

# Chapter 1

## Atcoder

## 1.1 散场电影

### 1.1.1 问题描述

电影院有  $N$  排座椅，每排座椅有  $N$  个座位，这样整个电影院就有  $N^2$  个座位，每排都按相同方向从左向右计序号，第 1 排从左到右序号分别是  $1, 2, 3, \dots, N$ ，第 2 排从左到右分别为  $N + 1, N + 2, \dots, 2N$ ，最后一排分别是  $N^2 - N + 1, N^2 - N + 2, \dots, N^2$ 。

这一场电影坐无虚席。散场时所有观众将会按照序列  $\{P_i\}$  的顺序离场，但是如果  $x$  离场的时候，与他的座位相邻上下左右四个方向都还有其它观众没离场，那么  $x$  的离场就需要强行越过这个人出去，那么就会招致这个观众  $y$  对  $x$  的不满，将这一组不满意记录记为  $(x, y)$ 。此问题给出电影院的规模  $N$  和离场顺序，计算这个离场顺序最少有多少不满意记录。

### 1.1.2 分析

## 1.2 冷冻沙丁鱼

### 1.2.1 问题描述

钓了  $N$  条沙丁鱼，每条沙丁鱼的美味度为  $A$ ，鲜度为  $B$ 。现要将沙丁鱼存放到冰箱里冷冻。但是沙丁鱼的冷冻是有特殊要求的：如果有两条沙丁鱼  $i, j$  的美味度和鲜度满足： $A_i A_j + B_i B_j = 0$ ，则这两条鱼会急速腐化，导致整个冰箱中的鱼都坏掉，所以不能让这样的鱼同时冷冻在冰箱中。分别给出  $N$  条沙丁鱼的美味度  $A_i$  与鲜度  $B_i$ ，计算有多少种选法可以正常进行冷冻沙丁鱼而不致其腐化。（结果可能较大，所以输出对 1000000007 取余的值）

[https://vjudge.net/problem/AtCoder-abc168\\_e](https://vjudge.net/problem/AtCoder-abc168_e)

### 1.2.2 分析

$(A_i, B_i) = (0, 0)$  のイワシは他のどの個体とも仲が悪いので、「 $(A_i, B_i) = (0, 0)$  なイワシをどれか 1 匹だけ選ぶ」「 $(A_i, B_i) = (0, 0)$  なイワシをどれも選ばない」のいずれかです。勿論，前者の選び方はそのようなイワシの個数に等しいので，以下では  $(A_i, B_i) = (0, 0)$  なイワシを除外して考えます。

イワシの「傾き」を  $\frac{A_i}{B_i}$  として、これを既約分数で表すことを考えます。具体的には、次のように決めます

1.  $B_i = 0$  のとき、傾きは  $1/0$

2.  $B_i > 0$  のとき、傾きは  $\frac{A_i}{\gcd(A_i, B_i)} / \frac{B_i}{\gcd(A_i, B_i)}$

3.  $B_i < 0$  のとき、傾きは  $-\frac{A_i}{\gcd(A_i, B_i)} / -\frac{B_i}{\gcd(A_i, B_i)}$

ちょっとした場合分けから、傾きと仲の悪さの関係について次のことが言えます。

1. 傾き  $1/0$  のイワシと仲が悪いのは、傾き  $0/1$  のイワシ全てのみ。逆もまた然り。

斜率  $1/0$  の沙丁鱼会与斜率为  $0/1$  的发生腐化，反之亦然

2. 傾き  $a/b$  ( $a, b \neq 0$ ) のイワシと仲が悪いのは、傾き  $-b/a$  (を  $a$  の符号で通分したもの) のイワシ全てのみ。逆もまた然り。

$(A_i, B_i) = (0, 0)$  の沙丁鱼会和任意的沙丁鱼组合都会腐化，就有只选一只  $(A_i, B_i) = (0, 0)$  的沙丁鱼，和  $(A_i, B_i) = (0, 0)$  一只都不选这两类，显然，前者的选法和  $(A_i, B_i) = (0, 0)$  的沙丁鱼数量一样。接下来主要对后者情况讨论，即  $(A_i, B_i) \neq (0, 0)$ 。

定义沙丁鱼的“斜率”为  $\frac{A_i}{B_i}$ ，并且这个数值要表示为分数约分后的结果。更具体的规则如下所示

$B_i = 0$  时，斜率为  $1/0$

$B_i > 0$  时，斜率为  $\frac{A_i}{\gcd(A_i, B_i)} / \frac{B_i}{\gcd(A_i, B_i)}$

$B_i < 0$  时，斜率为  $\frac{A_i}{\gcd(A_i, B_i)} / \frac{B_i}{\gcd(A_i, B_i)}$

以下的情况中描述了斜率和相互腐化的关系

斜率为  $a/b$  ( $a, b \neq 0$ ) の沙丁魚会与斜率为  $-b/a$  的发生腐化, 反之亦然

たとえば以下の傾きが, 互いに”仲の悪い”  
がい”です.

例如以下几组斜率是相互被腐化的

$$1/0 \longleftrightarrow 0/1$$

$$5/3 \longleftrightarrow -3/5$$

$$2/1 \longleftrightarrow -1/2$$

したがって,”仲の悪い”になる傾きの  
ペア (高々  $N$  通り) 全てについて, 連想配列など  
を使って各傾きになるイワシの個数が求まれば,  
その後は基礎的な数え上げの範疇です. オーバー  
フローには気を付けてください.

$(A_i, B_i)$  の制約が大きいので, 傾きを有理数  
の代わりに実数で表現しようとする (long double  
でも) 精度が足りないおそれがあります.

因此, 可以使用组成“沙丁魚腐化斜率数对”(共  
 $N$  个), 把斜率出现的个数存储在关联数组 (例如  
C++ 的 map 或 Python 的 dict) 当中, 然后进行  
计数即可. 请小心数据溢出.

由于此题  $(A_i, B_i)$  的数值可能很大, 如果用相  
应的小数代替分数, 即使是用 long double 精度仍  
然不够.

```

1 #include <cstdio>
2 #include <map>
3
4 class ratNode
5 {
6     private:
7         long long num, den;
8         long long gcd(long long a, long long b)
9         {
10
11         }
12     public:
13         ratNode(long long n, long long d)
14         {
15             if (d==0)
16             {
17                 den=0;
18                 if (n>0)
19                     num=1;
20                 else if (n<0)
21                     num=-1;
22             else
23                 num=0;
24         }

```

```
25         else if (d>0)
26         {
27             long long g = gcd(n,d);
28             num = n/g;
29             den = d/g;
30         }
31         else
32         {
33             long long g = gcd(n,d);
34             num = -n/g;
35             den = -d/g;
36         }
37     }
38 }
39 bool operator< (const ratNode &o)
40 {
41
42 }
43 };
44
45 int main(void)
```

## 1.3 Pay to Win

### 1.3.1 问题描述

你现在手里只有数字 0，但你有非常多的筹码。

1. 花费 A 枚筹码将手中的数字变为 2 倍
2. 花费 B 枚筹码将手中的数字变为 3 倍
3. 花费 C 枚筹码将手中的数字变为 5 倍
4. 花费 D 枚筹码将手中的数字增加 1 或者减少 1.

你的筹码非常多，但是还是要省着用。这一场给定一个数字 N，试计算要将手中的 0 变为数字 N，最少消耗几枚筹码。

[https://atcoder.jp/contests/agc044/tasks/agc044\\_a](https://atcoder.jp/contests/agc044/tasks/agc044_a)

### 1.3.2 分析

我们可以把这个问题反过来想：手中初始是数字 N 最终变为 0。可以采取以下的操作：

1. 如果  $x$  能被 2 整除，则用  $\frac{x}{2}$  替代  $x$  并花费 A 枚筹码
2. 如果  $x$  能被 3 整除，则用  $\frac{x}{3}$  替代  $x$  并花费 B 枚筹码
3. 如果  $x$  能被 5 整除，则用  $\frac{x}{5}$  替代  $x$  并花费 C 枚筹码
4. 用  $x+1$  或  $x-1$  替代  $x$ ，花费 D 枚筹码

一种方法是，我们全用花费 D 的方法，这样从 N 到 0 的花费为  $ND$ 。另外，我们可以采用这样的策略：

$$N \xrightarrow{D} \dots \xrightarrow{D} ky \xrightarrow{k} y$$

其中  $k \in \{2, 3, 5\}$ 。需要说明的是  $y \in \{\lfloor \frac{N}{k} \rfloor, \lceil \frac{N}{k} \rceil\}$ 。事实上，如果  $y < \lfloor \frac{N}{k} \rfloor$  (同样的理由对  $y > \lceil \frac{N}{k} \rceil$  也适用)，就有如下的变换序列，比上述序列的花费更低

$$N \xrightarrow{D} \dots \xrightarrow{D} k \lfloor \frac{N}{k} \rfloor \xrightarrow{k} \lfloor \frac{N}{k} \rfloor \xrightarrow{D} \dots \xrightarrow{k} y$$

所以，我们的最优策略是直接利用  $\pm 1$  的方式由 N 到 0，或到某个  $\{\lfloor \frac{N}{k} \rfloor, \lceil \frac{N}{k} \rceil\}$  ( $k \in \{2, 3, 5\}$ )。特别地，如果记  $f(n)$  为从 N 到 0 的最小花费，那么如果我们知道了  $f(\lfloor \frac{N}{k} \rfloor)$  和  $f(\lceil \frac{N}{k} \rceil)$ ，那么  $f(n)$  便可容易计算出。

上述讨论可以用动态规划方法实现，但是还需要解决从 N 到 0 有多少状态的问题。可以证明，可达的数字将由以下两式限定

$$\lfloor \frac{N}{2^a 3^b 5^c} \rfloor, \lceil \frac{N}{2^a 3^b 5^c} \rceil$$

由于  $0 \leq a \leq 60, 0 \leq b \leq 40, 0 \leq c \leq 30$  (不然分母将大于分子)，可达的数字小于  $2 \times 61 \times 41 \times 31 = 155062$  (为了 Accepted，程序要保证  $f(n)$  总是在 n 的可达数字以内)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long LL;
4
5  LL A, B, C, D;
6  map<LL, LL> memo;
7  LL solve(LL N) {
8      if (N == 0) return 0;
9      if (N == 1) return D;
10     if (memo.count(N)) return memo[N];
11     LL l2 = (N/2)*2;
12     LL r2 = ((N+1)/2)*2;
13     LL l3 = (N/3)*3;
14     LL r3 = ((N+2)/3)*3;
15     LL l5 = (N/5)*5;
16     LL r5 = ((N+4)/5)*5;
17
18     LL res = 1e18;
19     if (N < res/D) res = N*D;
20     res = min(res, llabs(l2-N)*D + A + solve(l2/2));
21     res = min(res, llabs(r2-N)*D + A + solve(r2/2));
22
23     res = min(res, llabs(l3-N)*D + B + solve(l3/3));
24     res = min(res, llabs(r3-N)*D + B + solve(r3/3));
25
26     res = min(res, llabs(l5-N)*D + C + solve(l5/5));
27     res = min(res, llabs(r5-N)*D + C + solve(r5/5));
28
29     memo[N] = res;
30
31     return res;
32 }
33
34 int main() {
35     int T;
36     cin >> T;
37     for (int t = 0; t < T; t++) {
38         memo.clear();
39         LL N;
40         cin >> N;
41         cin >> A >> B >> C >> D;
42         cout << solve(N) << "\n";

```

```
43     }  
44     return 0;  
45 }
```