

补题

int: $2147483647 = 2 \times 10^9$

牛客第一场 A

 a_1, a_2, \dots, 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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3
4  void solve() {
5      int n,m,p;
6      std::cin >> n >> m >> p;
7      auto ksm = [&](ll a, ll b, ll p) {
8          ll res = 1;
9          while(b) {
10             if(b & 1) res = res * a % p;
11             a = a * a % p;
12             b >>= 1;
13         }
14         return res;
15     };
16     std::vector<std::array<ll,5002>> c(n+1);
17     for(int i = 0; i <= n; i++) c[i][0] = 1;
18     for(int i = 1; i <= n; i++) {
19         for(int j = 1; j <= i; j++) {
20             c[i][j] = c[i-1][j] + c[i-1][j-1];
21             c[i][j] %= p;
22         }
23     }
24     ll ans = 0;
25     for(int k = 1; k <= n; k++) {
26         ans = ans + c[n][k] * ksm(2, (m-1)*(n-k), p) % p * ksm(ksm(2, k-1, p),
m-1, p) % p;
27         ans = (ans + p) % p;
28     }
29     std::cout << ans << '\n';
30     return;
31 }
32
33 int main() {
34     std::ios::sync_with_stdio(false);
35     std::cin.tie(nullptr);
36     int _ = 1;
37     //std::cin >> _;
38     while(_ --) {
39         solve();
40     }
41     return 0;

```

div3D

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 2e6 + 50;
4
5  void solve()
6  {
7      int n,x;
8      std::cin >> n >> x;
9      ll ans = 0;
10     for(int a = 1; a <= n; a++) {
11         for(int b = 1; a * b <= n && a + b <= x; b++)
12             ans += std::min((n - a * b) / (a + b), x - (a + b));
13     }
14     std::cout << ans << '\n';
15     return;
16 }
17 int main()
18 {
19     std::ios::sync_with_stdio(false);
20     std::cin.tie(nullptr);
21     int _ = 1;
22     std::cin >> _ ;
23     while(_ --) {
24         solve();
25     }
26     return 0;
27 }
28

```

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 2e6 + 50;
4
5  void solve()
6  {
7      int n,q;
8      std::cin >> n >> q;
9      std::vector<std::array<int,26>> pre(n+1);
10     std::string a,b;
11     std::cin >> a >> b;

```

```

12     for(int i = 1; i <= n; i++) {
13         pre[i] = pre[i-1];
14         pre[i][a[i-1] - 'a']++;
15         pre[i][b[i-1] - 'a']--;
16     }
17
18     while(q--) {
19         int l, r;
20         ll ans = 0;
21         std::cin >> l >> r;
22         for(int c = 0; c < 26; c++) {
23             ans += std::max(0, pre[r][c] - pre[l-1][c]);
24         }
25         std::cout << ans << '\n';
26     }
27
28     return;
29 }
30 int main()
31 {
32     std::ios::sync_with_stdio(false);
33     std::cin.tie(nullptr);
34     int _ = 1;
35     std::cin >> _;
36     while(_--) {
37         solve();
38     }
39     return 0;
40 }
41

```

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3
4  void solve() {
5      int n;
6      std::cin >> n;
7      std::vector<ll> a(n), d(n);
8      for(int i = 0; i < n; i++) {
9          std::cin >> a[i];
10     }
11     for(int i = 1; i < n; i++) {
12         d[i] = a[i] - a[i-1];
13     }
14     std::set<int> Z, P, N;
15     auto work = [&](int i) {
16         Z.erase(i);

```

```

17     P.erase(i);
18     N.erase(i);
19     if(d[i] > 0) P.insert(i); // +
20     else if(d[i] < 0) N.insert(i); // -
21     else Z.insert(i); // 0
22 };
23 for(int i = 1; i < n; i++) work(i);
24
25 auto query = [&](int o,int l,int r) {
26     if(o == 2) {
27         return P.upper_bound(l) == P.lower_bound(r) && N.upper_bound(l)
== N.lower_bound(r);
28     } else if(o == 3) {
29         return Z.upper_bound(l) == Z.lower_bound(r) && N.upper_bound(l)
== N.lower_bound(r);
30     } else if(o == 4) {
31         return P.upper_bound(l) == P.lower_bound(r) && Z.upper_bound(l)
== Z.lower_bound(r);
32     } else if(o == 5) {
33         if(Z.upper_bound(l) != Z.lower_bound(r))
34             return false;
35         auto pl = P.upper_bound(l);
36         auto pr = P.lower_bound(r);
37         auto nl = N.upper_bound(l);
38         auto nr = N.lower_bound(r);
39         if(pl == pr || nl == nr) return false;
40
41         return *std::prev(pr) < *nl;
42     }
43 };
44 int q;
45 std::cin >> q;
46 while(q --) {
47     int o,l,r;
48     std::cin >> o >> l >> r;
49     l --;
50     if(o == 1) {
51         int x;
52         std::cin >> x;
53         if(l > 0) {
54             d[l] += x;
55             work(l);
56         }
57         if(r < n) {
58             d[r] -= x;
59             work(r);
60         }
61     }
62     else {
63         std::cout << query(o,l,r) << '\n';
64     }
65 }
66 return;
67 }
68
69 int main() {
70     std::ios::sync_with_stdio(false);
71     std::cin.tie(nullptr);

```

```

72     int _ = 1;
73     //std::cin >> _;
74     while(_ --) {
75         solve();
76     }
77     return 0;
78 }

```

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 2e6 + 50;
4
5  void solve()
6  {
7      int n,q;
8      std::cin >> n >> q;
9      std::vector<int> a(n), b(q),k(q);
10     for(int i = 0; i < n; i ++) std::cin >> a[i];
11     std::sort(a.begin(), a.end());
12     for(int i = 0; i < q; i ++) {
13         std::cin >> b[i] >> k[i];
14         int l = 0, r = 2e8, x = 0;
15         while(l <= r) {
16             int mid = (l + r) >> 1;
17             int ml = b[i] - mid, mr = b[i] + mid;
18             int cnt = std::upper_bound(a.begin(), a.end(), mr) -
std::lower_bound(a.begin(), a.end(), ml);
19             if(cnt >= k[i]) {
20                 x = mid;
21                 r = mid - 1;
22             }
23             else l = mid + 1;
24         }
25         std::cout << x << '\n';
26     }
27     return;
28 }
29 int main()
30 {
31     std::ios::sync_with_stdio(false);
32     std::cin.tie(nullptr);
33     int _ = 1;
34     //std::cin >> _ ;
35     while(_ --) {
36         solve();
37     }
38     return 0;
39 }
40

```

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。

```
1  bitset<N> f//相当于bool f[N]
2  f.any()//整个二进制中是否有一个位置被置为1
3  f.none()//是否整个f中所有位上都是0
4  f.count()//二进制中1的个数
5  f.set()//将二进制各个位置全都置为1
6  f.set(i)//相当于f[i]=1
7  f.reset()//将二进制各个位置全都置为0
8  f.reset(i)//相当于f[i]=0
9  f.flip()//将二进制所有位取反
10 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

```
1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 1e6 + 50;
4  void solve() {
5      int n;
6      std::cin >> n;
7      std::bitset<N> ans[101];
8      ans[0][0] = 1;
9      for(int i = 1, l, r; i <= n; i++) {
10         std::cin >> l >> r;
11         for(int j = l; j <= r; j++) {
12             ans[i] |= ans[i-1] << j * j;
13         }
14     }
15     std::cout << ans[n].count() << '\n';
16     return;
17 }
18
19 int main() {
20     std::ios::sync_with_stdio(false);
21     std::cin.tie(nullptr);
22     int _ = 1;
23     //std::cin >> _;
24     while(_ --) {
25         solve();
26     }
27     return 0;
28 }
```

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。

```
1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 1e6 + 50;
4  int a[105][50];
5  void solve() {
6      int n,m;
7      std::cin >> n >> m;
8      std::bitset<N> ans[101];
9      ans[0][0] = 1;
10     for(int i = 1; i <= n; i++) {
11         for(int j = 1; j <= m; j++)
12             std::cin >> a[i][j];
13     }
14     for(int i = 1; i <= n; i++) {
15         for(int j = 1; j <= m; j++) {
16             ans[i] |= ans[i-1] << a[i][j];
17         }
18     }
19     int tar , res = N;
20     std::cin >> tar;
21     for(int i = 0; i <= 10000; i++) {
22         if(ans[n][i]) {
23             res = std::min(res, std::abs(tar - i));
24         }
25     }
26     std::cout << res << '\n';
27
28     return;
29 }
30
31 int main() {
32     std::ios::sync_with_stdio(false);
33     std::cin.tie(nullptr);
34     int _ = 1;
35     //std::cin >> _;
36     while(_ --) {
37         solve();
38     }
39     return 0;
40 }
```

E

已知长度为 n 的序列 a_1, a_2, \dots, 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 $O(n \log n)$

```
1  #include <bits/stdc++.h>
2  typedef long long ll;
3
```

```

4 void solve() {
5     int n;
6     std::cin >> n;
7     std::vector<int> a(n), b(n);
8     for(int i = 0; i < n; i++) std::cin >> a[i];
9     auto check = [&] (int t) {
10         b = a;
11         std::set<int> s;
12         for(int i = 0; i < n; i++) {
13             while(b[i] > t) b[i] /= 2;
14             while(s.count(b[i]) && b[i] > 0) b[i] /= 2;
15             s.insert(b[i]);
16         }
17         return (s.size() == (t + 1));
18     };
19     int l = 0, r = n + 1, x = 0;
20     while(l <= r) {
21         int mid = (l + r) >> 1;
22         if(check(mid)) {
23             l = mid + 1;
24             x = mid;
25         }
26         else r = mid - 1;
27     }
28     std::cout << x + 1 << '\n';
29     return;
30 }
31
32 int main() {
33     std::ios::sync_with_stdio(false);
34     std::cin.tie(nullptr);
35     int _ = 1;
36     std::cin >> _;
37     while(_ --) {
38         solve();
39     }
40     return 0;
41 }

```

1992E

$$num = n \times a - b$$

$$i = len \times a - b$$

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

```

1 #include <bits/stdc++.h>
2 typedef long long ll;
3
4 void solve() {
5     int n;
6     std::cin >> n;
7
8     if(n == 1) {
9         std::cout << 9999 << '\n';
10        for(int i = 2; i <= 10000; i++) {
11            std::cout << i << " " << i - 1 << '\n';

```



```

12     }
13     return;
14 }
15 std::string s = std::to_string(n), o = s;
16 for(int i = 1; i <= 7; i++) s = s + o; // 10101010101010
17 std::set<std::pair<int,int>> se;
18 for(int i = 1; i <= 7; i++) {
19     std::string c = s.substr(0,i); // 10101
20     int num = 0;
21     for(int j = 0; j < i; j++) {
22         num = num * 10 + (c[j] - '0');
23     }
24     int p = n - o.length();
25     int q = num - i;
26     if(p == 0) continue;
27     if(q % p == 0 && p != 0){
28         int A = q / p;
29         int B = n * A - num;
30         if(A != 0 && B != 0 && A <= 10000) se.insert({A,B});
31     }
32 }
33 std::cout << se.size() << '\n';
34 for(auto [a,b] : se) {
35     std::cout << a << " " << b << '\n';
36 }
37 return;
38 }
39
40 int main() {
41     std::ios::sync_with_stdio(false);
42     std::cin.tie(nullptr);
43     int _ = 1;
44     std::cin >> _;
45     while(_ --) {
46         solve();
47     }
48     return 0;
49 }

```

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  const int N = 2e6 + 50;
4  struct node{
5      int l,r,v;
6      int lmx,rmx,tmx; // 左右总
7  }t[N];
8
9  void pushup(node &u,node &l,node &r) {
10     u.lmx = std::max(l.lmx, l.lmx == l.r - l.l + 1?l.lmx + r.lmx:l.lmx);
11     u.rmx = std::max(r.rmx, r.rmx == r.r - r.l + 1?r.rmx + l.rmx:r.rmx);

```

```

12     u.tmx = std::max({l.tmx, r.tmx, l.rmx + r.lmx});
13 }
14
15 void push_up(int u) {
16     pushup(t[u], t[u<<1], t[u<<1|1]);
17 }
18 void build(int u, int l, int r) {
19     t[u].l = l; t[u].r = r;
20     if(l == r) {
21         t[u] = {l, r, 0, 1, 1, 1};
22         return;
23     }
24     int mid = (l + r) >> 1;
25     build(u << 1, l, mid);
26     build(u << 1 | 1, mid + 1, r);
27     push_up(u);
28 }
29
30 void modify(int u, int x) {
31     int l = t[u].l, r = t[u].r;
32     if(l == x && r == x) {
33         int pv = t[u].v ^ 1;
34         if(pv == 0) t[u] = {l, r, pv, 1, 1, 1};
35         else t[u] = {l, r, pv, 0, 0, 0};
36         return;
37     }
38     int mid = l + r >> 1;
39     if(x <= mid) modify(u << 1, x);
40     if(x > mid) modify(u << 1 | 1, x);
41     push_up(u);
42 }
43 node query(int u, int l, int r) {
44     if(l <= t[u].l && t[u].r <= r) return t[u];
45     int mid = t[u].l + t[u].r >> 1;
46     if(r <= mid) return query(u << 1, l, r);
47     else if(l > mid) return query(u << 1 | 1, l, r);
48     else {
49         node p, pl, pr;
50         pl = query(u << 1, l, r);
51         pr = query(u << 1 | 1, l, r);
52         pushup(p, pl, pr);
53         return p;
54     }
55 }
56 void solve() {
57     int n, q;
58     std::cin >> n >> q;
59     build(1, 1, n);
60     for(int i = 1, op, x, y; i <= q; i++) {
61         std::cin >> op;
62         if(op == 1) {
63             std::cin >> x;
64             modify(1, x);
65         }
66         else {
67             std::cin >> x >> y;
68             std::cout << query(1, x, y).tmx << '\n';
69         }

```

```

70     }
71     return;
72 }
73
74 int main() {
75     std::ios::sync_with_stdio(false);
76     std::cin.tie(nullptr);
77     int _ = 1;
78     //std::cin >> _;
79     while(_ --) {
80         solve();
81     }
82     return 0;
83 }

```

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1以及后面出现的所有即可, $\text{lowbit}(x)$ 返回第一个1代表的二的幂次

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  void solve() {
4      ll x;
5      std::cin >> x;
6      auto lowbit = [&] (ll i) {
7          return i & (-i);
8      };
9      if(x - lowbit(x) > 0) {
10         std::cout << x - lowbit(x) << '\n';
11     }
12     else std::cout << -1 << '\n';
13     return;
14 }
15
16 int main() {
17     std::ios::sync_with_stdio(false);
18     std::cin.tie(nullptr);
19     int _ = 1;
20     std::cin >> _;
21     while(_ --) {
22         solve();
23     }
24     return 0;
25 }

```

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 ,

```
1  #include <bits/stdc++.h>
2  typedef long long ll;
3
4  void solve() {
5      int n,ans = 0;
6      std::cin >> n;
7      std::string s[2];
8      std::cin >> s[0] >> s[1];
9      int dp[2] = {};
10     for(int i = 0; i < n; i++) {
11         if(s[0][i] == 'R') dp[0]++;
12         else dp[0] = 0;
13         if(s[1][i] == 'R') dp[1]++;
14         else dp[1] = 0;
15         if(s[0][i] == 'R' && s[1][i] == 'R') {
16             int p = dp[0], q = dp[1]; // 重复使用 防止更改
17             dp[0] = std::max(p,q + 1);
18             dp[1] = std::max(q,p + 1); //
19         }
20         ans = std::max({ans,dp[0],dp[1]});
21     }
22     std::cout << std::max(0,ans - 1);
23     return;
24 }
25
26 int main() {
27     std::ios::sync_with_stdio(false);
28     std::cin.tie(nullptr);
29     int _ = 1;
30     //std::cin >> _;
31     while(_ --) {
32         solve();
33     }
34     return 0;
35 }
```

牛客2H

更难的一道 dp

```
1  #include <bits/stdc++.h>
2  typedef long long ll;
3  // 题目要求经过 并非停留
4  void solve() {
5      int n,x,y;
6      std::cin >> n >> x >> y;
7      std::vector<std::array<int,2>> a(n+1);
8      a[0] = {0,0};
9      std::string s;
10     std::cin >> s;
11
12     for(int i = 0; i < n; i++) {
13         a[i+1] = a[i];
```

```

14         if(s[i] == 'W') a[i+1][1] ++;
15         else if(s[i] == 'S') a[i+1][1]--;
16         else if(s[i] == 'A') a[i+1][0]--;
17         else a[i+1][0]++;
18     }
19     ll ans = 0;
20     for(int i = 0; i <= n; i++) {
21         std::cout << a[i][0] << " " << a[i][1] << '\n'; // 前缀位置
22     }
23     std::map<std::array<int,2>,int> mp;
24     for(int i = n; i >= 0; i--) {
25         mp[{a[i][0],a[i][1]}] = i;
26         if(mp.count({a[i][0] + x, a[i][1] + y})) {
27             int j = mp[{a[i][0] + x, a[i][1] + y}]; // j = i, x y = 0
28             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。
29             //std::cout <<i << ": " << j << " ";
30             ans += n - j + 1;
31         }
32     }
33     std::cout <<'\n' << ans << '\n';
34     return;
35 }
36 /*
37 0 0    x,y: 1,1  // -2,-1 + 1,1 = -1,0(r = 5) -> 3-5 3-6
38 -1 0 (i = 1)
39 -1 -1
40 -2 -1
41 -2 0
42 -1 0
43 0 0
44 3: 5 2: 6
45 3
46 */
47 int main() {
48     std::ios::sync_with_stdio(false);
49     std::cin.tie(nullptr);
50     int _ = 1;
51     //std::cin >> _;
52     while(_ --) {
53         solve();
54     }
55     return 0;
56 }

```

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 $0 \leq x, y, z$, 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 $O(n \log n)$

```

1 while (r - l > eps) { // eps 看题目 整数三分
2     mid = (l + r) / 2;
3     lmid = mid - eps;
4     rmid = mid + eps;
5     if (f(lmid) < f(rmid))
6         r = mid;
7     else
8         l = mid;
9 }

```

```

1 #include <bits/stdc++.h>
2 typedef long long ll;
3
4 void solve() {
5     ll A,B,C,n,w;
6     std::cin >> A >> B >> C >> n >> W;
7     ll ans = 1e18;
8     auto check = [&] (int x,int y) {
9         int z = n - x - y;
10        return std::abs(A * x + B * y + C * z - W);
11    };
12    for(int x = 0; x <= n; x++) {
13        int l = 0, r = n - x;
14        while(r - l > 1) {
15            int mid = l + r >> 1;
16            int lmid = mid - 1;
17            int rmid = mid + 1;
18            if(check(x,lmid) < check(x,rmid)) r = mid;
19            else l = mid;
20        }
21        ans = std::min({ans,check(x,l),check(x,r)});
22    }
23    std::cout << ans << '\n';
24    return;
25 }
26
27 int main() {
28     std::ios::sync_with_stdio(false);
29     std::cin.tie(nullptr);
30     int _ = 1;
31     std::cin >> _;
32     while(_ --) {
33         solve();
34     }
35     return 0;
36 }

```

实数三分

```

1 double eps = 1e-9;
2 while(std::fabs(r - l) > eps) {
3     double m1 = (l * 2 + r) / 3;
4     double m2 = (l + r * 2) / 3;
5     if(check(m1) < check(m2)) r = m2;
6     else l = m1;
7 }

```

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

```

1 #include <bits/stdc++.h>
2 typedef long long ll;
3 const int N = 2e5 + 50;
4 struct E {
5     int u,v,w;
6     bool operator < (const E & x) const {
7         return w < x.w;
8     }
9 };
10 E e[N];
11 std::vector<std::pair<int,int>> G[N];
12 int d[N], tim = 0;
13 int fa[N],tag[N];
14 int find(int x) {
15     return fa[x] == x ? x : fa[x] = find(fa[x]);
16 }
17 void solve() {
18     int k;
19     std::cin >> k;
20     tim++;
21     std::vector<int> a(k+1);
22
23     for(int i = 1; i <= k; i++) {
24         std::cin >> a[i];
25         tag[a[i]] = tim;
26         fa[a[i]] = a[i];
27     }
28
29     if(k == 1) {
30         std::cout << 0 << '\n';
31         return;
32     }
33     std::vector<E> ve;
34     for(int i = 1; i <= k; i++) {
35         int x = a[i];
36         for(auto [v,w] : G[x]) {
37             if(tag[v] == tim) ve.push_back({x,v,w});
38         }
39     }
40
41     std::sort(ve.begin(),ve.end());

```

```

42
43     ll ans = 0, cnt = 0;
44     for(auto [u,v,w] : ve) {
45         u = find(u), v = find(v);
46         if(u == v) continue;
47         fa[v] = u;
48         ans += w; cnt += 1;
49         if(cnt == k - 1) {
50             std::cout << ans << '\n';
51             return;
52         }
53     }
54
55     std::cout << -1 << '\n';
56 }
57
58 int main() {
59     std::ios::sync_with_stdio(false);
60     std::cin.tie(nullptr);
61     int n,m,q;
62     std::cin >> n >> m >> q;
63     for(int i = 1; i <= m; i++) {
64         std::cin >> e[i].u >> e[i].v >> e[i].w;
65         d[e[i].u]++;
66         d[e[i].v]++;
67     }
68     for(int i = 1; i <= m; i++) {
69         int u = e[i].u, v = e[i].v;
70         if(d[u] < d[v] || d[u] == d[v] && u < v) G[u].push_back({v,e[i].w});
71         else G[v].push_back({u,e[i].w});
72     }
73
74     while(q --) {
75         solve();
76     }
77     return 0;
78 }

```

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。

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3  ll gcd(ll a,ll b) {
4      if(b == 0) return a;
5      return gcd(b,a%b);
6  }
7
8  void solve() {
9      ll n,D;

```



```

10     std::cin >> n >> D;
11     std::vector<ll> h(n);
12     ll g = 0;
13     for(int i = 0; i < n; i++) {
14         std::cin >> h[i];
15         g = gcd(h[i], g);
16     }
17     std::cout << std::min(D % g, g - D % g) << '\n';
18     return;
19 }
20
21 int main() {
22     std::ios::sync_with_stdio(false);
23     std::cin.tie(nullptr);
24     int _ = 1;
25     //std::cin >> _;
26     while(_ --) {
27         solve();
28     }
29     return 0;
30 }

```

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

```

1  #include <bits/stdc++.h>
2  typedef long long ll;
3
4  void solve() {
5      int n, L, R;
6      int num = 0, ans = 0;
7      std::cin >> n >> L >> R;
8      int s = (n - R - 1) / (R - L) + 1;
9      std::vector<int> h(n), a(n);
10     for(int i = 0; i < n; i++) {
11         std::cin >> h[i];
12         if(h[i] > 2) {
13             a[i] = (h[i] - 1) / 2;
14             num++;
15         }
16     }
17 }

```

```

16     }
17     if(R >= n) {
18         std::cout << "Yes" << '\n';
19         return;
20     }
21     else {
22         if(num < L) std::cout << "No" << '\n';
23         else {
24             for(int i = 0; i < n; i++) {
25                 ans += std::min(a[i],s);
26             }
27             if(ans >= s * L) {
28                 std::cout << "Yes" << '\n';
29             }
30             else std::cout << "No" << '\n';
31         }
32     }
33     return;
34 }
35
36 int main() {
37     std::ios::sync_with_stdio(false);
38     std::cin.tie(nullptr);
39     int _ = 1;
40     //std::cin >> _;
41     while(_ --) {
42         solve();
43     }
44     return 0;
45 }

```

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 个身体部位的状况，0代表健康，1代表不健康。

一共有 k 种药，每种药也用长度为 m 的01串表示，第 i 个字符为'1'代表该药可以治愈第 i 个部位的状况。

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。

$1 \leq m \leq 20$

$1 \leq n \leq 10^4$

$1 \leq k \leq 10$

```

1 // Problem: 医生
2 // Contest: NowCoder
3 // URL: https://ac.nowcoder.com/acm/contest/92972/D

```

```

4 // Memory Limit: 524288 MB
5 // Time Limit: 2000 ms
6 //
7 // Powered by CP Editor (https://cpeditor.org)
8
9 #include <bits/stdc++.h>
10 typedef long long ll;
11 typedef unsigned long long ull;
12 using i64 = long long;
13 const int N = 2e3 + 4;
14 const int mod = 1e9 + 7;
15 void solve() {
16     int n,m;
17     std::cin >> n >> m;
18     auto fsum = [&] (std::string str) -> int {
19         int sum = 0;
20         int len = str.length();
21         for(int i = 0; i < len; i++) {
22             sum = sum * 2 + (str[i] - '0');
23         }
24         return sum;
25     };
26     std::vector<int> p(n);
27     std::vector<int> v;
28     std::vector<bool> vis(1<<m, false);
29     for(int i = 0; i < n; i++) {
30         std::string s;
31         std::cin >> s;
32         p[i] = fsum(s);
33         if(vis[p[i]] == false) {
34             v.push_back(p[i]);
35             vis[p[i]] = true;
36         }
37     }
38     int k;
39     std::cin >> k;
40     std::vector<int> d(k);
41     for(int i = 0; i < k; i++) {
42         std::string s;
43         std::cin >> s;
44         d[i] = fsum(s);
45     }
46     std::vector<std::pair<int,int> > pa;
47     for(int t = 1; t < (1 << k) ; t++) { // 药的所有混合方案
48         int A = 0, B = 0;
49         for(int i = 0; i < k ; i++) { // 解构t的每一位
50             if(t & (1 << i)) {
51                 A |= d[i]; // 自己有的药 混合
52                 B ++;
53             }
54         }
55         pa.push_back({B,A});
56     }
57     std::sort(pa.begin(), pa.end()); // 按照 混合的数量排序
58     std::vector<int> ans((1<<m), -1);
59     for(auto c : v) {
60         if(c == 0) { // 无病 00000
61             ans[c] = 0;

```

```

62         continue;
63     }
64     for(auto [b,a] : pa) {
65         if( (a & c) == c) { // 枚举所有混合药 能治
66             ans[c] = b;
67             break;
68         }
69     }
70 }
71 for(auto c : p) {
72     std::cout << ans[c] << '\n';
73 }
74 return;
75 }
76
77 int main() {
78     std::ios::sync_with_stdio(false);
79     std::cin.tie(nullptr);
80     int _ = 1;
81     //std::cin >> _;
82     while(_ --) {
83         solve();
84     }
85     return 0;
86 }

```

[E - Mod Sigma Problem](#)

问题陈述

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

$$\sum_{1 \leq l < r \leq N} \text{for } (1 \leq r \leq N) \text{ for } (1 \leq l \leq r)$$

$$s_0, s_1, s_2, \dots, s_{n-1}$$

$$s_r - s_{l-1} \% M = (s_r \% M - s_{l-1} \% M) \% M$$

$$\sum_{1 \leq l < r \leq N} \left(\sum_{k=l}^r A_i \% M \right)$$

$$= \sum_{1 \leq l < r \leq N} (s_r - s_{l-1} \% M) \cdot \frac{s_{l-1} \cdot (n-l+1)}{s_{n-1} \cdot 1}$$

$$(s_r - s_{l-1}) \% M = \begin{cases} s_r - s_{l-1} & (s_r \geq s_{l-1}) \\ s_r - s_{l-1} + M & (s_r < s_{l-1}) \end{cases}$$

$$s_r \leq M$$

$$= \sum_{1 \leq l < r \leq N} (s_r - s_{l-1} + M \cdot (s_r < s_{l-1}))$$

$$= \sum_{1 \leq l < r \leq N} s_r - \sum_{1 \leq l < r \leq N} s_{l-1} + M \cdot \sum_{1 \leq l < r \leq N} (s_r < s_{l-1})$$

$$= \sum_{r=1}^N s_r \cdot r - \sum_{l=1}^N s_{l-1} \cdot (n-l+1) + M \cdot \text{逆序对数量}$$

```

1 // Problem: E - Mod Sigma Problem
2 // Contest: AtCoder - AtCoder Beginner Contest 378
3 // URL: https://atcoder.jp/contests/abc378/tasks/abc378_e
4 // Memory Limit: 1024 MB
5 // Time Limit: 2000 ms
6 //
7 // Powered by CP Editor (https://cpeditor.org)
8
9 #include <bits/stdc++.h>
10 typedef long long ll;
11 typedef unsigned long long ull;
12 using i64 = long long;
13 const int N = 2e5 + 4;
14 const int mod = 1e9 + 7;
15 int b[N];
16 i64 ans = 0;
17 i64 msort(std::vector<i64> &a, int l, int r)
18 {
19     if(l >= r) return 0;
20     int mid = (l + r) / 2;
21     msort(a, l, mid);
22     msort(a, mid + 1, r);
23     int i = l, j = mid + 1, k = l;

```

```

24     while(i <= mid && j <= r)
25     {
26         if(a[i] <= a[j]) b[k++] = a[i++];
27         else {
28             b[k++] = a[j++];
29             ans += (mid - i + 1);
30         }
31     }
32     while(i <= mid) b[k++] = a[i++];
33     while(j <= r) b[k++] = a[j++];
34     for(int i = 1; i <= r; i++) a[i] = b[i];
35     return ans;
36 }
37
38 void solve() {
39     int n,m;
40     std::cin >> n >> m;
41     std::vector<i64> a(n+1),s(n+1),t(n+1);
42
43     for(int i = 1; i <= n; i++) {
44         std::cin >> a[i];
45         s[i] = s[i-1] + a[i];
46         s[i] %= m;
47     }
48     std::copy(s.begin(),s.end(),t.begin());
49     i64 res = msort(t,1,n);
50     i64 Ans = 0;
51     for(int i = 1; i <= n; i++) {
52         Ans += s[i] * i;
53     }
54     for(int i = 1; i <= n; i++) {
55         Ans -= (s[i-1] * (n - i + 1));
56     }
57     Ans += m * res;
58     std::cout << Ans << '\n';
59     return;
60 }
61
62 int main() {
63     std::ios::sync_with_stdio(false);
64     std::cin.tie(nullptr);
65     int _ = 1;
66     //std::cin >> _;
67     while(_ --) {
68         solve();
69     }
70     return 0;
71 }

```