

# 7.27题目分享

📅 7.27题目分享



**Problem - C - Codeforces**  
Codeforces. Programming competitions and contests, programming community



题意： 给一个完全由0,6,8,9组成的字符串，可以且必须进行一次操作： 将一个子串180° 翻转，问有多少种不同的结果

## ▼ 题解

若一个子串是中心对称的，那么翻转后与翻转前是一样的，除此之外所有子串翻转后都会和翻转前不同，只要一个区间 $[l, r]$ 的开头末尾是00或者88或者69或者96，那么这个翻转这个区间得到的结果和翻转 $[l + 1, r - 1]$ 得到的结果是一样的，因此只要减去这些没必要算的区间有多少即可，需要根据是否存在中心对称的子串决定答案是否加1

## ▼ code

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#define ll long long
using namespace std;
const int maxn = 2e6 + 10;
int T, n, cnt;
int num[10];
char s[maxn];
int main ()
{
    scanf("%d", &T);
    while (T--) {
        scanf("%s", s + 1);
        n = strlen(s + 1);
        num[0] = num[6] = num[8] = num[9] = 0;
        for (int i = 1; i <= n; ++i) ++num[s[i] - '0'];
        ll ans = 1ll * (1 + n) * n / 2;
        ans -= 1ll * num[6] * num[9];
        ans -= 1ll * num[0] * (num[0] - 1) / 2 + num[0];
        ans -= 1ll * num[8] * (num[8] - 1) / 2 + num[8];
        if (num[0] || num[8] || (num[6] && num[9])) ++ans;
        printf("%lld\n", ans);
    }
    return 0;
}
```



**Problem - 1852A - Codeforces**  
Codeforces. Programming competitions and contests, programming community




题意： 有无穷个数从1 开始到正无穷，进行 $k$  轮操作，有一个位置数组 $a_1, a_2, \dots, a_n$ ，每一次操作会同时删掉排名为 $a_1, a_2, \dots, a_n$  的数，问 $k$  轮后第一个位置的数是多少

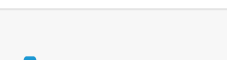
## ▼ 题解


考虑一轮后原本排名为 $x$  的数到哪去了，若有 $k$  个 $a$  小于 $x$ ，则说明在 $x$  前面有 $k$  个数被删掉了，则排名为 $x$  的数排名会变成会变成 $x - k$ ，现在反过来考虑，现在排名为 $x$  在上一轮的排名是多少，假设 $a$  数组为 $\{1, 2, 3\}$ ，现在排名为1，则上一轮可以很简单的想到是4，而如果 $a$  数组为 $\{1, 2, 4\}$ ，则上一轮排名是3，考虑相邻两个 $a_i$  和 $a_{i+1}$  中间有 $a_{i+1} - a_i - 1$  个位置没有被删掉，因此可以想到如果这些位置上的数会按照顺序成为新的排名，如果这些位置的个数大于等于 $x$ ，则可知 $x$  上一轮的位置在这些位置中，将这些空位加起来找到第一个大于等于 $x$  的下标 $i$ ，则 $x$  上一轮为 $x + i$ ， $x$  初值赋值为1然后如此做即可

## ▼ code

```
#include <cstdio>
#define ll long long
using namespace std;
const int maxn = 2e5 + 10;
struct IO {
    template <class T>
    IO operator>>(T &res) {
        res = 0;
        char ch;
        bool flag = false;
        while ((ch = getchar()) > '9' || ch < '0') flag |= ch == '-';
        while (ch >= '0' && ch <= '9') res = (res << 3) + (res << 1) + (ch ^ '0'), ch = getchar();
        if (flag) res = ~res + 1;
        return *this;
    }
} cin;
int T, n, k;
int a[maxn], s[maxn];
int main()
{
    cin >> T;
    while (T--) {
        cin >> n >> k;
        for (int i = 1; i <= n; ++i) cin >> a[i], s[i] = s[i - 1] + a[i] - a[i - 1] - 1;
        int cur = 1;
        ll x = 1;
        while (k--) {
            while (cur <= n && x > s[cur]) ++cur;
            x += cur - 1;
        }
        printf("%lld\n", x);
    }
    return 0;
}
```



**Problem - F - Codeforces**  
Codeforces. Programming competitions and contests, programming community




一个长度为 $n = 2^k \leq 2^{20}$ 的数列，每秒钟所有位置 $a_i$ 变为 $a_i \oplus a_{(i+1) \bmod n}$ ，问最短时间使整个数列都变成0，可以证明一定会全部变为0


## ▼ 题解


考虑若干轮后第一个位置的数变成了什么，第0 轮： $a_1$ ，第一轮： $a_1 \oplus a_2$ ，第二轮： $a_1 \oplus a_3$ ，第三轮： $a_1 \oplus a_2 \oplus a_3 \oplus a_4$ ，第四轮： $a_1 \oplus a_5 \dots$  第 $2^k$  轮： $a_1 \oplus a(1 + 2^k)$ ，由于全0后不会再变，并且一定存在全0，并且第 $2^k$ 轮后长啥样很容易可以求出来，所以可以倍增的整

## ▼ code

```
#include <cstdio>
using namespace std;
const int maxn = 2e6 + 5;
int n, ans;
int a[maxn];
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);
    while (n > 1) {
        int k = n >> 1;
        bool flag = false;
        for (int i = 0; i < k; ++i)
            if (a[i] != a[i + k]) {
                flag = true;
                break;
            }
        if (flag) {
            ans += k;
            for (int i = 0; i < k; ++i) a[i] ^= a[i + k];
        }
        n >>= 1;
    }
    if (a[0]) ans++;
    printf("%d\n", ans);
}
```



**Problem - J - Codeforces**  
Codeforces. Programming competitions and contests, programming community



题意： 给 $n$  个区间，需要在每个区间选一个数，使得将这些数与起来的结果最大， $n$  个区间相互独立（即可选择相同的数）

## ▼ 题解

从大到小考虑每一位是否能填1，同时构造出每个区间选的数是什么，设 $now_i$  表示第 $i$  个区间在满足之前贪心的条件下，目前选的数是什么，现在考虑到第 $k$  位了，假设第 $k$  位为1，则 $now_i$  会变成 $now_i | 2^k$ ，之后能表达的数在区间 $[now_i + 2^k, now_i + 2^{k+1} - 1]$ 范围内，如果这个区间和 $[l_i, r_i]$ 有交集，则说明第 $i$  个区间在满足前面位的情况下第 $k$  位可以为1，若所有区间第 $k$  位都可为1，那么所有的 $now_i = 2^k$ ，否则就要考虑 $now_i$  第 $k$  位是否为1，若本就不可为1 自不必说，而如果 $now_i$  第 $k$  位可以为1，也可不为1，我们可以考虑设之前说的区间为 $[nl, nr]$ ，现在我们其实并不关心第 $k$  位如何，为1 也好，不为1 也好，只要能让我们在考虑之后的某位能为1 时能尽可能满足条件即可，此时需要注意到，考虑 $nl - 1$  的二进制是什么样的， $nl - 1 = now_i + 2^k - 1$ ，也就是说， $nl - 1$  的第0 位到第 $k - 1$  位都是1，如果我们的区间 $[l_i, r_i]$  包含了 $nl - 1$ ，那么第 $k$  位不需要变成1，否则就只能为1了

## ▼ code

```
#include <cstdio>
using namespace std;
const int maxn = 1e5 + 10;
int T, n;
int l[maxn], r[maxn], now[maxn];
int main ()
{
    scanf("%d", &T);
    while (T--) {
        scanf("%d", &n);
        for (int i = 1; i <= n; ++i) scanf("%d%d", &l[i], &r[i]), now[i] = 0;
        int ans = 0;
        for (int i = 29; i >= 0; --i) {
            bool flag = true;
            for (int j = 1; j <= n && flag; ++j) {
                int nl = now[j] | (1 << i), nr = now[j] | ((2 << i) - 1);
                if (nl > r[j] || nr < l[j]) flag = false;
            }
            if (flag) {
                for (int j = 1; j <= n; ++j) now[j] |= 1 << i;
                ans |= 1 << i;
            }
            else {
                for (int j = 1; j <= n; ++j) {
                    int nl = now[j] | (1 << i), nr = now[j] | ((2 << i) - 1);
                    if (nl <= r[j] && nr >= l[j] && nl - 1 <= l[j]) now[j] |= 1 << i;
                }
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```