补题 int: 2147483647 = 2×10^9

牛客第一场 A

$a_1, a_2, \ldots, a_n$ 其中 a_i 值域 小于 2^m ,问有多少种子序列 方案数 使得 AND 值 为 1

首先选择 C_n^k 个 组成 AND 值为1 的 数， 他们有共同点 二进制下最后一位均是1， 还要保证 一共 $m-1$位情况下, 每位对应的一列 k个，不全为1， 共 2^k-1 种，$(0000-1110)$，再 $m-1$ 次幂。

未选择参与构成的元素 有以下特点： 最后一位必须为0， 其他位无要求， $2^{(m-1)\times(n-k)}$

k : $C_n^k \times (2^k-1)^{m-1} \times 2^{(m-1)\times(n-k)}$

美剧k 累加即可 建议列表发现组合数学规律

```cpp
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    int n,m,p;
    std::cin >> n >> m >> p;
    auto ksm = [&](ll a, ll b,ll p) {
        ll res = 1;
        while(b) {
            if(b & 1) res = res * a % p;
            a = a * a % p;
            b >>= 1;
        }
        return res;
    };
    std::vector<std::array<ll,5002>> c(n+1);
    for(int i = 0; i <= n; i++) c[i][0] = 1;
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= i; j++) {
            c[i][j] = c[i-1][j] + c[i-1][j-1];
            c[i][j] %= p;
        }
    }
    ll ans = 0;
    for(int k = 1; k <= n; k++) {
        ans = ans + c[n][k] * ksm(2,(m-1)*(n-k),p) %p * ksm( ksm(2,k-1,p) , m-1 , p) % p;
        ans = (ans + p) % p;
    }
    std::cout << ans << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
```

```
}
```

$div3D$

$n, x \le 10^6$ $a \times b + b \times c + c \times a \le n$ $a + b + c \le x$ 求正整数三元组a, b, c数量

就是美剧

```cpp
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e6 + 50;

void solve()
{
    int n,x;
    std::cin >> n >> x;
    ll ans = 0;
    for(int a = 1; a <= n; a ++) {
        for(int b = 1; a * b <= n && a + b <= x; b ++)
        ans += std::min((n - a * b) / (a + b), x - (a + b));
    }
    std::cout << ans << '\n';
    return;
}
int main()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _ ;
    while(_ --) {
        solve();
    }
    return 0;
}

```

$div3C$

A, B 两串， 看最少修改几次 能够使给定区间 $[l, r]$ 内元素 排序后相同

明显上下相同 不用改

特殊数据 $aaaa/bbbb$

美剧每个字符 在该区间内的上下差距数， 累加

```cpp
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e6 + 50;

void solve()
{
    int n,q;
    std::cin >> n >> q;
    std::vector<std::array<int,26>> pre(n+1);
    std::string a,b;
    std::cin >> a >> b;
```

```
    for(int i = 1; i <= n; i ++) {
        pre[i] = pre[i-1];
        pre[i][a[i-1] - 'a'] ++;
        pre[i][b[i-1] - 'a'] --;
    }

    while(q --) {
        int l,r;
        ll ans = 0;
        std::cin >> l >> r;
        for(int c = 0; c < 26; c ++) {
            ans += std::max(0,pre[r][c] - pre[l-1][c]);
        }
        std::cout << ans << '\n';
    }

    return;
}
int main()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _ ;
    while(_ --) {
        solve();
    }
    return 0;
}

```

$hdu3$ 1007

区间加， 区间判断 相等， 单调递增， 单调递减， 单峰数列

哥哥用的方法是差分， 将 每个位置i 的差分正负作为加入 set 的依据

若 正 和 零 的集合内没有位置 说明单调递减

类比 注意 单峰数列 首先不能平， 还要保证严格单增后单减

虽然这题很难碰到类似的 但是学习到了内置函数的设置方法

```
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    int n;
    std::cin >> n;
    std::vector<ll> a(n),d(n);
    for(int i = 0; i < n; i++) {
        std::cin >> a[i];
    }
    for(int i = 1; i < n; i++) {
        d[i] = a[i] - a[i-1];
    }
    std::set<int> Z,P,N;
    auto work = [&](int i) {
        Z.erase(i);
```

```
        P.erase(i);
        N.erase(i);
        if(d[i] > 0) P.insert(i);// +
        else if(d[i] < 0) N.insert(i);// -
        else Z.insert(i);// 0
    };
    for(int i = 1; i < n; i++) work(i);

    auto query = [&](int o,int l,int r) {
        if(o == 2) {
            return P.upper_bound(l) == P.lower_bound(r) && N.upper_bound(l) == N.lower_bound(r);
        } else if(o == 3) {
            return Z.upper_bound(l) == Z.lower_bound(r) && N.upper_bound(l) == N.lower_bound(r);
        } else if(o == 4) {
            return P.upper_bound(l) == P.lower_bound(r) && Z.upper_bound(l) == Z.lower_bound(r);
        } else if(o == 5) {
            if(Z.upper_bound(l) != Z.lower_bound(r))
                return false;
            auto pl = P.upper_bound(l);
            auto pr = P.lower_bound(r);
            auto nl = N.upper_bound(l);
            auto nr = N.lower_bound(r);
            if(pl == pr || nl == nr) return false;

            return *std::prev(pr) < *nl;
        }
    };
    int q;
    std::cin >> q;
    while(q --) {
        int o,l,r;
        std::cin >> o >> l >> r;
        l --;
        if(o == 1) {
            int x;
            std::cin >> x;
            if(l > 0) {
                d[l] += x;
                work(l);
            }
            if(r < n) {
                d[r]  -= x;
                work(r);
            }
        }
        else {
            std::cout << query(o,l,r) << '\n';
        }
    }
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
```

```
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

$abcD$

x 轴上 有 n 个点， 求给定点$b[i]$ 在该轴上距离第$k[i]$ 近的点 与 $b[i]$ 的(最近)距离

二分答案 距离 看看该点 向左右延伸的距离能否包含 $k[i]$个 点

```
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e6 + 50;

void solve()
{
    int n,q;
    std::cin >> n >> q;
    std::vector<int> a(n), b(q),k(q);
    for(int i = 0; i < n; i ++) std::cin >> a[i];
    std::sort(a.begin(), a.end());
    for(int i = 0; i < q; i ++) {
        std::cin >> b[i] >> k[i];
        int l = 0, r = 2e8, x = 0;
        while(l <= r) {
            int mid = (l + r) >> 1;
            int ml = b[i] - mid, mr = b[i] + mid;
            int cnt = std::upper_bound(a.begin(), a.end(), mr) -
std::lower_bound(a.begin(), a.end(), ml);
            if(cnt >= k[i]) {
                x = mid;
                r = mid - 1;
            }
            else l = mid + 1;
        }
        std::cout << x << '\n';
    }
    return;
}
int main()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _ ;
    while(_ --) {
        solve();
    }
    return 0;
}

```

dp + $bitset$

一共有n个数，第i个数是x_i, x_i 可以取 $[l_i, r_i]$ 中任意的一个值。设 $S = \sum {x_i}^2$，求 S 种类数。

```
bitset<N> f//相当于bool f[N]
f.any()//整个二进制中是否有一个位置被置为1
f.none()//是否整个f中所有位上都是0
f.count()//二进制中1的个数
f.set()//将二进制各个位置全都置为1
f.set(i)//相当于f[i]=1
f.reset()//将二进制各个位置全都置为0
f.reset(i)//相当于f[i]=0
f.flip()//将二进制所有位取反
f.flip(i)//相当于f[i]^=1
```

通过STL中的$bitset$去优化。
比如 11001 表示 1 4 5 可以用加法得到，因为 位置1 4 5上的数为1
总结一下，$bitset$的第X位为1表示整数X可以通过加法得到。

$ans[i][j]$ 表示 第i行 能不能出现 j 这个值 存的是长度为 N 的01串

```
#include <bits/stdc++.h>
typedef long long ll;
const int N = 1e6 + 50;
void solve() {
    int n;
    std::cin >> n;
    std::bitset<N> ans[101];
    ans[0][0] = 1;
    for(int i = 1,l,r; i <= n; i++) {
        std::cin >> l >> r;
        for(int j = l; j <= r; j++) {
            ans[i] |= ans[i-1] << j * j;
        }
    }
    std::cout << ans[n].count() << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

D

小红希望出一场题目，但是他的实力又不够，所以他想到可以从以前的比赛中各抽一题，来组成一场比赛。不过一场比赛的难度应该是有限制的，所以所以这一场比赛会给一个目标难度分数 target 。

小红选 n 场比赛，每场 m 个题，小红会从每一场选一道题，使其组成的题的难度分数尽量接

近 target 。小红想知道挑选的题的难度分数与 target 相差的最小值是多少。

```cpp
#include <bits/stdc++.h>
typedef long long ll;
const int N = 1e6 + 50;
int a[105][50];
void solve() {
    int n,m;
    std::cin >> n >> m;
    std::bitset<N> ans[101];
    ans[0][0] = 1;
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++)
            std::cin >> a[i][j];
    }
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) {
            ans[i] |= ans[i-1] << a[i][j];
        }
    }
    int tar , res = N;
    std::cin >> tar;
    for(int i = 0; i <= 10000; i++) {
        if(ans[n][i]) {
            res = std::min(res,std::abs(tar - i));
        }
    }
    std::cout << res << '\n';

    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

E

已知长度为 n 的序列 $a_1, a_2, \ldots, a_n$，定义一次操作的过程为：选择任意一个元素，随后，将 $\left\lfloor \frac{a_i}{2} \right\rfloor$（向下取整）添加到原序列的结尾，并将 a_i 从原序列中删除。

你可以进行任意多次操作（也可以一次操作都不做），要求使得序列的 MEX 最大。

数组的 MEX 定义为：没有出现在数组中的最小非负整数，例如，数组 $\{3, 1, 2\}$ 的 MEX 为 0。

二分最后能达成的值 $O(nlogn)$

```cpp
#include <bits/stdc++.h>
typedef long long ll;

```

```cpp
void solve() {
    int n;
    std::cin >> n;
    std::vector<int> a(n),b(n);
    for(int i = 0; i < n; i++) std::cin >> a[i];
    auto check = [&] (int t) {
        b = a;
        std::set<int> s;
        for(int i = 0; i < n; i++) {
            while(b[i] > t) b[i] /= 2;
            while(s.count(b[i]) && b[i] > 0) b[i] /= 2;
            s.insert(b[i]);
        }
        return (s.size() == (t + 1));
    };
    int l = 0, r = n + 1, x = 0;
    while(l <= r) {
        int mid = (l + r) >> 1;
        if(check(mid)) {
            l = mid + 1;
            x = mid;
        }
        else r = mid - 1;
    }
    std::cout << x + 1 << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

$1992E$

$num = n \times a - b$

$i = len \times a - b$

$a \leq 10000 \quad len \leq 2$ 因此i是个位数 枚举 注意$n = 1$

```cpp
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    int n;
    std::cin >> n;

    if(n == 1) {
        std::cout << 9999 << '\n';
        for(int i = 2; i <= 10000; i++) {
            std::cout << i << " " << i - 1<< '\n';
```

```
        }
        return;
    }
    std::string s = std::to_string(n),o = s;
    for(int i = 1; i <= 7; i++) s = s + o; // 10101010101010
    std::set<std::pair<int,int>> se;
    for(int i = 1; i <= 7; i++) {
        std::string c = s.substr(0,i);// 10101
        int num = 0;
        for(int j = 0; j < i; j++) {
            num = num * 10 + (c[j] - '0');
        }
        int p = n - o.length();
        int q = num - i;
        if(p == 0) continue;
        if(q % p == 0 && p != 0){
            int A = q / p;
            int B = n * A - num;
            if(A != 0 && B != 0 && A <= 10000) se.insert({A,B});
        }
    }
    std::cout << se.size() << '\n';
    for(auto [a,b] : se) {
        std::cout << a << " " << b << '\n';
    }
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

河南$3E$

在一个长度为n的纸带上，初始时所有位置颜色为白色,现在要执行以下两种操作一共q次

操作一：输入一个下标x,你需要将位置x的颜色翻转(白色变为黑色，黑色变为白色)

操作二; 输入两个正整数L, R,你需要输出区间$[L, R]$中的连续的白色区间长度最大值

```
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e6 + 50;
struct node{
    int l,r,v;
    int lmx,rmx,tmx; // 左 右 总
}t[N];

void pushup(node &u,node &l,node &r) {
    u.lmx = std::max(l.lmx, l.lmx == l.r - l.l + 1?l.lmx + r.lmx:l.lmx);
    u.rmx = std::max(r.rmx, r.rmx == r.r - r.l + 1?r.rmx + l.rmx:r.rmx);
```

```
    u.tmx = std::max({l.tmx,r.tmx,l.rmx + r.lmx});
}

void push_up(int u) {
    pushup(t[u],t[u<<1],t[u<<1|1]);
}
void build(int u,int l,int r) {
    t[u].l = l; t[u].r = r;
    if(l == r) {
        t[u] = {l,r,0,1,1,1};
        return;
    }
    int mid = (l + r) >> 1;
    build(u << 1,l,mid);
    build(u << 1 | 1,mid + 1,r);
    push_up(u);
}

void modify(int u,int x) {
    int l = t[u].l, r = t[u].r;
    if(l == x && r == x) {
        int pv = t[u].v ^ 1;
        if(pv == 0) t[u] = {l,r,pv,1,1,1};
        else t[u] = {l,r,pv,0,0,0};
        return;
    }
    int mid = l + r >> 1;
    if(x <= mid) modify(u << 1,x);
    if(x > mid) modify(u << 1 | 1,x);
    push_up(u);
}
node query(int u,int l,int r) {
    if(l <= t[u].l && t[u].r <= r) return t[u];
    int mid = t[u].l + t[u].r >> 1;
    if(r <= mid) return query(u << 1,l,r);
    else if(l > mid) return query(u << 1 | 1,l,r);
    else {
        node p,pl,pr;
        pl = query(u << 1,l,r);
        pr = query(u << 1 | 1,l,r);
        pushup(p,pl,pr);
        return p;
    }
}
void solve() {
    int n,q;
    std::cin >> n >> q;
    build(1,1,n);
    for(int i = 1,op,x,y; i <= q; i++) {
        std::cin >> op;
        if(op == 1) {
            std::cin >> x;
            modify(1,x);
        }
        else {
            std::cin >> x >> y;
            std::cout << query(1,x,y).tmx << '\n';
        }
```

```
    }
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

牛客$2E$

$gcd(x,y)=x\oplus y$ 给x，求 任意$y,y<x$

....1000 若后面都是0，表示能整除2^x，若y的1前面与其相同，异或后消失，能保证

....0000 能满足要求，只要x 减掉第一个一以及后面出现的所有即可，$lowbit(x)$ 返回第一个1代表的二的幂次

```
#include <bits/stdc++.h>
typedef long long ll;
void solve() {
    ll x;
    std::cin >> x;
    auto lowbit = [&] (ll i) {
      return i & (-i);
    };
    if(x - lowbit(x) > 0) {
        std::cout << x - lowbit(x) << '\n';
    }
    else std::cout << -1 << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

牛客$2C$

红色在一个 $2\cdot n$ 的网格上，某些单元格是红色的，其他单元格是白色的。

红色可以最初选择一个红色单元格，并且在每一步中，可以选择上方、下方、左侧或右侧的红色单元格。当红色离开一个单元格时，那个单元格会立刻变成白色。

红色想知道她可以走的最大步数。

如果没有初始红色单元格，请输出 0。

哥哥的从左到右正向dp,

```cpp
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    int n,ans = 0;
    std::cin >> n;
    std::string s[2];
    std::cin >> s[0] >> s[1];
    int dp[2] = {};
    for(int i = 0; i < n; i++) {
        if(s[0][i] == 'R') dp[0]++;
        else dp[0] = 0;
        if(s[1][i] == 'R') dp[1]++;
        else dp[1] = 0;
        if(s[0][i] == 'R' && s[1][i] == 'R') {
            int p = dp[0], q = dp[1];// 重复使用 防止更改
            dp[0] = std::max(p,q + 1);
            dp[1] = std::max(q,p + 1);//
        }
        ans = std::max({ans,dp[0],dp[1]});
    }
    std::cout << std::max(0,ans - 1);
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

牛客$2H$

更难的一道 dp

```cpp
#include <bits/stdc++.h>
typedef long long ll;
// 题目要求经过 并非停留
void solve() {
    int n,x,y;
    std::cin >> n >> x >> y;
    std::vector<std::array<int,2>> a(n+1);
    a[0] = {0,0};
    std::string s;
    std::cin >> s;

    for(int i = 0; i < n; i++) {
        a[i+1] = a[i];
```

```
        if(s[i] == 'W') a[i+1][1] ++;
        else if(s[i] == 'S') a[i+1][1]--;
        else if(s[i] == 'A') a[i+1][0]--;
        else a[i+1][0]++;
    }
    ll ans = 0;
    for(int i = 0; i <= n; i++) {
        std::cout << a[i][0] << " " << a[i][1] << '\n'; // 前缀位置
    }
    std::map<std::array<int,2>,int> mp;
    for(int i = n; i >= 0; i--) {
        mp[{a[i][0],a[i][1]}] = i;
        if(mp.count({a[i][0] + x, a[i][1] + y})) {
            int j = mp[{a[i][0] + x, a[i][1] + y}]; // j = i, x y = 0
            j = std::max(j,i+1);//更新索引j，确保它是当前索引i和满足条件的索引j中的较大值。
            //std::cout <<i << ": " <<  j << "  ";
            ans += n - j + 1;
        }
    }
    std::cout <<'\n' <<  ans << '\n';
    return;
}
/*
0 0   x,y: 1,1  // -2,-1 + 1,1 = -1,0(r = 5) -> 3-5 3-6
-1 0 (i = 1)
-1 -1
-2 -1
-2 0
-1 0
0 0
3: 5  2: 6
3
*/
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

河南$3G$

现有三个正整数A,B,C，请你找三个整数x,y,z满足$x+y+z=n$，且$0\le x,y,z$，使得$|x*A+y*B+z*C-W|$最小。

为了简化题目，现在只要找到$|x*A+y*B+z*C-W|$的最小值。

多组数据T，n的和小于10^6

如果需要求出单峰函数的极值点，通常使用二分法衍生出的三分法求单峰函数的极值点。

枚举x, $y+z=n-x$，保留y或x，$f(x)=x*A+y*B+(n-x-y)*C-w$

由于绝对值， 先递减后递增， 符合单峰函数， 时间复杂度$O(nlogn)$

```
while (r - l > eps) {  // eps 看题目 整数三分
  mid = (l + r) / 2;
  lmid = mid - eps;
  rmid = mid + eps;
  if (f(lmid) < f(rmid))
    r = mid;
  else
    l = mid;
}
```

```
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    ll A,B,C,n,W;
    std::cin >> A >> B >> C >> n >> W;
    ll ans = 1e18;
    auto check = [&] (int x,int y) {
        int z = n - x - y;
        return std::abs(A * x + B * y + C * z - W);
    };
    for(int x = 0; x <= n; x ++) {
        int l = 0, r = n - x;
        while(r - l > 1) {
            int mid = l + r >> 1;
            int lmid = mid - 1;
            int rmid = mid + 1;
            if(check(x,lmid) < check(x,rmid)) r = mid;
            else l = mid;
        }
        ans = std::min({ans,check(x,l),check(x,r)});
    }
    std::cout << ans << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

实数三分

```
double eps = 1e-9;
    while(std::fabs(r - l) > eps) {
        double m1 = (l * 2 + r) / 3;
        double m2 = (l + r * 2) / 3;
        if(check(m1) < check(m2)) r = m2;
        else l = m1;
    }
```

牛客2B

求树 的 部分最小生成树

由于数据多 用点度相对大小存边

每次询问的用时间戳标识， 表示不同批次

```
#include <bits/stdc++.h>
typedef long long ll;
const int N = 2e5 + 50;
struct E {
    int u,v,w;
    bool operator < (const E & x) const {
        return w < x.w;
    }
};
E e[N];
std::vector<std::pair<int,int>> G[N];
int d[N], tim = 0;
int fa[N],tag[N];
int find(int x) {
    return fa[x] == x ? x : fa[x] = find(fa[x]);
}
void solve() {
    int k;
    std::cin >> k;
    tim ++;
    std::vector<int> a(k+1);

    for(int i = 1; i <= k; i++) {
        std::cin >> a[i];
        tag[a[i]] = tim;
        fa[a[i]] = a[i];
    }

    if(k == 1) {
        std::cout << 0 << '\n';
        return;
    }
    std::vector<E> ve;
    for(int i = 1; i <= k; i++) {
        int x = a[i];
        for(auto [v,w] : G[x]) {
            if(tag[v] == tim) ve.push_back({x,v,w});
        }
    }

    std::sort(ve.begin(),ve.end());
```

```
    ll ans = 0, cnt = 0;
    for(auto [u,v,w] : ve) {
        u = find(u), v = find(v);
        if(u == v) continue;
        fa[v] = u;
        ans += w; cnt += 1;
        if(cnt == k - 1) {
            std::cout << ans << '\n';
            return;
        }
    }

    std::cout <<  -1 << '\n';
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int  n,m,q;
    std::cin >> n >> m >> q;
    for(int i = 1; i <= m; i++) {
        std::cin >> e[i].u >> e[i].v >> e[i].w;
        d[e[i].u]++;
        d[e[i].v]++;
    }
    for(int i = 1; i <= m; i++) {
        int u = e[i].u, v = e[i].v;
        if(d[u] < d[v] || d[u] == d[v] && u < v) G[u].push_back({v,e[i].w});
        else G[v].push_back({u,e[i].w});
    }

    while(q --) {
        solve();
    }
    return 0;
}
```

牛客3 B

给定n次操作， 每次可将x 修改为 $|x - a_i|$

求从D 开始可以得到的最小值。

$n \leq 100, 1 \leq a_i, D \leq 10^{18}$

```
#include <bits/stdc++.h>
typedef long long ll;
ll gcd(ll a,ll b) {
    if(b == 0) return a;
    return gcd(b,a%b);
}

void solve() {
    ll n,D;
```

```
    std::cin >> n >> D;
    std::vector<ll> h(n);
    ll g = 0;
    for(int i = 0; i < n; i++) {
        std::cin >> h[i];
        g = gcd(h[i],g);
    }
    std::cout << std::min(D % g, g - D % g) << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

牛客3 A

一群n步行者在夜晚来到河边。他们想用一艘船渡河，这艘船最初是在他们这边的。这艘船一次最多能容纳R名步行者，至少需要L $(1 \leq L < R)$名步行者来操作。

划船是一项累人的工作。每次用船运送一群步行者到河的另一边，船上所有步行者的耐力必须大于0，每个步行者的耐力在旅程结束后会减少1。最初，i步行者$(1 \leq i \leq n)$的耐力值为h_i。

你需要确定它是否可以运输。

需要从河的右侧往回送运趟数最小值：$s = \left\lceil \frac{n-R}{R-L} \right\rceil$

令$a_i = \left\lfloor \frac{h_i - 1}{2} \right\rfloor$ 为第i个人多来回的趟数

必要条件：$\sum_{i=1}^{n} min(a_i, s) \geq sL$

贪心： 从右侧往回运相当于每次将a_1, a_2, a_i, a_n 最大的 L 个元素减一

```
#include <bits/stdc++.h>
typedef long long ll;

void solve() {
    int n,L,R;
    int num = 0, ans = 0;
    std::cin >> n >> L >> R;
    int s = (n - R - 1) / (R - L) + 1;
    std::vector<int> h(n),a(n);
    for(int i = 0; i < n; i ++) {
        std::cin >> h[i];
        if(h[i] > 2) {
            a[i] = (h[i] - 1) / 2;
            num ++;
        }
```

```cpp
    }
    if(R >= n) {
        std::cout << "Yes" << '\n';
        return;
    }
    else {
        if(num < L) std::cout << "No" << '\n';
        else {
            for(int i = 0; i < n; i++) {
                ans += std::min(a[i],s);
            }
            if(ans  >= s * L) {
                std::cout << "Yes" << '\n';
            }
            else std::cout << "No" << '\n';
        }
    }
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

题目描述

MarkDown视图 Copy 翻译

小红是一名医生。

现在小红对于每个病人的症状用一个长度为m的01串表示，第i个字符代表第i个身体部位的症状，0代表健康，1代表不健康。
一共有k种药，每种药也用一个长度为m的01串表示，第i个字符为'1'代表该药可以治愈第i个部位的症状。
对于每个病人，请你帮小红求出治愈该病人需要开的最少的药数量。

输入描述:

MarkDown视图 Copy 翻译

第一行输入两个正整数n,m，代表病人数量和症状种类数。
接下来的n行，每行输入一个长度为m的01串，代表每个病人的症状情况。
接下来一行输入一个正整数k，代表药物的数量。
接下来的k行，每行输入一个长度为m的01串，代表每个药物可以治愈的症状情况。
$1 \le m \le 20$
$1 \le n \le 10^4$
$1 \le k \le 10$

```cpp
// Problem: 医生
// Contest: NowCoder
// URL: https://ac.nowcoder.com/acm/contest/92972/D
```

```cpp
// Memory Limit: 524288 MB
// Time Limit: 2000 ms
//
// Powered by CP Editor (https://cpeditor.org)

#include <bits/stdc++.h>
typedef long long ll;
typedef unsigned long long ull;
using i64 = long long;
const int N = 2e3 + 4;
const int mod = 1e9 + 7;
void solve() {
    int n,m;
    std::cin >> n >> m;
    auto fsum = [&] (std::string str) -> int {
        int sum = 0;
        int len = str.length();
        for(int i = 0; i < len; i++) {
            sum = sum * 2 + (str[i] - '0');
        }
        return sum;
    };
    std::vector<int> p(n);
    std::vector<int> v;
    std::vector<bool> vis(1<<m,false);
    for(int i = 0; i < n; i++) {
        std::string s;
        std::cin >> s;
        p[i] = fsum(s);
        if(vis[p[i]] == false) {
            v.push_back(p[i]);
            vis[p[i]] = true;
        }
    }
    int k;
    std::cin >> k;
    std::vector<int> d(k);
    for(int i = 0; i < k; i++) {
        std::string s;
        std::cin >> s;
        d[i] = fsum(s);
    }
    std::vector<std::pair<int,int> > pa;
    for(int t = 1; t < (1 << k) ; t++) { // 药的所有混合方案
        int A = 0, B = 0;
        for(int i = 0;i < k ;i++) { // 解构t的每一位
            if(t & (1 << i)) {
                A |= d[i];  // 自己有的药 混合
                B ++;
            }
        }
        pa.push_back({B,A});
    }
    std::sort(pa.begin(),pa.end()); // 按照 混合的数量排序
    std::vector<int> ans((1<<m),-1);
    for(auto c : v) {
        if(c == 0) { // 无病 00000
            ans[c] = 0;
```

```
            continue;
        }
        for(auto [b,a] : pa) {
            if( (a & c)  ==  c) { // 枚举所有混合药  能治
                ans[c] = b;
                break;
            }
        }
    }
    for(auto c : p) {
        std::cout << ans[c] << '\n';
    }
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```

[E - Mod Sigma Problem]

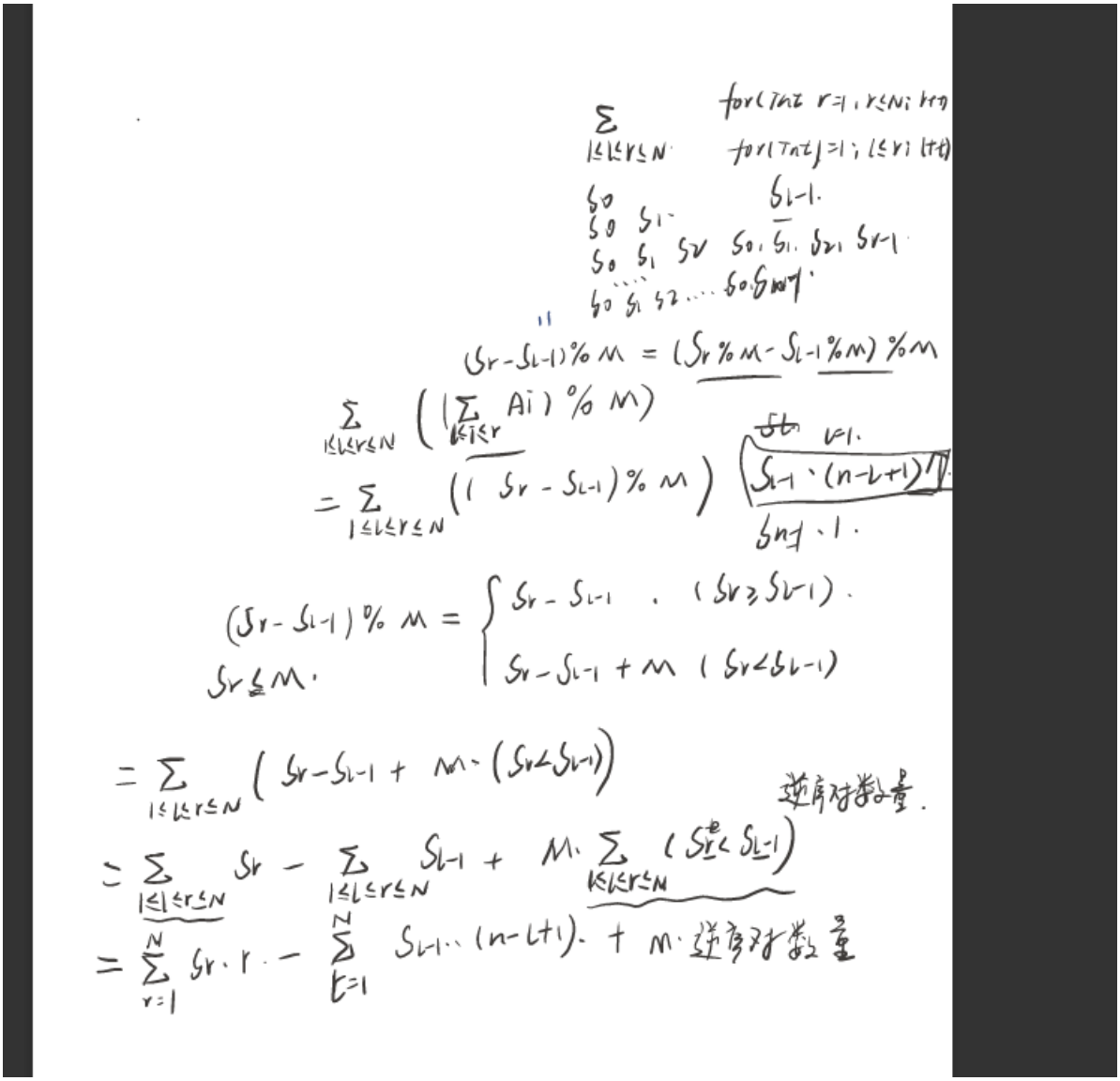
问题陈述

给定一个由 N 组成的非负整数序列 $A = (A_1, A_2, \ldots, A_N)$ 和一个正整数 M 。

查找以下值：

$$\sum_{1 \leq l \leq r \leq N} \left(\left(\sum_{l \leq i \leq r} A_i \right) \bmod M \right).$$

其中， $X \bmod M$ 表示非负整数 X 除以 M 时的余数。



```cpp
// Problem: E - Mod Sigma Problem
// Contest: AtCoder - AtCoder Beginner Contest 378
// URL: https://atcoder.jp/contests/abc378/tasks/abc378_e
// Memory Limit: 1024 MB
// Time Limit: 2000 ms
//
// Powered by CP Editor (https://cpeditor.org)

#include <bits/stdc++.h>
typedef long long ll;
typedef unsigned long long ull;
using i64 = long long;
const int N = 2e5 + 4;
const int mod = 1e9 + 7;
int b[N];
i64 ans = 0;
i64 msort(std::vector<i64> &a,int l,int r)
{
    if(l >= r) return 0;
    int mid = (l + r) / 2;
    msort(a,l,mid);
    msort(a,mid+1,r);
    int i = l, j = mid + 1,k = l;
```

```cpp
    while(i <= mid && j <= r)
    {
        if(a[i] <= a[j])  b[k++] = a[i++];
        else {
            b[k++] = a[j++];
            ans += (mid - i + 1);
        }
    }
    while(i <= mid) b[k++] = a[i++];
    while(j <= r) b[k++] = a[j++];
    for(int i = l; i <= r; i++)  a[i] = b[i];
    return ans;
}

void solve() {
    int n,m;
    std::cin >> n >> m;
    std::vector<i64> a(n+1),s(n+1),t(n+1);

    for(int i = 1; i <= n; i++) {
        std::cin >> a[i];
        s[i] = s[i-1] + a[i];
        s[i] %= m;
    }
    std::copy(s.begin(),s.end(),t.begin());
    i64 res = msort(t,1,n);
    i64 Ans = 0;
    for(int i = 1; i <= n; i++) {
        Ans += s[i] * i;
    }
    for(int i = 1; i <= n; i++) {
        Ans -= (s[i-1] * (n - i + 1));
    }
    Ans += m * res;
    std::cout << Ans << '\n';
    return;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int _ = 1;
    //std::cin >> _;
    while(_ --) {
        solve();
    }
    return 0;
}
```