复赛题型解析

2013~2017

目录

魔法阵（普及P4难度）

六十年一次的魔法战争就要开始了，大魔法师准备从附近的魔法场中汲取魔法能量。

大魔法师有 m 个魔法物品，编号分别为 $1,2,\ldots,m$。每个物品具有一个魔法值，我们用 x_i 表示编号为 i 的物品的魔法值。每个魔法值 x_i 是不超过 n 的正整数，可能有多个物品的魔法值相同。

大魔法师认为，当且仅当四个编号为 a,b,c,d 的魔法物品满足 $x_a < x_b < x_c < x_d$，$x_b - x_a = 2(x_d - x_c)$，并且 $x_b - x_a < (x_c - x_b) \div 3$ 时，这四个魔法物品形成了一个魔法阵，他称这四个魔法物品分别为这个魔法阵的A 物品，B 物品，C 物品，D 物品。

现在，大魔法师想要知道，对于每个魔法物品，作为某个魔法阵的A 物品出现的次数，作为B 物品的次数，作为C 物品的次数，和作为D 物品的次数。

$1 \le n \le 15000, 1 \le m \le 40000$

魔法阵（普及P4/提高P2+难度）

Sample Input	Sample Output	Sample Input2	Sample Output2
30 8 1 24 7 28 5 29 26 24	4 0 0 0 0 0 1 0 0 2 0 0 0 0 1 1 1 3 0 0 0 0 0 2 0 0 2 2 0 0 1 0	15 15 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	5 0 0 0 4 0 0 0 3 5 0 0 2 4 0 0 1 3 0 0 0 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 2 1 0 0 3 2 0 0 4 3 0 0 5 4 0 0 0 5

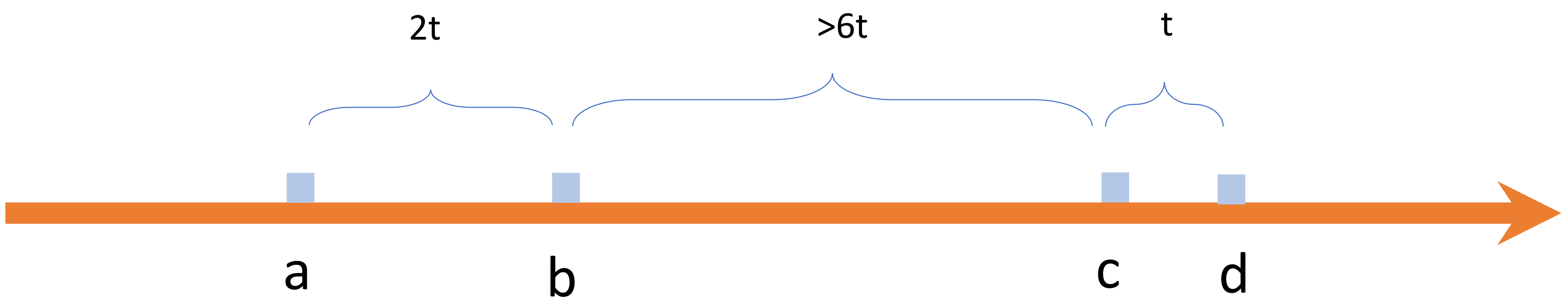
分析

$x_a < x_b < x_c < x_d$

$x_b - x_a = 2(x_d - x_c)$

$x_b - x_a < (x_c - x_b) \div 3$

- a、b、c、d满足单调递增关系
- 令最小的单元 $x_d - x_c = t$，可推出 $x_b - x_a = 2t$，$x_c - x_b > 6t$
- 因此我们可以枚举这个t，t的上限为 $t * 9 < n$
- 对每个t，枚举出a、d（可以对应确定b、c），枚举的时候需要考虑到a、d的合理范围



分析

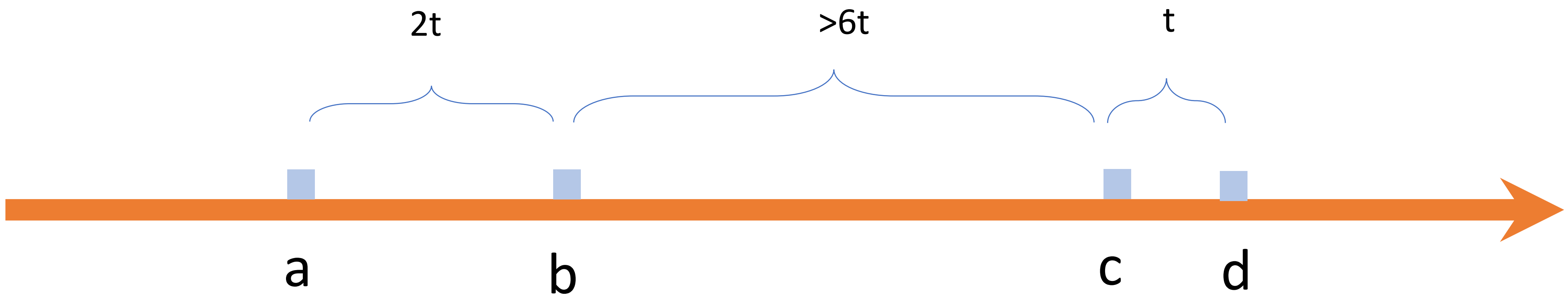
$$x_a < x_b < x_c < x_d$$

$$x_b - x_a = 2(x_d - x_c)$$

$$x_b - x_a < (x_c - x_b) \div 3$$

- 因为 n 最大才 15,000，所以可以直接开四个数组存每个物品的出现次数
- 设 a[i] 为 a 物品出现的次数，依乘法原理有

a[i] = b[i] * c[i] * d[i]



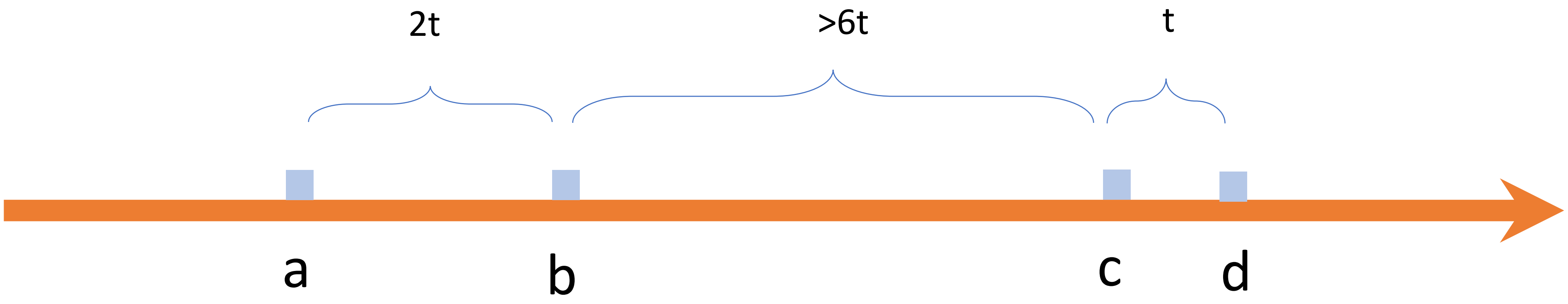
分析

$$x_a < x_b < x_c < x_d$$

$$x_b - x_a = 2(x_d - x_c)$$

$$x_b - x_a < (x_c - x_b) \div 3$$

- 这时 c[i] * d[i] 可以逆向前缀和预处理加速
- a 的枚举范围为[1, n - 9 * t - 1]（因为都是整数）
- 枚举 d 的时候，d[i] = a[i] * b[i] * c[i]，a[i] * b[i]用前缀和预处理加速
- d 的枚举范围为[9 * t + 2, n]



参考代码

- 读入时借助桶排思想

```
for (int i = 0; i < m; i ++)
    scanf("%d", &num[i]), vis[num[i]] ++;  //桶排思想
```

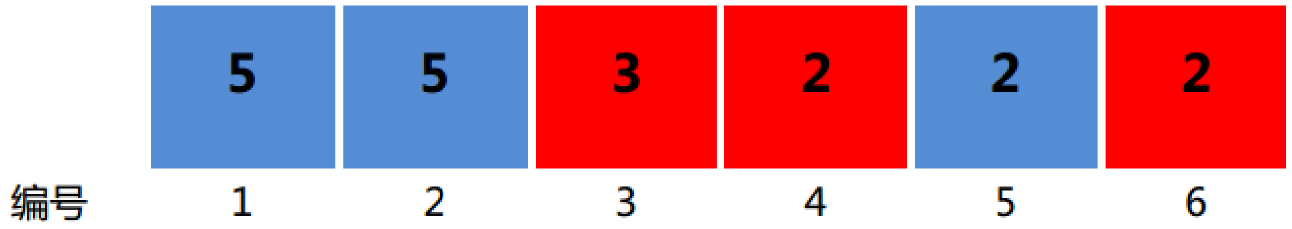
参考代码

- 枚举 d 时借助前缀和思想
- 枚举 a 时同样，不过要逆向循环

```
sum += vis[a] * vis[b];  //前缀和思想
C[c] += vis[d] * sum;
D[d] += vis[c] * sum;
```

求和（普及P3难度）

一条狭长的纸带被均匀划分出了 n 个格子，格子编号从 1 到 n。每个格子上都染了一种颜色$color_i$（用[1，m]当中的一个整数表示），并且写了一个数字$number_i$。



定义一种特殊的三元组：(x, y, z)，其中 x，y，z 都代表纸带上格子的编号，这里的三元组要求满足以下两个条件：

1. x，y，z都是整数，$x < y < z$，$y - x = z - y$
2. $color_x = color_z$

满足上述条件的**三元组的分数**规定为$(x + z) * (number_x + number_z)$。整个**纸带的分数**规定为所有满足条件的三元组的分数的和。这个分数可能会很大，你只要输出整个纸带的分数除以 10,007 所得的余数即可。

$1 \le n \le 100000, 1 \le m \le 100000$

求和（普及P3难度）

Sample Input	Sample Output
6 2 5 5 3 2 2 2 2 2 1 1 2 1	82
15 4 5 10 8 2 2 2 9 9 7 7 5 6 4 2 4 2 2 3 3 4 3 3 2 4 4 4 4 1 1 1	1388

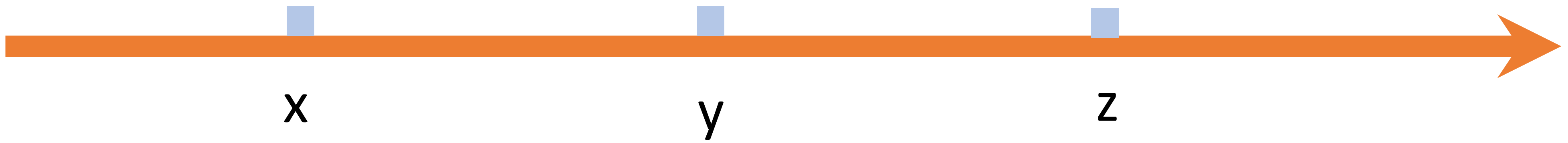
分析

$$(x + z) * (number_x + number_z)$$

- 如果直接枚举 x、y、z，O(n^3)的复杂度
- 其实我们看这个分数计算公式就知道，枚举 x、z即可，O(n^2)

（x，(x + z) / 2，z)

- 怎样可以做到更优呢？
- 还有一个条件，$color_x = color_z$
- 处理颜色和位置关系：同色、位置编号同奇偶的 x 和 z，因为(x + z) / 2为整数



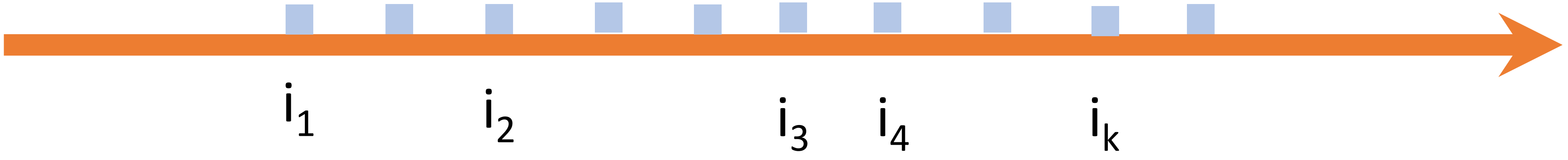
分析

$(x+z)*(number_x+number_z)$

$(x + z) * (number_x + number_z)$

$= x * number_x + x * number_z + z * number_x + z * number_z$

- 这个公式是对两个点的。那么我们考虑把同色的点分一组（只有这样的才有分数）
- 假设某个同色分组内有 k 个点：i_1、i_2、i_3、i_4、……、i_k



分析

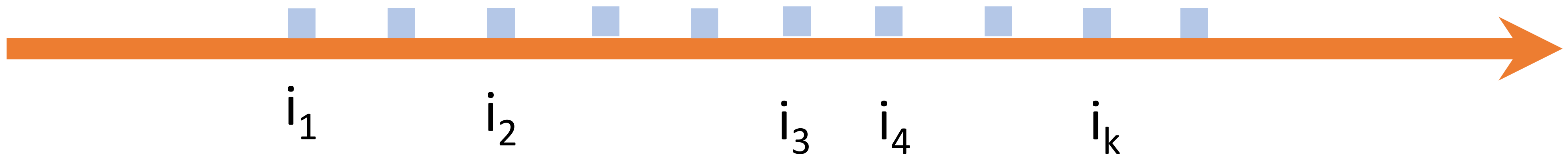
$(x+z) * (number_x + number_z)$

sum = $(i_1 * number_{i1} + i_1 * number_{i2} + i_2 * number_{i1} + i_2 * number_{i2}) + (i_1 * number_{i1} + i_1 * number_{i3} + i_3 * number_{i1} + i_3 * number_{i3}) + \ldots\ldots$

- 合并同类项（以 i_1 为例）：

$i_1 * number_{i1} + i_1 * number_{i2} + i_1 * number_{i1} + i_1 * number_{i3} + \ldots\ldots$

$= i_1 * (number_{i1} + number_{i2} + \ldots\ldots + number_{ik}) + (k-2) * i_1 * number_{i1}$



分析

$$(x+z) * (number_x + number_z)$$

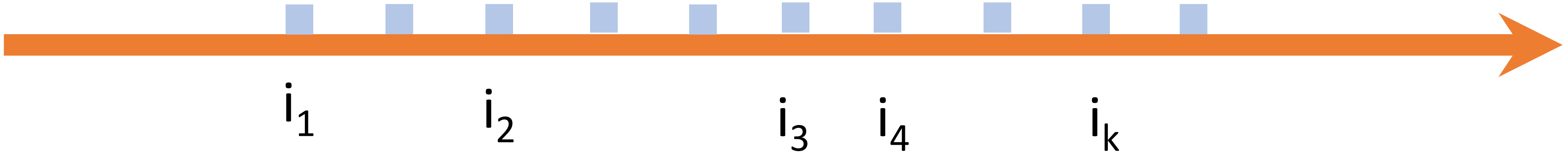
$i_1 * (number_{i1} + number_{i2} + \ldots\ldots + number_{ik}) + (k-2) * i_1 * number_{i1}$

- 推广开来，我们得到：

$$sum = \sum(i_j * number_{ij}) + \sum i_j * number_z + \sum number_{ij} * z + \sum(z * number_z)$$

$$(1 \le j \le k，枚举\ z)$$

- 或者也可以直接把 i_1、i_2、i_3、……、i_k 的得分加起来
- 结果非常大，中间过程记得也要取模



参考代码

- 开 [maxn][2] 的二维数组便于处理奇偶性，下标本身可以存储颜色

```
number[maxn], color[maxn], a[maxn][2], sum[maxn][2],
```

参考代码

- 按颜色和奇偶性分组

```
for (int i = 1; i <= n; i ++)
{
    scanf("%d", &color[i]);
    sum[color[i]][i % 2] += number[i];
    sum[color[i]][i % 2] %= mod;
    a[color[i]][i % 2] ++;
}
```

参考代码

- 按推导的公式计算

```
for (int i = 1; i <= n; i ++)
{
    ans += i % mod * ((sum[color[i]][i % 2] + (a[color[i]][i % 2] - 2) % mod * number[i] + mod) % mod);
    ans %= mod;
}
```

推销员（普及P4难度）

阿明是一名推销员，他奉命到螺丝街推销他们公司的产品。螺丝街是一条死胡同，出口与入口是同一个，街道的一侧是围墙，另一侧是住户。螺丝街一共有 N 家住户，第 i 家住户到入口的距离为 S_i 米。由于同一栋房子里可以有多家住户，所以可能有多家住户与入口的距离相等。阿明会从入口进入，依次向螺丝街的 X 家住户推销产品，然后再原路走出去。

阿明每走 1 米就会积累 1 点疲劳值，向第 i 家住户推销产品会积累 A_i 点疲劳值。阿明是工作狂，他想知道，对于不同的 X，**在不走多余的路的前提下**，他最多可以积累多少点疲劳值。

$1 \leqslant N \leqslant 100000$

推销员（普及P4难度）

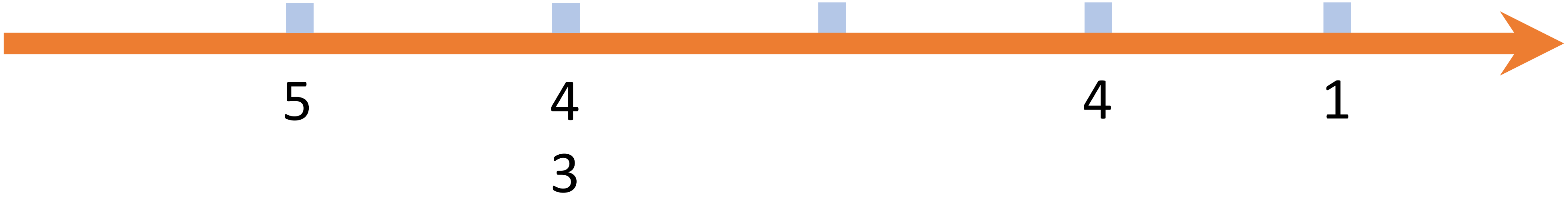
Sample Input	Sample Output
5 1 2 3 4 5 1 2 3 4 5	15 19 22 24 25
5 1 2 2 4 5 5 4 3 4 1	12 17 21 24 27

分析

- 一家住户对答案的贡献包括两方面：他本身的距离 d 和疲劳值 a。我们可以用式子来表示：

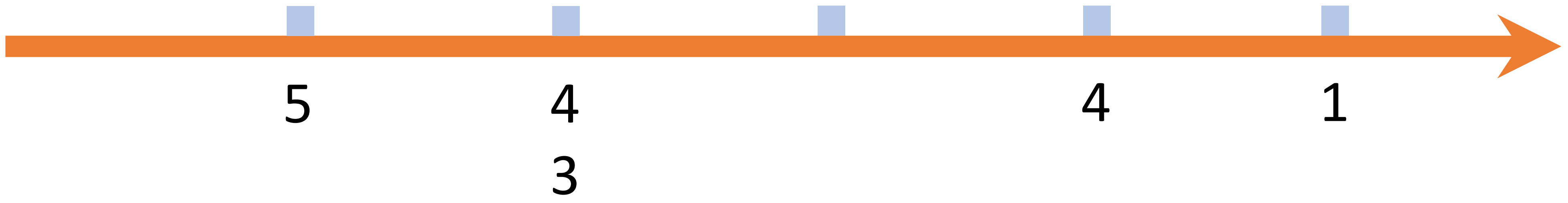
  $\max(0, 2 * (d_i - dist)) + a_i$ ，dist为当前走的最远距离

- 当 x = 1 时，肯定是 $2 * d_i + a_i$ 值最大的那个
- 当 x > 1 时，应该要在 x = 1 时选择的基础上再去选（贪心）
- 关于这个贪心思想，可以通过研究样例发现



分析

- 当 x = 2 时，应该要在 x = 1 时选择的基础上再去选。这时就存在向左走和向右走的问题：
  - 向左走：顺带推销，只贡献 a 值
  - 向右走：除了 a 值外还要加上多走的 d 值

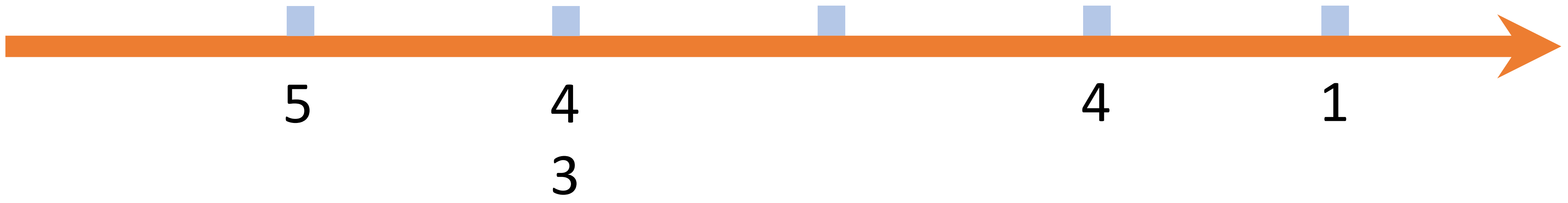


分析

– 向左走：顺带推销，只贡献 a 值
– 向右走：除了 a 值外还要加上多走的 d 值

- 这时我们可以模拟，也可以按 $2 * (d_i - dist) + a_i$ 和 a_i 分别排序，然后取 max。

- 最后要记得选择了一个最大点，加入答案之后要删除该点



参考代码

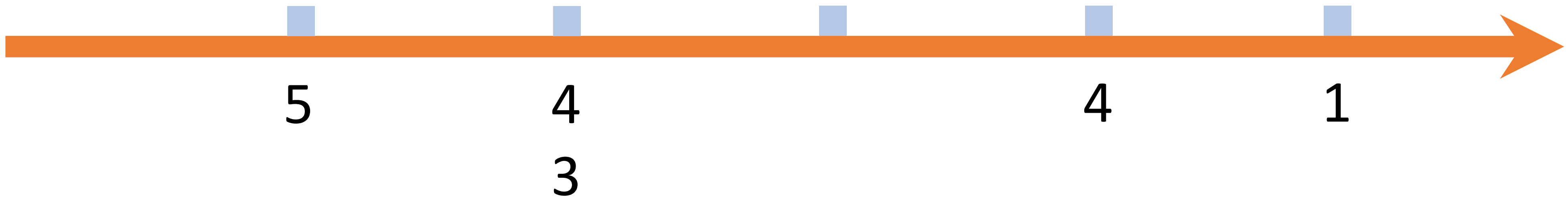
- 疲劳值相同但距离有不同，距离相同但疲劳值有不同

```
for (int i = 1; i <= n; i ++)
{
    scanf("%d", &temp);
    a[temp] ++;
    c[temp][a[temp]] = i;  // c[]数组处理疲劳值 a相同的住户
}
```

继续分析

- 这题除了模拟外，还可以：

1. 维护两个优先队列/堆，把找寻 max 的复杂度降到O(logn)，总复杂度O(nlogn)
2. 线段树/树状数组进行两端的区间维护，总复杂度也是O(nlogn)



比例简化（普及P2难度）

在社交媒体上，经常会看到针对某一个观点同意与否的民意调查以及结果。例如，对某一观点表示支持的有 1498 人，反对的有 902 人，那么赞同与反对的比例可以简单的记为 1498:902。

不过，如果把调查结果就以这种方式呈现出来，大多数人肯定不会满意。因为这个比例的数值太大，难以一眼看出它们的关系。对于上面这个例子，如果把比例记为 5:3，虽然与真实结果有一定的误差，但依然能够较为准确地反映调查结果，同时也显得比较直观。

现给出支持人数 A，反对人数 B，以及一个上限 L，请你将 A 比 B 化简为 A'比 B'，要求在 A'和 B'均不大于 L 且 A'和 B'互质（两个整数的最大公约数是 1）的前提下，$A'/B' \geq A/B$ 且 $A'/B' - A/B$ 的值尽可能小。

$1 \leqslant A \leqslant 1{,}000{,}000$，$1 \leqslant B \leqslant 1{,}000{,}000$，$1 \leqslant L \leqslant 100$

分析

Sample Input	Sample Output
1498 902 10	5 3

- L ≤ 100，所以直接枚举即可
- 程序中尽量回避除法以免产生浮点精度问题：

  A’ / B’ ≥ A / B，化为：A’ * B ≥ A * B’
- 这样枚举 [1，L] 间互质的A’、B’，带入上面的式子，找到最优的解即可
- （不过这题 double 也可以过关）

参考代码

```
int ansa = 100, ansb = 1;
for (int i = 1; i <= L; i ++)
    for (int j = 1; j <= L; j ++)
        if (gcd(i, j) == 1 && i * b >= j * a && ansa * j > ansb * i)
            ansa = i, ansb = j;
```

小朋友的数字（普及P3难度）

有 n 个小朋友排成一列。每个小朋友手上都有一个数字，这个数字可正可负。规定每个小朋友的特征值等于排在他前面（包括他本人）的小朋友中连续若干个（最少有一个）小朋友手上的数字之和的最大值。

作为这些小朋友的老师，你需要给每个小朋友一个分数，分数是这样规定的：第一个小朋友的分数是他的特征值，其它小朋友的分数为排在他前面的所有小朋友中（不包括他本人），小朋友分数加上其特征值的最大值。

请计算所有小朋友分数的最大值，**输出时保持最大值的符号，将其绝对值对 p 取模后输出。**

$1 \le n \le 1{,}000{,}000$，$1 \le p \le 10^9$

分析

- 典型的最大连续子段和模型
- 剩下的按题意模拟就好
- 在找最值的过程中可以用堆加速
- 另外注意数据范围，不开 long long 会被卡成50分

参考代码

```
for (int i = 1; i<= n; i++)
{
    scanf("%lld", &x);
    sum[i] = max(x, sum[i-1] + x);
    maxsum = max(sum[i], maxsum);  //最大连续子段和
    tz[i] = maxsum % p;  //存储特征值
}
```

参考代码

```
for (int i = 2; i <= n; i++)
{
    maxsum = max(tz[i-1] + score[i-1], maxsum);
    score[i] = maxsum;  //更新分数值
    if (ans < maxsum) ans = maxsum % p;
 }
```

组合数问题（提高P1难度）

组合数 C_n^m 表示的是从 n 个物品中选出 m 个物品的方案数。举个例子，从 $(1,2,3)$ 三个物品中选择两个物品可以有 $(1,2),(1,3),(2,3)$ 这三种选择方法。根据组合数的定义，我们可以给出计算组合数 C_n^m 的一般公式：

$$C_n^m = \frac{n!}{m!(n-m)!}$$

其中 $n! = 1 \times 2 \times \cdots \times n$。

小葱想知道如果给定 n, m 和 k，对于所有的 $0 \le i \le n, 0 \le j \le \min(i, m)$ 有多少对 (i, j) 满足 C_i^j 是 k 的倍数。

分析

- 首先我们要知道组合恒等式：$C(n,m) = C(n-1,m) + C(n-1,m-1)$
- 它同时也是杨辉三角的递推式

- 但是这题的数据范围决定了要做前缀和预处理：

$$sum[i][j] = sum[i][j-1] + sum[i-1][j] - sum[i-1][j-1]$$

- 另外要开 long long

Sample Input	Sample Output	Sample Input2	Sample Output2
1 2 3 3	1	2 5 4 5 6 7	0 7

参考代码

```
for (int i = 0; i <= 2000; i ++)
    f[i][i] = f[i][0] = 1;  //初始化
for (int i = 1; i <= 2000; i ++)
    for (int j = 1; j <= 2000; j ++)
    {
        f[i][j] = (f[i-1][j] + f[i-1][j-1]) % k;
        sum[i][j] = sum[i][j-1] + sum[i-1][j] - sum[i-1][j-1];  //前缀和
        if (f[i][j] == 0 && j <= i) sum[i][j] ++;
    }
for (int i = 1; i <= t; i ++)
{
    scanf("%lld%lld", &n, &m);
    printf("%lld\n", sum[n][m]);
}
```

跳石头（提高P1难度）

一年一度的“跳石头”比赛又要开始了！

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

$0 \leqslant M \leqslant N \leqslant 50{,}000$，$1 \leqslant L \leqslant 1{,}000{,}000{,}000$

分析

- 典型的最小值最大/最大值最小问题，二分答案。
- 注意到通常二分答案的题，数据范围都不小

- 可以二分的题，需要满足两个条件：

1. 可能的答案有明确的范围
2. 区间内的答案分布满足单调性

Sample Input	Sample Output
25 5 2 2 11 14 17 21	4

分析

- 具体到这道题，我们可以先“确定”一个距离，然后以它为标准去模拟：写一个 judge 函数来判断是否可行
- 如果可行，那么很有可能它不是最大的，就继续往右边二分；反之说明当前距离过大，就继续往左边二分
- judge 函数实现的功能：假定当前“确定”距离就是最短距离，然后计算需要移走的石头数量，最后和限制的数量做比较看是否符合要求

参考代码

```
int L = 1, R = d;
while (L <= R)
{
    int mid = (L + R) / 2;
    if (check(mid))
        ans = mid, L = mid + 1;
    else R = mid - 1;
}
```

参考代码

```
bool check(int x)
{
    int sum = 0, last = 0;
    for (int i = 1; i <= n; i ++)
        if (a[i] - a[last] < x) sum ++;
            else last = i;
    return sum <= m;
 }
```

联合权值（提高P2难度）

无向连通图 G 有 n 个点，n-1 条边。点从 1 到 n 依次编号，编号为 i 的点的权值为 W_i，每条边的长度均为 1。图上两点(u, v)的距离定义为 u 点到 v 点的最短距离。对于图 G 上的点对(u, v)，若它们的距离为 2，则它们之间会产生$W_u \times W_v$的联合权值。

请问图 G 上所有可产生联合权值的**有序点对**中，联合权值最大的是多少？所有联合权值之和是多少？

$1 < n \leq 200{,}000$，$0 < W_i \leq 10{,}000$

Sample Input	Sample Output
5 1 2 2 3 3 4 4 5 1 5 2 3 10	20 74

分析

- n 个点 n-1 条边，所以这是棵树，这样我们可以放心枚举而不必担心有环
- 距离为2，也就是中间间隔一个点的两点会产生权值

- 我们可以暴力枚举**中间点** x，然后一一枚举与 x 相连的点

- 复杂度O(n^2)，可以拿60分

分析

- 优化60分算法中统计的那部分：

1. 求最大值：权值最大的点和权值次大的点的乘积，就是经过这个点的最大权值点对
2. 求和：

$$sum = a_1a_2 + a_1a_3 + a_1a_4 + \ldots\ldots + a_2a_3 + a_2a_4 + \ldots\ldots$$
$$= a_1a_2 + (a_1 + a_2) * a_3 + (a_1 + a_2 + a_3) * a_4 + \ldots\ldots$$

分析

$$sum = a_1a_2 + (a_1 + a_2) * a_3 + (a_1 + a_2 + a_3) * a_4 + \ldots\ldots$$

- 所以我们可以枚举每一个元素，并将当前元素的权值与之前所有元素的权值和相乘，然后把所有的结果加起来即可

- 这里利用了乘法的结合律，同时还可以前缀和优化
- 时间复杂度O(n)

分析

- 然后这题有个非常大的坑：有序点对。
- 所以（a_1,a_2）和（a_2,a_1）是不同的点对，即统计的时候，要么就前向星存无向图然后遍历图，要么就计算 **sum * 2**

分析

- 也可以借助这个式子直接计算：

$$(a_1 + a_2 + a_3 + \ldots\ldots)^2 - （a_1{}^2 + a_2{}^2 + a_3{}^2 + \ldots\ldots）$$
$$= 2 * (a_1a_2 + a_1a_3 + a_2a_3 + \ldots\ldots)$$

- 要记得取模不然平方会爆

参考代码

```
struct edge
{
    int next;  // next记录下条边的编号
    int to;  // to记录当前边的终点
 } e[2 * N];
void add(int u, int v)
{
    e[++tot].next = head[u];  //head记录链首
    head[u] = tot;  //更新链首为新加入的边
    e[tot].to = v;  //新加入的边终点为 v
}


for (int i = 1; i <= n - 1; i ++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        add(u, v); add(v, u);   //读入无向图
    }
```

参考代码

```
for (int i = 1; i <= n; i ++)
{
    int max1 = 0, max2 = 0;  //最大权值和次大权值
    int s1 = 0, s2 = 0;  // s1和的平方, s2平方和
    for (int j = head[i]; j; j = e[j].next)
    {
        if (w[e[j].to] > max1)
            max2 = max1, max1 = w[e[j].to];
        else if (w[e[j].to] > max2)
            max2 = w[e[j].to];    //扫一遍统计出最大和次大
        s1 = (s1 + w[e[j].to]) % mod;
        s2 = (s2 + w[e[j].to] * w[e[j].to]) % mod;
     }
    s1 = s1 * s1 % mod;
    sum = (sum + s1 - s2 + mod) % mod; //有减法一定加上模数再取模
    if (maxsum < max1 * max2) maxsum = max1 * max2;
}
printf("%d %d\n", maxsum, sum);
```

细节

- 有减法一定记得加上模数再取模，否则有可能出现负数

```
sum = (sum + s1 - s2 + mod) % mod;
```