

Lesson2

排序不等式

排队打水

思路：

- 总共等待时间为： $t_1 * (n-1) + t_2 * (n-2) + t_3 * (n-3) + \dots + t_{(n-1)} * 1$ ，且乘的系数 $n-1 > n-2 > \dots > 1$
- 故**从小到大排序**，总时间最小

证明：（反证法）

- 假设最优解不是从小到大排序，则必然存在两个相邻元素，前一个 t_i 比后一个 $t_{(i+1)}$ 大，此时我们交换 t_i 和 $t_{(i+1)}$ 的位置。交换前等待 t_i 和 $t_{(i+1)}$ 的时间为 $t_i * (n-i) + t_{(i+1)}(n-i-1)$ ，而交换后的等待时间为 $t_{(i+1)} * (n-i) + t_i * (n-i-1)$ ，且 $t_i * (n-i) + t_{(i+1)}(n-i-1) - [t_{(i+1)} * (n-i) + t_i * (n-i-1)] = t_i - t_{(i+1)} > 0$ ，所以交换 t_i 与 $t_{(i+1)}$ 之后，总时间将严格变小。与其是最优解矛盾，因此最优解一定是从小到大排序。

具体实现

```
1  static int N = 100010;
2
3  static int n;
4  static int[] t = new int[N];
5
6  public static void main(String[] args) throws Exception {
7      ins.nextToken(); n = (int)ins.nval;
8
9      for (int i=0; i<n; i++) { ins.nextToken(); t[i] = (int)ins.nval; }
10
11     Arrays.sort(t, 0, n);    //将t[i]从小到大排序
12
13     long res = 0;
14     for (int i=0; i<n; i++) res += t[i]*(n-i-1);
15
16     out.println(res);
17
18     out.flush();
19 }
```

绝对值不等式

货舱选址

思路

- $f(x) = |x_1 - x| + |x_2 - x| + \dots + |x_n - x|$
- 猜测：若 x_n 为奇数，则当 x 取到中位数时函数值最小；若 x_n 为偶数，则当 x 取到中间两个数之间（包括左右两个端点）时最小

证明：

- 将右侧分成若干组，变为

$$f(x) = (|x_1 - x| + |x_n - x|) + (|x_2 - x| + |x_{(n-1)} - x|) + \dots$$

考虑 $|a-x| + |b-x|$ ，当 x 落在 $[a, b]$ 之间时 $|a-x| + |b-x|$ 取最小值 $|b-a|$

所以上述 $f(x)$ 等式右侧 $\geq (x_n - x_1) + (x_{(n-1)} - x_2) + \dots$

对于没一项，取等的条件分别为 $x \in [x_1, x_n]$, $x \in [x_2, x_{(n-1)}]$, ...

所以若 x_n 为奇数，则当 x 取到中位数时函数值最小；若 x_n 为偶数，则当 x 取到中间两个数之间（包括左右两个端点）时最小。

具体实现

```
1  static int N = 100010;
2
3  static int n;
4  static int[] a = new int[N];
5
6  public static void main(String[] args) throws Exception {
7      ins.nextToken(); n = (int)ins.nval;
8
9      for (int i=0; i<n; i++) { ins.nextToken(); a[i] = (int)ins.nval; }
10
11     Arrays.sort(a, 0, n);    //从小到大排序，用于求中位数
12
13     long res = 0;
14     for (int i=0; i<n; i++) res += Math.abs(a[i]-a[(n-1)/2]);
15
16     out.println(res);
17
18     out.flush();
19 }
```

推公式

耍杂技的牛

思路：

- 按照 $w_i + s_i$ 从小到大的顺序排，最大的风险系数一定是最小的。

证明：($ans \geq cnt \ \&\& \ ans \leq cnt$)

- 该贪心得到的答案，一定大于等于最优解($ans \geq cnt$)

- (反证法) 若不是按照 w_i+s_i 从小到大的顺序排, 则一定存在相邻两头牛, 且 $w_i+s_i > w_{i+1}+s_{i+1}$, 交换第 i 个位置与第 $i+1$ 个位置上的牛。交换前, 第 i 个位置牛的危险系数为: $w_1+w_2+\dots+w_{i-1}-s_i$, 第 $i+1$ 个位置牛危险系数为: $w_1+w_2+\dots+w_i-s_{i+1}$ 。交换之后, 第 i 个位置牛的危险系数为: $w_1+w_2+\dots+w_{i-1}-s_{i+1}$, 第 $i+1$ 个位置牛危险系数为: $w_1+w_2+\dots+w_{i+1}-s_i$ 。每个位置危险系数同时减去 $w_1+w_2+\dots+w_{i-1}$, 加上 s_i 与 s_{i+1} , 所以现在每个位置的危险系数为 s_{i+1} [交换前 i], w_i+s_i [交换前 $i+1$], s_i [交换后 i], $w_{i+1}+s_{i+1}$ [交换后 $i+1$]。且交换后 s_i [交换后 i], $w_{i+1}+s_{i+1}$ [交换后 $i+1$]严格小于 w_i+s_i [交换前 $i+1$], 因此也严格小于 $\max(s_{i+1}$ [交换前 i], w_i+s_i [交换前 $i+1$])。所以交换之后, 其余所有的风险系数不变, 而交换之后的两个风险系数变小。重复这一操作, 序列中最大值一定不会变大, 只可能变小或者不变, 因此贪心得到的答案一定小于等于最优解, 即 $ans \leq cnt$
- 综上所述, $ans = cnt$

具体实现

```

1  static int N = 500010;
2
3  static int n;
4  static Cow[] cows = new Cow[N];
5
6  public static void main(String[] args) throws Exception {
7      ins.nextToken(); n = (int)ins.nval;
8
9      for (int i=0; i<n; i++) {
10         ins.nextToken(); int w = (int)ins.nval;
11         ins.nextToken(); int s = (int)ins.nval;
12         cows[i] = new Cow(w+s, w);
13     }
14
15     //按w+s从小到大排序
16     Arrays.sort(cows, 0, n, (o1, o2) -> (o1.l-o2.l));
17
18     long res = (int)-2e9, sum = 0;
19     for (int i=0; i<n; i++) {
20         int s = cows[i].l-cows[i].r, w = cows[i].r;
21         res = Math.max(res, sum-s);
22         sum += w;
23     }
24
25     out.println(res);
26
27     out.flush();
28 }
29
30 static class Cow {
31     int l, r;
32     Cow(int ll, int rr) {
33         l = ll; r = rr;
34     }
35 }

```

