3
4
0 0 0 0
3
1 2 3
```

【输出样例1】

```
3
0
3
```

【输入样例2】

```
3
4
-1 -2 -3 7
4
2 3 4 -8
5
-1 -1 6 -1 -1
```

【输出样例2】

```
5
7
4
```

【分析】

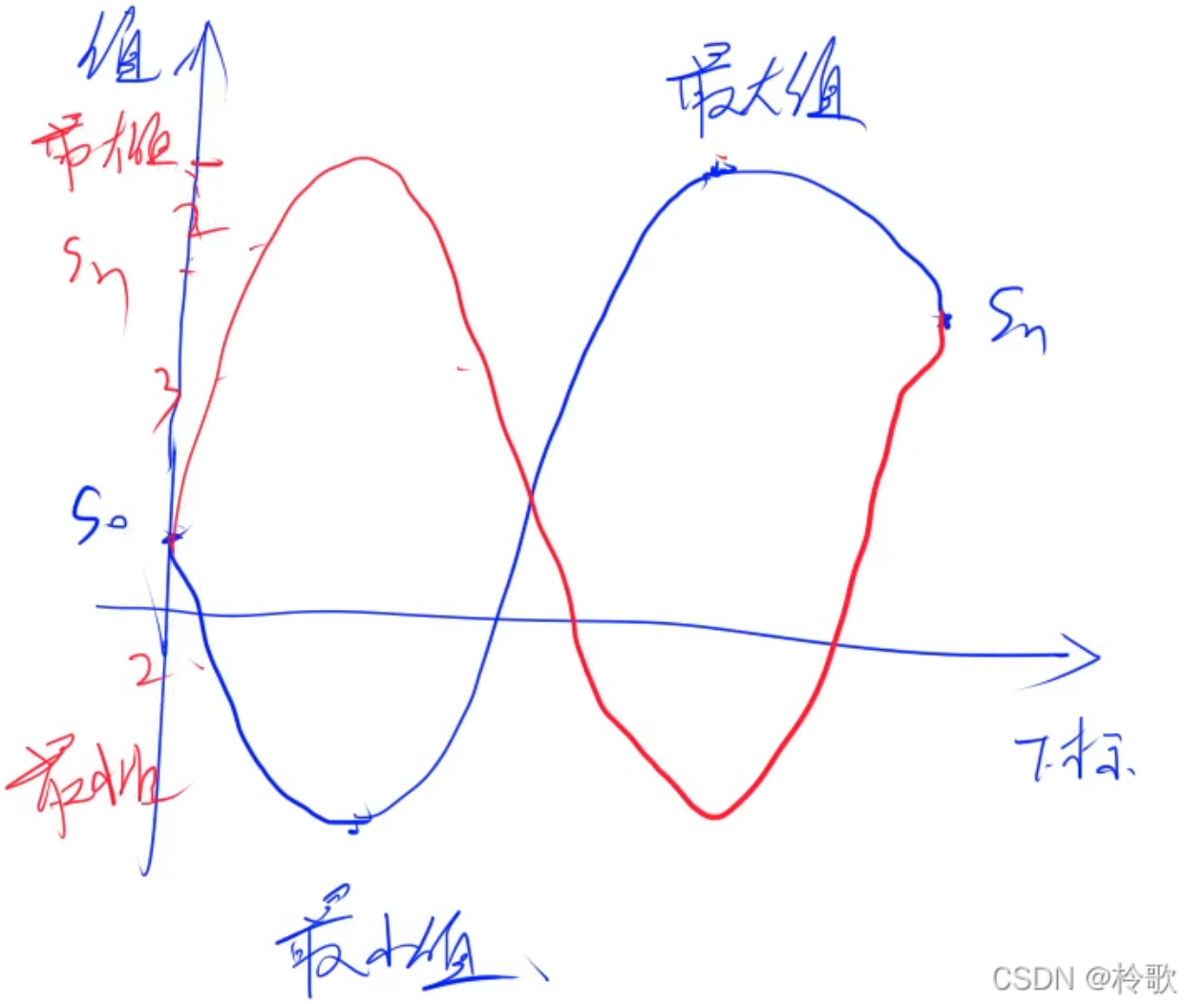
假设s为数组a的前缀和，那么每次进行操作时：$a_{i-1}+=a_i$，也就是s_{i-1}变成了s_i；$a_i-=2a_i$，其中有一个a_i加到了a_{i+1}上，也就是s_i变成了s_{i-1}；s_{i+1}不变。因此每次操作会交换s_{i-1}和s_i。

我们需要求的是$max(|a_i|)$的最小值，也就是$max(|s_i-s_{i-1}|)$的最小值，那么假设s是有序的，显然$max(|s_i-s_{i-1}|)$就是最小的，但是本题第一个和最后一个武士不能传输灵能，也就是s_0和s_n的位置是不能变的，这样就产生了新的问题。我们最终得到的一个序列并不一定是单调的，所以接下来我们就要通过一系列操作解决不单调序列的问题。

通过画图可知一个有两个拐点的曲线重叠部分最小时，单调部分最多，而一个曲线符合下列两种情况时便符合最优解的要求：

1. 左端点小于右端点，即要求$s_0<s_n$。如果不满足，那么我们交换一下两个数，从反方向的角度求解也是一样的；
2. 极小值在极大值左边，也就是曲线满足$s_0\to minv\to maxv\to s_n$。该点要求我们在后续选点的时应先从$s_0$向左取，从$s_n$向右取，因为只有这样才能取得两边的极值。

可以画图理解一下，如下图所示，显然蓝线比红线更优，因为红线中间那段重复走了三遍，而蓝线只走过一次。



【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

typedef long long LL;
const int N = 300010;
LL s[N], f[N];
bool st[N];
int n;

int main()
{
    int T;
    cin >> T;
    while (T--)
```

```
    {
        cin >> n;
        s[0] = 0;//多组测试数据注意初始化
        memset(st, false, sizeof st);
        for (int i = 1; i <= n; i++) { cin >> s[i]; s[i] += s[i - 1]; }
        LL s0 = s[0], sn = s[n];//注意要用LL
        if (s0 > sn) swap(s0, sn);
        sort(s, s + n + 1);
        for (int i = 0; i <= n; i++)
            if (s[i] == s0) { s0 = i; break; }//找出s[0]在排序后的位置
        for (int i = n; i >= 0; i--)
            if (s[i] == sn) { sn = i; break; }//找出s[n]在排序后的位置
        int l = 0, r = n;
        for (int i = s0; i >= 0; i -= 2)//从s[0]先向最小值跳
            f[l++] = s[i], st[i] = true;
        for (int i = sn; i <= n; i += 2)//从s[n]先向最大值跳
            f[r--] = s[i], st[i] = true;
        for (int i = 0; i <= n; i++)//将没遍历到的剩余的s[i]遍历一遍
            if (!st[i]) f[l++] = s[i];
        LL res = 0;
        for (int i = 1; i <= n; i++)
            res = max(res, abs(f[i] - f[i - 1]));
        cout << res << endl;
    }
    return 0;
}
```