梦熊 J组模拟赛——第一套

回文立方数 （cube）

出题人想不到怎么给部分分。但是被要求给部分分。如果是先表所有的回文数再检查是不是立方的话，可以通过第一档。

只有 $O(N^{\frac{1}{3}})$ 个小于或等于 N 的立方数。因此，可以对所有立方数 N 进行暴力枚举，并检查其十进制表示是否是回文。

出题人良心的馈赠。

```
#include <bits/stdc++.h>
using namespace std;

#define int long long

int n;

namespace GTR {
    const int bufl = 1 << 15;
    char buf[bufl], *s = buf, *t = buf;
    inline int fetch() {
        if (s == t) { t = (s = buf) + fread(buf, 1, bufl, stdin); if (s == t)
return EOF; }
        return *s++;
    }
    inline int read() {
        int a = 0, b = 1, c = fetch();
        while (c < 48 || c > 57) b ^= c == '-', c = fetch();
        while (c >= 48 && c <= 57) a = (a << 1) + (a << 3) + c - 48, c =
fetch();
        return b ? a : -a;
    }
} using GTR::read;

int judge(int x) {
    int p[50] = {0};
    int m = 1;
    for ( ; x != 0; ++ m, x /= 10) p[m] = x % 10;
    -- m;
    for (int i = 1, j = m; i <= m / 2; ++ i, -- j) {
        if (p[i] != p[j]) {
            return 0;
        }
    }
    return 1;
}

signed main() {
    n = read();
    int ans = 0;
```

```
    for (int i = 0; i * i * i <= n; ++ i) {
        int x = i * i * i;
        if (judge(x)) {
            ans = x;
        }
    }
    printf("%lld\n", ans);
    return 0;
}
```

数学小店的奇妙兑换 (drink.cpp)

通过观察本题可以得知答案是 $\frac{n}{k-1}$，由于数字较大，我们可以使用一个大整数除法的模板来解决本题。

这个问题的本质是学习除法。在原始场景中（10 个饮料瓶，每 3 个瓶子换一瓶饮料，最终能喝 5 瓶饮料），我们可以看到，10 个饮料瓶最终喝了 5 瓶饮料。根据除法的定义：总价除以数量等于单价，得到每瓶饮料的单价是 $\frac{10}{5}=2$。得到单价之后，根据除法的定义继续：总价除以单价等于数量，也就是说假设有 n 元，则可以喝 $\frac{n}{2}$ 瓶饮料。

推广到每 k 个瓶子换一瓶饮料，可以简单地发现答案变为 $\frac{n}{k-1}$。

如果还需要证明的话，我们可以根据方程的思想来解决这个问题：

k 个空瓶子 = 1 瓶饮料 + 1 个空瓶子

两边同时减 1，解得

1 瓶饮料 = $k-1$ 个空瓶子

烧烤(bbq.cpp)

首先暴力肯定是没问题，出题人的良心已经体现在这额外的 10% 的分数中了。

下面直接介绍正解。

1. **回文判断**：首先通过哈希或者 manacher 快速判断起始字符串是否为回文串。如果是回文串，那么先手玩家立即输掉比赛。
2. **局面分析**：对于任意一个局面，若先手无法进行任何操作，则说明无论删去开头还是结尾都会得到回文串。我们可以发现符合条件的字符串只能形如 `ab`，`abab`，`ababab` 等，这说明终止状态的长度一定是偶数。因此，**输赢只和起始字符串长度的奇偶性有关。**
3. **时间复杂度**：该方法的时间复杂度为 $O(n+q)$，其中 n 是字符串的长度，q 是查询的数量。

minmax求和 (minmax.cpp)

对于 20% 的部分，直接暴力就好。

还送了额外 10% 的分数，A_i 都一样， 输出即可。无疑是出题人良心的馈赠。

为了简化问题，我们首先假设序列 A 中的元素是互不相同的。设 $maxA=M$。

我们要对所有 (i,j) 对进行求和，其中 $i<j$。由于加法是对称的，对于 i 和 j，我们可以重新排列序列 A 而不改变答案。现在我们假设 A 是按升序排列的。

当 A 按升序排列时，如果 $i<j$ 则 $A_i \leq A_j$，因此

$$\left\lfloor \frac{\min(A_i, A_j)}{\max(A_i, A_j)} \right\rfloor = \left\lfloor \frac{A_i}{A_j} \right\rfloor$$

对于固定的 i，考虑每一个 $\left\lfloor \frac{A_i}{A_j} \right\rfloor$ 而不是 i。也就是说，将其变形为

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left\lfloor \frac{A_i}{A_j} \right\rfloor = \sum_{i=1}^{N-1} \sum n \times f(A_i, n)$$

其中 $f(d,n)$ 是使得 $\left\lfloor \frac{d}{A_j} \right\rfloor = n$ 的 j 的数量。（这种替换相当于将例如“1+1+1+2+2+2+2+5+5”看作“1×3 + 2×4 + 3×0 + 4×0 + 5×2”。）

设 C_X 是 $A_i = X$ 的 i 的数量。通过预先计算 C 的累积和，可以在 $O(1)$ 时间内找到 $f(d,n)$。对于固定的 i，n 的范围是 $n \leq \frac{A_i}{M}$，所以要求和的项数是

$$\sum_{i=1}^{N-1} \frac{A_i}{M} \leq \sum_{d=1}^{N} \frac{d}{M} = O(M \log N) \text{（调和级数的和）}$$

因此，问题在总共 $O(M \log N)$ 时间内解决。

即使 A_i 有重复元素，也可以适当地一次性处理它们，以相同的复杂度找到答案。