Содержание

1 Общее

1.1 Пробный тур

1. Пописать код и посдавать каждому члену команды.
2. Сравнить скорости компа и тестирующей системы на Флойде ($n \sim 1050$).
3. Запустить все IDE, проверить, что все настройки работают, проверить, что работает c++11.
4. Проверить, есть ли int128.
5. Проверить работу прагм в тестирующей системе.
6. Почекать максимальный размер отправляемого кода.

1.2 Чеклист при отправке

1. Протестить на всех тестах из примера и других рандомных тестах.
2. Протестить все крайние случаи.
3. Убрать дебаг вывод.
4. Проверить, все ли хорошо с мультитестом.

1.3 Факты из жизни IMO KINGS

- 1 января 2000 года – суббота, 1 января 1900 года – понедельник.
- Високосные года: если $400 \mid a$, либо если $4 \mid a$, но не $100 \mid a$.
- $INTMIN = -2.147.483.648$, $INTMAX = 2.147.483.647$, $UINTMAX = 4.294.967.295$, $SHRTMIN = -32.768$, $SHRTMAX = 32.767$, $LLONGMIN = -9.223.372.036.854.775.808$, $LLONGMAX = 9.223.372.036.854.775.807$, 18.446.744.073.709.551.615.

1.4 Простые числа

10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
	−47	−63	−113	−123	−137	−111	−213	−267	−231	−231	−233	−447	−203	−429	−369	−413	−369
	−41	−59	−99	−119	−117	−99	−179	−261	−219	−179	−153	−411	−179	−423	−359	−273	−363
	−39	−53	−93	−99	−93	−93	−173	−249	−213	−171	−143	−357	−171	−357	−357	−261	−291
	−33	−47	−77	−93	−83	−71	−161	−243	−183	−167	−137	−341	−147	−341	−329	−239	−263
	−29	−33	−71	−77	−69	−69	−153	−239	−167	−149	−123	−299	−77	−191	−191	−177	−251
	−27	−29	−69	−71	−47	−63	−69	−203	−149	−129	−101	−267	−71	−173	−183	−81	−171
−8	−21	−23	−59	−39	−41	−57	−59	−117	−119	−93	−63	−237	−69	−123	−149	−57	−137
−7	−17	−17	−51	−29	−39	−29	−41	−107	−71	−57	−41	−201	−41	−117	−113	−39	−123
−5	−11	−9	−33	−11	−21	−27	−29	−71	−57	−53	−39	−137	−29	−53	−83	−23	−33
−3	−3	−3	−27	−9	−17	−9	−11	−63	−33	−23	−11	−29	−27	−11	−63	−3	−11

+1	+1	+9	+7	+3	+3	+19	+7	+7	+19	+3	+39	+37	+31	+37	+61	+3	+3
+3	+3	+13	+9	+19	+33	+79	+37	+9	+33	+19	+61	+51	+67	+91	+69	+13	+9
+7	+7	+19	+37	+43	+37	+103	+39	+21	+61	+57	+63	+99	+97	+159	+79	+19	+31
+9	+9	+21	+39	+49	+39	+121	+49	+33	+69	+63	+91	+129	+99	+187	+99	+21	+79
+13	+13	+31	+61	+57	+81	+139	+73	+87	+97	+69	+121	+183	+133	+223	+453	+49	+177
+19	+27	+33	+67	+69	+99	+141	+81	+93	+103	+73	+163	+259	+139	+241	+481	+81	+183
+21	+31	+39	+69	+103	+117	+169	+123	+97	+121	+91	+169	+267	+169	+249	+597	+99	+201
+27	+37	+49	+79	+109	+121	+189	+127	+103	+141	+103	+177	+273	+183	+259	+613	+141	+283
+31	+39	+51	+91	+129	+133	+223	+193	+123	+147	+129	+189	+279	+261	+273	+639	+181	+381
+33	+49	+61	+93	+151	+151	+229	+213	+181	+207	+171	+193	+283	+357	+279	+669	+337	+387

1.5 Максимальное количество делителей

$\leqslant N$	n	Факторизация	$d(n)$
20	12	$2^2 \cdot 3^1$	6
50	48	$2^4 \cdot 3^1$	10
100	60	$2^2 \cdot 3^1 \cdot 5^1$	12
10^3	840	$2^3 \cdot 3^1 \cdot 5^1 \cdot 7^1$	32
10^4	9240	$2^3 \cdot 3^1 \cdot 5^1 \cdot 7^1 \cdot 11^1$	64
10^5	83 160	$2^3 \cdot 3^3 \cdot 5^1 \cdot 7^1 \cdot 11^1$	128
10^6	720 720	$2^4 \cdot 3^2 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot 13^1$	240
10^7	8 648 640	$2^6 \cdot 3^3 \cdot 5^1 \cdot 7^1 \cdot 11^1 \cdot 13^1$	448
10^8	91 891 800	$2^3 \cdot 3^3 \cdot 5^2 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^1$	768
10^9	931 170 240	$2^6 \cdot 3^2 \cdot 5^2 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1$	1344
10^{11}	97 772 875 200	$2^6 \cdot 3^3 \cdot 5^2 \cdot 7^2 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1$	4032
10^{12}	963 761 198 400	$2^6 \cdot 3^4 \cdot 5^2 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1 \cdot 23^1$	6720
10^{15}	866 421 317 361 600	$2^6 \cdot 3^4 \cdot 5^2 \cdot 7^1 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1 \cdot 23^1 \cdot 29^1 \cdot 31^1$	26 880
10^{18}	897 612 484 786 617 600	$2^8 \cdot 3^4 \cdot 5^2 \cdot 7^2 \cdot 11^1 \cdot 13^1 \cdot 17^1 \cdot 19^1 \cdot 23^1 \cdot 29^1 \cdot 31^1 \cdot 37^1$	103 680

1.6 Формулы

- Расстояние между точками по сфере: $L = R \cdot \arccos(\cos\theta_1 \cdot \cos\theta_2 + \sin\theta_1 \cdot \sin\theta_2 \cdot \cos(\varphi_1 - \varphi_2))$, где θ – широты (от $-\pi$ до π), φ – долготы (от $-\pi$ до π)
- Объем шарового сегмента: $V = \pi h^2(R - \frac{1}{3}h)$, где h – высота от вершины сектора до секущей плоскости
- Площадь поверхности шарового сегмента: $S = 2\pi Rh$, где h – высота
- $2^{23} \cdot 7 \cdot 17 + 1 = 998\,244\,353$ – простое, первообразный корень – 3
- Код Грея: $g_n = n \operatorname{XOR} \frac{n}{2}$
- Числа Фибоначчи: $F_0 = 0, F_1 = 1, F_n = \dfrac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$
- Числа Стирлинга: $s(n,k)$ — количество перестановок на n элементах, в которых ровно k циклов. $S(n,k)$ – это количество способов разбить n-элементное множество на k непустых подмножеств.
  - † $s(n,k) = (n-1) \cdot s(n-1,k) + s(n-1,k-1)$
  - † $S(n,k) = k \cdot S(n-1,k) + S(n-1,k-1)$
  - † $x^{\underline{n}} = x \cdot (x-1) \cdot ... \cdot (x-n+1) = \sum\limits_{i=1}^{n} (-1)^{n-i} \cdot s(n,\ i) \cdot x^i$
  - † $x^n = \sum\limits_{i=0}^{n} S(n,\ i) \cdot x^{\underline{i}}$
- Число разбиений на неубывающие натуральные слагаемые:

i	p_i	i	p_i	i	p_i	i	p_i	i	p_i	i	p_i
0	1	10	42	20	627	30	5604	40	37 338	50	204 226
1	1	11	56	21	792	31	6842	41	44 583	51	**239**943
2	2	12	77	22	1002	32	8349	42	53 174	52	281 589
3	3	13	101	23	1255	33	10 143	43	63 261	53	329 931
4	5	14	135	24	1575	34	12 310	44	75 175	54	386 155
5	7	15	176	25	1958	35	14 883	45	89 134	55	451 276
6	11	16	231	26	2436	36	17 977	46	105 558	56	526 823
7	15	17	297	27	3010	37	21 637	47	124 754	57	614 154
8	22	18	385	28	3718	38	26 015	48	147 273	58	715 220
9	30	19	490	29	4565	39	31 185	49	173 525	59	831 820

$$p_{100} = 190\,569\,292$$

- Gambler's ruin. У первого игрока есть n_1 монет, у второго n_2. На каждом шаге с вероятностью p второй отдает одну монету первому, а с вероятностью $q = 1 - p$ первый отдает одну монету второму. Игра заканчивается, когда у кого-нибудь не остается монет. Тогда первый выигрывает с вероятностью

$$\frac{1 - \left(\frac{p}{q}\right)^{n_1}}{1 - \left(\frac{p}{q}\right)^{n_1+n_2}}.$$

1.7 Чеклист при WA

1. Распечатать.
2. Отдать комп следующему.
3. Проверить код по строчке.
4. Проверить, не забыл ли случаи.
5. Потестить на компе.
6. Написать стресс.
7. Возможно, решение вообще неправильное.
8. Возможно, WA где-то не там, где ты думаешь.

2 Коды

2.1 Basic setup

1. Запустить все IDE.
   2. Сразу сделать файлы для всех задач.
   3. Для каждой задачи свой input file.
   4. Сделать template file, из него скопировать во все, не забыв поменять название input file.

```
// SPBU: Havka - ne papstvo
#ifdef ONPC
    # define _GLIBCXX_DEBUG
    #define deb(a) cerr << "========" << #a << " = " << a << endl;
#else
    #define deb(a)
#endif
#define sz(a) ((int)((a).size()))
#include <bits/stdc++.h>
#define string basic_string<unsigned char>
#define char unsigned char

using namespace std;
mt19937 rnd(239);
//mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
//mt19937_64 rnd(239);

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef vector<int> vi;
typedef vector<ll> vl;

int solve()
{
    int n;
    if (!(cin >> n))
        return 1;

    return 0;
}

int32_t main()
{
#ifdef ONPC
    freopen("inA.txt", "r", stdin);
#endif
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int TET = 1e9;
    // cin >> TET;
    for (int i = 1; i <= TET; i++)
    {
        if (solve())
            break;
        #ifdef ONPC
            cout << "\n____________________________" << endl;
        #endif
    }
    #ifdef ONPC
        cerr << endl << "finished in " << clock() * 1.0 / CLOCKS_PER_SEC << " sec" << endl;
    #endif
}
```

2.2 Vim setup

```
colo desert
"colo darkblue

noremap <F1> <ESC>:tabprev <CR>
inoremap <F1> <ESC>
inoremap {<CR> {<CR>}<ESC>k$A<CR>

noremap <F9> <ESC> :wall! <CR> :!g++ -fsanitize=address -std=c++17 -Wall -Wextra
-Wshadow -Wno-unused-result -DONPC -O2 -o %< % && ./%< < inp<CR>

inoremap <F9> <ESC> :wall! <CR> :!g++ -fsanitize=address -std=c++17 -Wall -Wextra
-Wshadow -Wno-unused-result -DONPC -O2 -o %< % && ./%< < inp<CR>

command! Kek source ~/.vimrc
autocmd FocusLost * redraw!
"command! LLDB :!clang++ -fsanitize=address -std=c++17 -O0 -g -o "%<" "%" && lldb %<
"command! Gdb !g++ -std=c++17 -O0 -g -o %<

set autoindent
set autoread
set cin
set expandtab
set history=1000
set hlsearch
set number
set sw=4
set smarttab
set tabstop=4

syntax on
```

2.3 libraries

```
#include <climits> #include <cmath>
#include <cstdio> #include <cstdlib>
#include <cstring> #include <ctime>
#include <algorithm> #include <bitset>
#include <complex> #include <deque>
#include <iostream> #include <map>
#include <queue> #include <set>
#include <stack> #include <string>
#include <vector> #include <utility>
#include <random> #include <tuple>
#include <unordered_map> #include <unordered_set>
#include <cassert>
```

2.4 Полезное

```
Прагмы:
#pragma comment(linker, "/stack:200000000")
#pragma GCC optimize("Ofast")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,avx,avx2")

Встроенный декартач:
#include <ext/pb_ds/assoc_container.hpp> // Общий файл.
#include <ext/pb_ds/tree_policy.hpp> // Содержит класс tree_order_statistics_node_update
//#include <ext/pb_ds/detail/standard_policies.hpp>
using namespace __gnu_pbds;
typedef
tree<
```

```
  pair<int,int>,
  null_type,
  less<pair<int,int>>,
  rb_tree_tag,
  tree_order_statistics_node_update>
            }
ordered_set;

ordered set q;
q.find_by_order(1);
q.order_of_key(2);

Atomic hashset:
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
gp_hash_table<int, int> table;
-----------------------------------------------
const int RANDOM = chrono::high_resolution_clock::now().time_since_epoch().count();
struct chash {
    int operator()(int x) const { return x ^ RANDOM; }
};
gp_hash_table<key, int, chash> table;

Atomic hashmap:

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef cc_hash_table<int, int, hash<int>> ht;


Фишки битсета:
bitset<N> a;
for (int i = a._Find_first(); i != a.size(); i = a._Find_next(i)) { cout << i };
```

2.5 Python example

```
def main():
    n, k = list(map(int, input().split()))
    for i in range(1, n):
        if i in lst:
            print(i ** 2)
```

2.6 Дробь с минимальным знаменателем между двумя другими

```
rat find_best(rat l, rat r) {
    if (l.x >= l.y) {
        ll d = l.x / l.y;
        rat res = find_best(rat(l.x - d * l.y, l.y), rat(r.x - d * r.y, r.y));
        res.x += res.y * d;
        return res;
    }
    if (r.x > r.y)
        return rat(1, 1);
    rat res = find_best(rat(r.y, r.x), rat(l.y, l.x));
    return rat(res.y, res.x);
}
```

2.7 Битсет

```
const int SZ = 6;
```

```
const int BASE = (1 << SZ);
const int MOD = BASE - 1;

struct Bitset {
    typedef unsigned long long T;
    vector<T> data;
    int n;
    void resize(int nn) {
        n = nn;
        data.resize((n + BASE - 1) / BASE);
    }
    void set(int pos, int val) {
        int id = pos >> SZ;
        int rem = pos & MOD;
        data[id] ^= data[id] & (1LL << rem);
        data[id] |= val * (1LL << rem);
    }
    int get(int pos) {
        return (data[pos >> SZ] >> (pos & MOD)) & 1;
    }
    // k > 0 -> (*this) << k
    // k < 0 -> (*this) >> (-k)
    Bitset shift (int k) {
        Bitset res;
        res.resize(n);
        int s = k / BASE;
        int rem = k % BASE;
        if (rem < 0) {
            rem += BASE;
            s--;
        }
        int p1 = BASE - rem;
        T mask = (p1 == 64)? -1: (1LL << p1) - 1;
        for (int i = max(0, -s); i < sz(data) - max(s, 0); i++) {
            res.data[i + s] |= (data[i] & mask) << rem;
        }
        if (rem != 0) {
            for (int i = max(0, -s - 1); i < sz(data) - max(s + 1, 0); i++) {
                res.data[i + s + 1] |= (data[i] >> p1) & ((1LL << rem) - 1);
            }
        }
        int cc = data.size() * BASE - n;
        res.data.back() <<= cc;
        res.data.back() >>= cc;
        return res;
    }
};
```

2.8 FFT

```
// typedef complex<ld> cmpl;
const ll MOD = 998244353, MOD2 = MOD * MOD, root = 3, sub = 15311432, sub_inv = 469870224;
const ld PI = atan2(0, -1);
ll st[T]; // cmpl
inline ll power(ll a, ll b) // cmpl cmpl cmpl {
    if (b == 0) return 1;
    ll t = power(a, (b >> 1)); // cmpl
    t = (t * t) % MOD;
    if (b & 1LL) t = (t * a) % MOD;
    return t;
}
```

```
inline int inv(int n, int b) {
    int ans = 0;
    for (int i = 0; i < b; i++)
    {
        ans = (ans << 1) + (n & 1);
        n >>= 1;
    }
    return ans;
}
int p;
ll w; // ld
inline void fft(vector<ll> &a) // cmpl {
    int u = 1, ps, t;
    ll bi, bii; // cmpl
    for (int i = 0; i < sz(a); i++) {
        ps = inv(i, p);
        if (ps < i) swap(a[i], a[ps]);
    }
    for (int z = 0; z < p; z++, u <<= 1)
        for (int i = 0; i < (1 << p); i++) {
            ps = i >> z;
            if ((ps & 1) == 0) {
                t = i & (u - 1);
                bi = (a[i] + st[t << (p - z - 1)] * a[i + u]) % MOD;
                bii = (a[i] + st[(t + u) << (p - z - 1)] * a[i + u]) % MOD;
                a[i] = bi;
                a[i + u] = bii;
            }
        }
}
vector<ll> multiply(vector<ll> a, vector<ll> b) {
    int k = sz(a) + sz(b);
    p = 0;
    while ((1 << p) < k) p++;
    w = power(sub, (1 << (23 - p))); // al = PI / (ld)(1 << p), w = cmpl(cos(al), sin(al))
    st[0] = 1LL;
    for (int i = 1; i < (1 << p); i++)
        st[i] = (st[i - 1] * w) % MOD;
    while (sz(a) < (1 << p)) a.push_back(0);
    while (sz(b) < (1 << p)) b.push_back(0);
    fft(a);
    fft(b);
    w = power(sub_inv, (1 << (23 - p))); // al = -PI / (ld)(1 << p), w = cmpl(cos(al), sin(al))
    st[0] = 1LL;
    for (int i = 1; i < (1 << p); i++)
        st[i] = (st[i - 1] * w) % MOD;
    vector<ll> v; // ld
    for (int i = 0; i < (1 << p); i++) v.push_back((a[i] * b[i]) % MOD);
    fft(v);
    ll d = power((1 << p), MOD - 2);
    for (int i = 0; i < (1 << p); i++) v[i] = (d * v[i]) % MOD; // v[i] /= (ld)(1 << p)
    while (sz(v) && v.back() == 0) v.pop_back();
    return v;
}
```

2.9 Сумма линейных функций по модулю

```
// sum(i=0..n-1) (a + b * i) div m
ll solve(ll n, ll a, ll b, ll m) {
    if (b == 0) return n * (a / m);
    if (a >= m) return n * (a / m) + solve(n, a % m, b, m);
```

```
    if (b >= m) return n * (n - 1) / 2 * (b / m) + solve(n, a, b % m, m);
    return solve((a + b * n) / m, (a + b * n) % m, m, b);
}
```

2.10 Штор-Вагнер

```
const int MAXN = 500;
int n, g[MAXN][MAXN];
int best_cost = 1e9;
vector<int> best_cut, v[MAXN];
int w[MAXN];
bool exist[MAXN], in_a[MAXN];

void mincut() {
    for (int i = 0; i < n; i++)
        v[i].assign(1, i);
    memset(exist, true, sizeof exist);
    for (int ph = 0; ph < n - 1; ph++) {
        memset(in_a, false, sizeof in_a);
        memset(w, 0, sizeof w);
        for (int it = 0, prev = 0; it < n - ph; it++) {
            int sel = -1;
            for (int i = 0; i < n; i++)
                if (exist[i] && !in_a[i] && (sel == -1 || w[i] > w[sel]))
                    sel = i;
            if (it == n - ph - 1) {
                if (w[sel] < best_cost)
                    best_cost = w[sel],  best_cut = v[sel];
                v[prev].insert(v[prev].end(), v[sel].begin(), v[sel].end());
                for (int i = 0; i < n; i++)
                    g[prev][i] = g[i][prev] += g[sel][i];
                exist[sel] = false;
            }
            else {
                in_a[sel] = true;
                for (int i = 0; i < n; i++)
                    w[i] += g[sel][i];
                prev = sel;
            }
        }
    }
    // ans: best_cost, best_cut
}
```

2.11 Диниц

```
const int MAXN = ...;
const int INF = 1e9 + 9;

struct edge {
    int a, b, cap, flow;
};

int n, s, t, d[MAXN], ptr[MAXN], q[MAXN];
vector<edge> e;
vector<int> g[MAXN];

void add_edge (int a, int b, int cap) {
    edge e1 = { a, b, cap, 0 };
    edge e2 = { b, a, 0, 0 };
    g[a].push_back ((int) e.size());
```

```
    e.push_back (e1);
    g[b].push_back ((int) e.size());
    e.push_back (e2);
}

bool bfs() {
    int qh=0, qt=0;
    q[qt++] = s;
    memset (d, -1, n * sizeof d[0]);
    d[s] = 0;
    while (qh < qt && d[t] == -1) {
        int v = q[qh++];
        for (size_t i=0; i<g[v].size(); ++i) {
            int id = g[v][i],
                to = e[id].b;
            if (d[to] == -1 && e[id].flow < e[id].cap) {
                q[qt++] = to;
                d[to] = d[v] + 1;
            }
        }
    }
    return d[t] != -1;
}

int dfs (int v, int flow, int can) {
    if (!flow)  return 0;
    if (v == t)  return flow;
    for (; ptr[v]<(int)g[v].size(); ++ptr[v]) {
        int id = g[v][ptr[v]],
            to = e[id].b;
        if (d[to] != d[v] + 1 || e[id].cap - e[id].flow < can)  continue;
        int pushed = dfs (to, min (flow, e[id].cap - e[id].flow), can);
        if (pushed) {
            e[id].flow += pushed;
            e[id^1].flow -= pushed;
            return pushed;
        }
    }
    return 0;
}
int dinic() {
    int flow = 0;
    for (;;) {
        if (!bfs())  break;
        memset (ptr, 0, n * sizeof ptr[0]);
        for (int k = 30; k >= 0; k--) // just k = 0, if no scaling {
            while (int pushed = dfs(s, INF, (1 << k)))
                flow += pushed;
        }
    }
    return flow;
}
```

2.12 Dynamic CHT

```
const ll is_query = -(1LL << 62);

struct Line {
    ll m, b;
    mutable function<const Line *()> succ;
```

```
    bool operator<(const Line &rhs) const {
        if (rhs.b != is_query) return m < rhs.m;
        const Line *s = succ();
        if (!s) return 0;
        ll x = rhs.m;
        return b - s->b < (s->m - m) * x;
    }
};

struct HullDynamic : public multiset<Line> {
    bool bad(iterator y) {
        auto z = next(y);
        if (y == begin()) {
            if (z == end()) return 0;
            return y->m == z->m && y->b <= z->b;
        }
        auto x = prev(y);
        if (z == end()) return y->m == x->m && y->b <= x->b;
        return (x->b - y->b) * (z->m - y->m) >= (y->b - z->b) * (y->m - x->m);
    }

    void insert_line(ll m, ll b) {
        auto y = insert({m, b});
        y->succ = [=] { return next(y) == end() ? 0 : &*next(y); };
        if (bad(y)) {
            erase(y);
            return;
        }
        while (next(y) != end() && bad(next(y))) erase(next(y));
        while (y != begin() && bad(prev(y))) erase(prev(y));
    }

    ll eval(ll x) {
        auto l = *lower_bound((Line) {x, is_query});
        return l.m * x + l.b;
    }
```

2.13 Пересечение матроидов

```
    // check(ctaken, 1) -- first matroid
    // check(ctaken, 2) -- second matroid
    vector<char> taken(m);
    while (1) {
        vector<vector<int>> e(m);
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < m; j++) {
                if (taken[i] && !taken[j]) {
                    auto ctaken = taken;
                    ctaken[i] = 0;
                    ctaken[j] = 1;
                    if (check(ctaken, 2)) {
                        e[i].push_back(j);
                    }
                }
                if (!taken[i] && taken[j]) {
                    auto ctaken = taken;
                    ctaken[i] = 1;
                    ctaken[j] = 0;
                    if (check(ctaken, 1)) {
                        e[i].push_back(j);
```

```
                }
            }
        }
    }
    vector<int> type(m);
    // 0 -- cant, 1 -- can in \2, 2 -- can in \1
    for (int i = 0; i < m; i++) {
        if (!taken[i]) {
            auto ctaken = taken;
            ctaken[i] = 1;
            if (check(ctaken, 2)) type[i] |= 1;
        }
        if (!taken[i]) {
            auto ctaken = taken;
            ctaken[i] = 1;
            if (check(ctaken, 1)) type[i] |= 2;
        }
    }
    vector<int> w(m);
    for (int i = 0; i < m; i++) {
        w[i] = taken[i] ? ed[i].c : -ed[i].c;
    }
    vector<pair<int, int>> d(m, {INF, 0});
    for (int i = 0; i < m; i++) {
        if (type[i] & 1) d[i] = {w[i], 0};
    }
    vector<int> pr(m, -1);
    while (1) {
        vector<pair<int, int>> nd = d;
        for (int i = 0; i < m; i++) {
            if (d[i].first == INF) continue;
            for (int to : e[i]) {
                if (nd[to] > make_pair(d[i].first + w[to], d[i].second + 1)) {
                    nd[to] = make_pair(d[i].first + w[to], d[i].second + 1);
                    pr[to] = i;
                }
            }
        }
        if (d == nd) break;
        d = nd;
    }
    int v = -1;
    for (int i = 0; i < m; i++) {
        if ((d[i].first < INF && (type[i] & 2)) && (v == -1 || d[i] < d[v])) v = i;
    }
    if (v == -1) break;
    while (v != -1) {
        sum += w[v];
        taken[v] ^= 1;
        v = pr[v];
    }
    ans[--cnt] = sum;
}
```

2.14 Пожилой суффиксный автомат

```
struct state {
    int len, link;
    map<char,int> next;
};
const int MAXLEN = 112345;
```

```
state st[MAXLEN*2];
int sz, last;
void sa_init() {
    sz = last = 0;
    st[0].len = 0;
    st[0].link = -1;
    ++sz;
    /*
    // этот код нужен, только если автомат
    // строится много раз для разных строк:
    for (int i=0; i<MAXLEN*2; ++i)
        st[i].next.clear();
    */
}
void sa_extend (char c) {
    int cur = sz++;
    st[cur].len = st[last].len + 1;
    int p;
    for (p=last; p!=-1 && !st[p].next.count(c); p=st[p].link)
        st[p].next[c] = cur;
    if (p == -1)
        st[cur].link = 0;
    else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len)
            st[cur].link = q;
        else {
            int clone = sz++;
            st[clone].len = st[p].len + 1;
            st[clone].next = st[q].next;
            st[clone].link = st[q].link;
            for (; p!=-1 && st[p].next[c]==q; p=st[p].link)
                st[p].next[c] = clone;
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}
```

2.15 Быстрый ввод-вывод

```
#include <cstdio>

/** Interface */
inline int readChar();
template <class T = int> inline T readInt();
template <class T> inline void writeInt( T x, char end = 0 );
inline void writeChar( int x );
inline void writeWord( const char *s );

/** Read */
static const int buf_size = 4096;
inline int getChar() {
    static char buf[buf_size];
    static int len = 0, pos = 0;
    if (pos == len)
        pos = 0, len = fread(buf, 1, buf_size, stdin);
    if (pos == len)
        return -1;
    return buf[pos++];
}
```

```
inline int readChar() {
    int c = getChar();
    while (c <= 32)
        c = getChar();
    return c;
}
template <class T>
inline T readInt() {
    int s = 1, c = readChar();
    T x = 0;
    if (c == '-')
        s = -1, c = getChar();
    while ('0' <= c && c <= '9')
        x = x * 10 + c - '0', c = getChar();
    return s == 1 ? x : -x;
}

/** Write */
static int write_pos = 0;
static char write_buf[buf_size];

inline void writeChar( int x ) {
    if (write_pos == buf_size)
        fwrite(write_buf, 1, buf_size, stdout), write_pos = 0;
    write_buf[write_pos++] = x;
}
template <class T>
inline void writeInt( T x, char end ) {
    if (x < 0)
        writeChar('-'), x = -x;

    char s[24];
    int n = 0;
    while (x || !n)
        s[n++] = '0' + x % 10, x /= 10;
    while (n--)
        writeChar(s[n]);
    if (end)
        writeChar(end);
}
inline void writeWord( const char *s ) {
    while (*s)
        writeChar(*s++);
}
struct Flusher {
    ~Flusher() {
        if (write_pos)
            fwrite(write_buf, 1, write_pos, stdout), write_pos = 0;
    }
} flusher;

// cin - 3.02, scanf - 1.2, cin+sync - 0.71
// fastRead getchar - 0.53, fastRead fread - 0.15
```

2.16 Пересечение полуплоскостей

```
namespace hpi{
 const double eps = 1e-8;
 typedef pair<long double, long double> pi;
 bool z(long double x){ return fabs(x) < eps; }
 long double ccw(pi a, pi b, pi c){
```

```
 long double dx1 = b.first - a.first;
 long double dy1 = b.second - a.second;
 long double dx2 = c.first - a.first;
 long double dy2 = c.second - a.second;
 return dx1 * dy2 - dy1 * dx2;
}
struct line{
 long double a, b, c;
 bool operator<(const line &l)const{
  bool flag1 = pi(a, b) > pi(0, 0);
  bool flag2 = pi(l.a, l.b) > pi(0, 0);
  if(flag1 != flag2) return flag1 > flag2;
  long double t = ccw(pi(0, 0), pi(a, b), pi(l.a, l.b));
  return z(t) ? c * hypot(l.a, l.b) < l.c * hypot(a, b) : t > 0;
 }
 pi slope(){ return pi(a, b); }
};

pi cross(line a, line b){
 long double det = a.a * b.b - b.a * a.b;
 return pi((a.c * b.b - a.b * b.c) / det, (a.a * b.c - a.c * b.a) / det);
}
bool bad(line a, line b, line c){
 if(ccw(pi(0, 0), a.slope(), b.slope()) <= 0) return false;
 pi crs = cross(a, b);
 return crs.first * c.a + crs.second * c.b >= c.c;
}
bool solve(vector<line> v, vector<pi> &solution){ // ax + by <= c;
 sort(v.begin(), v.end());
 deque<line> dq;
 for(auto &i : v){
  if(!dq.empty() && z(ccw(pi(0, 0), dq.back().slope(), i.slope()))) continue;
  while(dq.size() >= 2 && bad(dq[dq.size()-2], dq.back(), i)) dq.pop_back();
  while(dq.size() >= 2 && bad(i, dq[0], dq[1])) dq.pop_front();
  dq.push_back(i);
 }
 while(dq.size() > 2 && bad(dq[dq.size()-2], dq.back(), dq[0])) dq.pop_back();
 while(dq.size() > 2 && bad(dq.back(), dq[0], dq[1])) dq.pop_front();
 vector<pi> tmp;
 for(int i=0; i<dq.size(); i++){
  line cur = dq[i], nxt = dq[(i+1)%dq.size()];
  if(ccw(pi(0, 0), cur.slope(), nxt.slope()) <= eps) return false;
  tmp.push_back(cross(cur, nxt));
 }
 solution = tmp;
 return true;
}
}
```

2.17 Поллард

```
const int maxc = 500010;
ll n, x[maxc];
ll mul(ll a, ll b, ll m) // m <= 8e18
    ll k = ((ld)a * b) / m;
    ll r = a * b - k * m;
    while (r < 0) r += m;
    while (r >= m) r -= m;
    return r;
}
void slow(int n) {
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) {
            cout << i << ' ' << n / i << endl;
            exit(0);
        }
    cout << "IMPOSSIBLE" << endl;
    exit(0);
}
int main() {
    cin >> n;
    if (n <= (int)1e6)
        Slow(n);

    ll C = 2 * pow(n, 1.0 / 4);
    for (int cnt = 0; cnt < 4; cnt++) {
        x[0] = abs((int)rnd()) % (n - 1) + 1;
        for (i = 0; i < C; i++)
            x[i + 1] = (mul(x[i], x[i], n) + 3) % n;
        for (int i = 0; i < C; i++) {
            ll g = gcd(abs(x[i] - x[C]), n);
            if (g != 1 && g != n) {
                cout << g << ' ' << n / g << endl;
                return 0;
            }
        }
    }
    cout << "IMPOSSIBLE" << endl;
    return 0;
}
```

2.18 mincostmaxflow

```
#define _MAX_COST 1000111
#define _MAX_FLOW 1000111
template<class Flow = int, class Cost = int>
struct MinCostFlow {
   struct Edge {
      int t;
      Flow f;
      Cost c;
      Edge*next, *rev;
      Edge(int _t, Flow _f, Cost _c, Edge*_next) :
            t(_t), f(_f), c(_c), next(_next) {
      }
   };
   vector<Edge*> E;
   int addV() {
      E.push_back((Edge*) 0);
      return E.size() - 1;
   }
   Edge* makeEdge(int s, int t, Flow f, Cost c) {
      return E[s] = new Edge(t, f, c, E[s]);
   }
   Edge* addEdge(int s, int t, Flow f, Cost c) {
      Edge*e1 = makeEdge(s, t, f, c);
      Edge*e2 = makeEdge(t, s, 0, -c);
      e1->rev = e2, e2->rev = e1;
      return e1;
   }
   pair<Flow, Cost> minCostFlow(int vs, int vt) {
   //flow,cost
      int n = E.size();
      Flow flow = 0;
      Cost cost = 0;
      const Cost MAX_COST = _MAX_COST;
      const Flow MAX_FLOW = _MAX_FLOW;
      for (;;) {
         vector<Cost> dist(n, MAX_COST);
         vector<Flow> am(n, 0);
         vector<Edge*> prev(n);
         vector<bool> inQ(n, false);
         queue<int> que;
         dist[vs] = 0;
         am[vs] = MAX_FLOW;
         que.push(vs);
         inQ[vs] = true;
         while (!que.empty()) {
            int u = que.front();
            Cost c = dist[u];
            que.pop();
            inQ[u] = false;
            for (Edge*e = E[u]; e; e = e->next)
               if (e->f > 0) {
                  Cost nc = c + e->c;
                  if (nc < dist[e->t]) {
                     dist[e->t] = nc;
                     prev[e->t] = e;
                     am[e->t] = min(am[u], e->f);
                     if (!inQ[e->t]) {
                        que.push(e->t);
                        inQ[e->t] = true;
                     }
                  }
               }
         }
         if (dist[vt] == MAX_COST)
            break;
         Flow by = am[vt];
         int u = vt;
         flow += by;
         cost += by * dist[vt];
         while (u != vs) {
            Edge*e = prev[u];
            e->f -= by;
            e->rev->f += by;
            u = e->rev->t;
         }
      }
      return make_pair(flow, cost);
   }
};
```

2.19 Гаусс

```
int gauss (vector<vector<double>> a,
           vector<double> & ans) {
    int n = (int) a.size();
    int m = (int) a[0].size() - 1;
    vector<int> where (m, -1);
    for (int col=0, row=0; col<m && row<n; ++col) {
        int sel = row;
        for (int i=row; i<n; ++i)
            if (abs (a[i][col]) > abs (a[sel][col]))
                sel = i;
        if (abs (a[sel][col]) < EPS)
            continue;
        for (int i=col; i<=m; ++i)
            swap (a[sel][i], a[row][i]);
        where[col] = row;
        for (int i=0; i<n; ++i)
            if (i != row) {
                double c =
                  a[i][col] / a[row][col];
                for (int j=col; j<=m; ++j)
                    a[i][j] -= a[row][j] * c;
            }
        ++row;
    }
    ans.assign (m, 0);
    for (int i=0; i<m; ++i)
        if (where[i] != -1)
            ans[i] =
            a[where[i]][m] / a[where[i]][i];
    for (int i=0; i<n; ++i) {
        double sum = 0;
        for (int j=0; j<m; ++j)
            sum += ans[j] * a[i][j];
        if (abs (sum - a[i][m]) > EPS)
            return 0;
    }
    for (int i=0; i<m; ++i)
        if (where[i] == -1)
            return INF;
    return 1;
}
```

Бинарный

```
int gauss (vector < bitset<N> > a, int n, int m,
           bitset<N> & ans) {
    vector<int> where (m, -1);
    for (int col=0,row=0;col<m&&row<n;++col){
        for (int i=row; i<n; ++i)
            if (a[i][col]) {
                swap (a[i], a[row]);
                break;
            }
        if (! a[row][col])
            continue;
        where[col] = row;
        for (int i=0; i<n; ++i)
            if (i != row && a[i][col])
                a[i] ^= a[row];
        ++row;
    }
}
```

2.20 Atomic Kuhn

```
const int N = 300005;
int curcol, used[N], covered[N], match[N];
vector<int> g[N];
bool dfs(int v) {
    used[v] = curcol;
    for (int u : g[v])
        if (match[u] == -1) {
            covered[v] = 1;
            match[u] = v;
            return 1;
        }
    for (int u : g[v])
        if (used[match[u]] != curcol
            && dfs(match[u])) {
            covered[v] = 1;
            match[u] = v;
            return 1;
        }
    return 0;
}
void kuhn(int n)
{
    curcol = 0;
for (int i = 0; i < n; i++)
    {
        used[i] = 0, match[i] = -1;
        covered[i] = 0;
    }
    while (true) {
        curcol++;
        bool ch = 0;
        for (int i = 0; i < n; i++)
            if (!covered[i] &&
                used[i] != curcol)
                ch |= dfs(i);
        if (!ch)
            break;
    }
}
```

2.21 Обратный по любому модулю

```
ll rev(ll a, ll m) // a, m > 0 {
    if (a == 1)
        return 1;
    return (1LL - rev(m % a, a) * m) / a + m;
}
int gcd(int a, int b, int &x, int &y) {
if (a == 0) {
x = 0; y = 1;
return b;
}
int newx = 0, newy = 0;
int d = gcd(b % a, a, newx, newy);
x = newy - (b / a) * newx;
y = newx;
return d;
```

```
}
```

2.22 Произведение по модулю, вылезающее за границы long long

```
ll mul(ll a, ll b, ll m) // m <= 8e18
    ll k = ((ld)a * b) / m;
    ll r = a * b - k * m;
    while (r < 0) r += m;
    while (r >= m) r -= m;
    return r;
}
```

2.23 Миллер-Рабин

```
bool check_prime(ll n) {
    ll m = n - 1, k = 0;
    while (m % 2 == 0) {
        m /= 2;
        k++;
    }
    ll S = 50; // error_prob <= 1/4^S
    for (int i = 0; i < S; i++) {
        ll a = random(1, n - 1);
        ll x = binpow(a, m, n);
        for (int j = 0; j < k; j++) {
            ll y = x * x % n;
            if (y == 1 && x != 1 && x != n - 1)
                return false;
            x = y;
        }
        if (x != 1)
            return false;
    }
    return true;
}
```

2.24 Венгерка

```
// graph matrix - a[][]
// 1-indexation
// p - matching
vector<int> u (n+1), v (m+1);
vector<int> p (m+1), way (m+1);
for (int i=1; i<=n; ++i) {
    p[0] = i;
    int j0 = 0;
    vector<int> minv (m+1, INF);
    vector<char> used (m+1, false);
    do {
        used[j0] = true;
        int i0 = p[j0],  delta = INF,  j1;
        for (int j=1; j<=m; ++j)
            if (!used[j]) {
                int cur = a[i0][j]-u[i0]-v[j];
                if (cur < minv[j]) {
                    minv[j] = cur;
                    way[j] = j0;
                }
                if (minv[j] < delta) {
                    delta = minv[j];
                    j1 = j;
                }
            }
        for (int j=0; j<=m; ++j)
            if (used[j]) {
                u[p[j]] += delta;
                v[j] -= delta;
            }
            else
                minv[j] -= delta;
        j0 = j1;
    } while (p[j0] != 0);
    do {
        int j1 = way[j0];
        p[j0] = p[j1];
        j0 = j1;
    } while (j0);
}
```

2.25 Мосты

```
void dfs(int v, int p = -1) {
    used[v] = true;
    tin[v] = fup[v] = timer++;
    for (int i = 0; i < sz(g[v]); i++) {
        int to = g[v][i];
        if (to == p)  continue;
        if (used[to])
            fup[v] = min(fup[v], tin[to]);
        else {
            dfs(to, v);
            fup[v] = min(fup[v], fup[to]);
            if (fup[to] > tin[v])
                IS_BRIDGE(v, to);
        }
    }
}
```

2.26 Точки сочленения

```
void dfs(int v, int p = -1) {
    used[v] = true;
    tin[v] = fup[v] = timer++;
    int children = 0;
    for (int i = 0; i < sz(g[v]); i++) {
        int to = g[v][i];
        if (to == p)  continue;
        if (used[to])
            fup[v] = min(fup[v], tin[to]);
        else {
            dfs(to, v);
            fup[v] = min(fup[v], fup[to]);
            if (fup[to] >= tin[v] && p != -1)
                IS_CUTPOINT(v);
            ++children;
        }
    }
    if (p == -1 && children > 1)
        IS_CUTPOINT(v);
}
```

2.27 Z-функция

```
vector<int> z_function(string s) {
    int n = sz(s);
    vector<int> z(n, 0);
    for (int i = 1, l = 0, r = 0; i < n; i++) {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n &&
               s[z[i]] == s[i + z[i]])
            z[i]++;
        if (i + z[i] - 1 > r)
            l = i,  r = i + z[i] - 1;
    }
    return z;
}
```

2.28 Префикс-функция

```
vector<int> prefix_function(string s) {
    int n = sz(s);
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 && s[i] != s[j])
            j = pi[j - 1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
    return pi;
}
```

2.29 Алгоритм Манакера

Нечетные:

```
vector<int> d1(n);
int l = 0, r = -1;
for (int i = 0; i < n; i++) {
    int k=(i>r?0:min(d1[l+r-i],r-i))+1;
    while (i + k < n && i - k >= 0 &&
           s[i + k] == s[i - k])
        k++;
    d1[i] = k--;
    if (i + k > r)
        l = i - k, r = i + k;
}
```

Четные:

```
vector<int> d_2(n);
l = 0, r = -1;
for (int i = 0; i < n; i++) {
    int k=(i>r?0:min(d2[l+r-i+1],r-i+1))+1;
    while (i + k - 1 < n && i - k >= 0 &&
           s[i + k - 1] == s[i - k])
        k++;
    d2[i] = --k;
    if (i + k - 1 > r)
        l = i - k,  r = i + k - 1;
}
```

2.30 Укконен

```
const int N=1123456,INF=1e9 + 7;
string a;
int t[N][26],l[N],r[N],p[N],s[N],tv,tp,ts,la;
void ukkadd (int c) {
    suff:;
    if (r[tv]<tp) {
        if (t[tv][c]==-1) {
            t[tv][c]=ts;  l[ts]=la;
            p[ts++]=tv;  tv=s[tv];
            tp=r[tv]+1;  goto suff;
        }
        tv=t[tv][c]; tp=l[tv];
    }
    if (tp==-1 || c==a[tp]-'a') tp++; else {
        l[ts+1]=la;  p[ts+1]=ts;
        l[ts]=l[tv];  r[ts]=tp-1;  p[ts]=p[tv];
        t[ts][c]=ts+1;  t[ts][a[tp]-'a']=tv;
        l[tv]=tp;  p[tv]=ts;
        t[p[ts]][a[l[ts]]-'a']=ts;  ts+=2;
        tv=s[p[ts-2]];  tp=l[ts-2];
        while (tp<=r[ts-2]) {
            tv=t[tv][a[tp]-'a'];  tp+=r[tv]-l[tv]+1;
        }
        if (tp==r[ts-2]+1)
            s[ts-2]=tv;
        else
            s[ts-2]=ts;
        tp=r[tv]-(tp-r[ts-2])+2;  goto suff;
    }
}
void build() {
    ts=2; tv=0; tp=0; fill(r,r+N,sz(a)-1);
    s[0]=1; l[0]=-1; r[0]=-1; l[1]=-1; r[1]=-1;
    memset (t, -1, sizeof t); fill(t[1],t[1]+26,0);
    for (la=0; la<sz(a); ++la)
        ukkadd (a[la]-'a');
}
```

2.31 Фенвик

```
int sum (int r)
{
    int result = 0;
    for (; r >= 0; r = (r & (r+1)) - 1)
        result += t[r];
    return result;
}
void inc (int i, int delta)
{
    for (; i < n; i = (i | (i+1)))
        t[i] += delta;
}
```

2.32 OR-XOR-AND-свертки

```
vector<ll> fib, cnt, cntab, cntc, cntde;
ll revers(ll x) {
    return binpow(x % MOD, MOD - 2);
}
void adamar(vector<ll> &a, int l, int r) {
```

```
    if (l + 1 >= r) return;
    int mid = (r + l) / 2;
    adamar(a, l, mid);
    adamar(a, mid, r);
    for (int i = l; i < mid; i++) {
        ll u = a[i], v = a[i + mid - l];
        a[i] = (u + v) % MOD;
        a[i + mid - l] = (u - v) % MOD;
    }
}
void andamar(vector<ll> &a, int l, int r) {
    if (l + 1 >= r) return;
    int mid = (r + l) / 2;
    andamar(a, l, mid);
    andamar(a, mid, r);
    for (int i = l; i < mid; i++)
        a[i] = (a[i] + a[i + mid - l]) % MOD;
}
void revandamar(vector<ll> &a, int l, int r) {
    if (l + 1 >= r) return;
    int mid = (r + l) / 2;
    revandamar(a, l, mid);
    revandamar(a, mid, r);
    for (int i = l; i < mid; i++)
        a[i] = (a[i] - a[i + mid - l]) % MOD;
}
void ordamar(vector<ll> &a, int l, int r) {
    if (l + 1 >= r) return;
    int mid = (r + l) / 2;
    ordamar(a, l, mid);
    ordamar(a, mid, r);
    for (int i = l; i < mid; i++)
    a[i + mid - l] = (a[i] + a[i + mid - l]) % MOD;
}
void revordamar(vector<ll> &a, int l, int r) {
    if (l + 1 >= r) return;
    int mid = (r + l) / 2;
    revordamar(a, l, mid);
    revordamar(a, mid, r);
    for (int i = l; i < mid; i++)
    a[i + mid - l] = (-a[i] + a[i + mid - l]) % MOD;
}
vector<ll> xormult(vector<ll> a, vector<ll> b) {
    adamar(a, 0, a.size());
    adamar(b, 0, b.size());
    for (int i = 0; i < a.size(); i++)
        a[i] = (a[i] * b[i]) % MOD;
    adamar(a, 0, a.size());
    ll revn = revers(a.size());
    for (int i = 0; i < a.size(); i++)
        a[i] = (a[i] * revn) % MOD;
    return a;
}
vector<ll> ormult(vector<ll> a, vector<ll> b) {
    ordamar(a, 0, a.size());
    ordamar(b, 0, b.size());
    for (int i = 0; i < a.size(); i++)
        a[i] = (a[i] * b[i]) % MOD;
    revordamar(a, 0, a.size());
    return a;
}
vector<ll> andmult(vector<ll> a, vector<ll> b) {
    andamar(a, 0, a.size());
    andamar(b, 0, b.size());
    for (int i = 0; i < a.size(); i++)
        a[i] = (a[i] * b[i]) % MOD;
    revandamar(a, 0, a.size());
    return a;
}
int main() {
    int n;
    cin >> n;
    // assign(N, 0) for
    // fib, cnt, cntab
    // cntc, cnde
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        cnt[x]++;
    }
    fib[0] = 0, fib[1] = 1;
    for (int i = 2; i < N; ++i)
        fib[i] = (fib[i - 1] + fib[i - 2]) % MOD;
    for (int i = 0; i < N; i++) {
    for (int j = i; j > 0; j = ((j - 1) & i))
    cntab[i] = (cntab[i] + cnt[j] * cnt[i ^ j]) % MOD;
    cntab[i] = (cntab[i] + cnt[0] * cnt[i]) % MOD;
    cntab[i] %= MOD;
    }
    cntc = cnt;
    /*ordamar(cntab, 0, N);
    for (int i = 0; i < N; i++)
        cntab[i] = (cntab[i] * cntab[i]) % MOD;
    revordamar(cntab, 0, N);
    adamar(cnt, 0, N);
    for (int i = 0; i < N; ++i)
        cntde[i] = (cnt[i] * cnt[i]) % MOD;
    adamar(cntde, 0, N);
    for (int i = 0; i < N; ++i)
        cntde[i] = (cntde[i] * revers(N)) % MOD;*/
    cntde = xormult(cnt, cnt);
    for (int i = 0; i < N; i++) {
        cntab[i] = (cntab[i] * fib[i]) % MOD;
        cntc[i] = (cntc[i] * fib[i]) % MOD;
        cntde[i] = (cntde[i] * fib[i]) % MOD;
    }
   /* andamar(cntab, 0, N);
    andamar(cntc, 0, N);
    andamar(cntde, 0, N);
    for (int i = 0; i < N; ++i)
    cntc[i] = cntc[i] * cntab[i] % MOD
              * cntde[i] % MOD;
    revandamar(cntc, 0, N);*/
    cntc = andmult(andmult(cntab, cntc), cntde);
    ll ans = 0;
    for (int i = 0; i < L; ++i)
        ans = (ans + cntc[(1 << i)]) % MOD;
    cout << (ans % MOD + MOD) % MOD;
    return 0;
}
```

2.33 Суфмас

```
const char C = 'a' - 1;
// before first letter // change
const char maxchar = 'z'; // change
vector<int> suffarray(string s)
// without $ at the end
{
    vector<int> p, c, pn, cn, cnt;
    int n = (int)s.size();
    c.assign(n, 0);
    for (int i = 0; i < n; i++)
        c[i] = s[i] - C;
    for (int j = 0; j <= (maxchar - C); j++)
        for (int i = 0; i < n; i++)
            if (c[i] == j)
                p.push_back(i);
    int maxc = c[p.back()];
    pn.resize(n);
    for (int k = 0; (1 << k) <= 2 * n; k++) {
        for (int i = 0; i < n; i++)
            pn[i] =
            ((p[i] - (1 << k)) % n + n) % n;
        cnt.assign(maxc + 3, 0);
        for (int i = 0; i < n; i++)
            cnt[c[i] + 1]++;
        for (int i = 1; i <= maxc + 2; i++)
            cnt[i] += cnt[i - 1];
        for (int i = 0; i < n; i++)
            p[cnt[c[pn[i]]]++] = pn[i];
        cn.assign(n, 0);
        cn[p[0]] = 1;
        for (int i = 1; i < n; i++)
            if (c[p[i]] == c[p[i - 1]] &&
            c[(p[i] + (1 << k)) % n] ==
            c[(p[i - 1] + (1 << k)) % n])
                cn[p[i]] = cn[p[i - 1]];
            else
                cn[p[i]] = cn[p[i - 1]] + 1;
        maxc = cn[p.back()];
        c = cn;
    }
    return p;
}
vector<int> findlcp(string s, vector<int> p) {
    vector<int> lcp, mem;
    int n = (int)s.size();
    mem.resize(n);
    for (int i = 0; i < n; i++)
        mem[p[i]] = i;
    lcp.assign(n, 0);
    for (int i = 0; i < n; i++) {
        if (i)
            lcp[mem[i]] = max(lcp[mem[i - 1]] - 1, 0);
        if (mem[i] == n - 1)
            continue;
        while (max(i, p[mem[i] + 1]) +
               lcp[mem[i]] < n
               && s[i + lcp[mem[i]]] ==
               s[p[mem[i] + 1] + lcp[mem[i]]])
            lcp[mem[i]]++;
    }
    return lcp;
}
```

2.34 ConvexHullTrick

```
typedef long long integer;
typedef long double ld;
struct Line {
    integer k, b;
    Line():
        k(0), b(0) {}
    Line(integer k, integer b):
        k(k), b(b) {}
    ld operator()(ld x) {
        return x * (ld)k + (ld)b;
    }
};
const integer INF = 2e18; // change
struct CHT {
    vector<Line> lines;
    bool mini; // cht on minimum
    ld f(Line l1, Line l2) {
        return (ld)(l1.b - l2.b) / (ld)(l2.k - l1.k);
    }
    void addLine(integer k, integer b) {
        if (!mini) {
            k = -k;
            b = -b;
        }
        Line l(k, b);
        while (lines.size() > 1) {
            if (lines.back().k == k) {
                if (lines.back().b > b)
                    lines.pop_back();
                else
                    break;
                continue;
            }
            ld x1 = f(lines.back(), l);
            ld x2 = f(lines.back(),
                      lines[lines.size() - 2]);
            if (x1 > x2)
                break;
            lines.pop_back();
        }
        if (!lines.size() || lines.back().k != k)
            lines.push_back(l);
    }
    CHT(vector<pair<integer, integer> > v,
    bool ok = 1) // change
    {
        mini = ok;
        lines.clear();
        for (int i = 0; i < v.size(); i++)
            addLine(v[i].first, v[i].second);
    }
    integer getmin(integer x)
    //find of integer!
    {
        if (!lines.size())
            return (mini ? INF : -INF);
```

```
        int l = 0, r = lines.size();
        while (r - l > 1) {
            int mid = (r + l) / 2;
            if (f(lines[mid], lines[mid - 1])
                <= (ld)x)
                l = mid;
            else
                r = mid;
        }
        integer ans = lines[l].k * x +
                      lines[l].b;
        return (mini ? ans : -ans);
    }
};
```

2.35 DCP

```
const int N = 300010;
map<pair<int, int>, int> q;
vector<int> quer;
int ans[N], pr[N], sz[N], curans;
struct edge {
    int u, v, l, r;
    edge(): u(0), v(0), l(0), r(0) {}
    edge(int u, int v, int l, int r):
    u(u), v(v), l(l), r(r) {}
};
vector<pair<pair<int, int>,pair<int,int>>>qq;
vector<edge> edges;
void add(int u, int v, int l, int r) {
    qq.push_back({{u, v}, {l, r}});
}
vector<int> st;
int findset(int v) {
    return ((v == pr[v]) ? v : findset(pr[v]));
}
int unionsets(int v, int u) {
    v = findset(v);
    u = findset(u);
    if (v == u) return 0;
    if (sz[v] < sz[u]) swap(v, u);
    st.push_back(u);
    sz[v] += sz[u];
    pr[u] = v;
    curans--;
    return 1;
}
void remove() {
    int u = st.back();
    st.pop_back();
    sz[pr[u]] -= sz[u];
    pr[u] = u;
    curans++;
}
void divide(int l, int r, vector<edge> edges) {
    if (r < l)
        return;
    if (l == r) {
        ans[l] = curans;
        return;
    }
    int mid = (r + l) / 2;
    vector<edge> edges1;
    int num = 0;
    for (auto i : edges) {
        if (i.l > mid || i.r < l)
            continue;
        if (i.r >= mid && i.l <= l)
            num += unionsets(i.u, i.v);
        else
            edges1.push_back(i);
    }
    divide(l, mid, edges1);
    for (int i = 0; i < num; i++)
        remove();
    num = 0;
    edges1.clear();
    for (auto i : edges) {
        if (i.l > r || i.r <= mid)
            continue;
        if (i.l <= mid + 1 && i.r >= r)
            num += unionsets(i.u, i.v);
        else
            edges1.push_back(i);
    }
    divide(mid + 1, r, edges1);
    for (int i = 0; i < num; i++) remove();
}
int main() {
    for (int i = 0; i < k; i++) {
        char c;
        cin >> c;
        if (c == '?') quer.push_back(i);
        else {
            int u, v;
            cin >> u >> v;
            if (u > v) swap(u, v);
            if (c == '+') q[make_pair(u, v)] = i;
            else {
                add(u, v, q[make_pair(u, v)], i);
                q.erase(make_pair(u, v));
            }
        }
    }
    for (auto it : q) {
        int u = it.first.first;
        int v = it.first.second;
        add(u, v, it.second, k);
    }
    add(1, 2, k + 1, k + 2);
    for (int i = 0; i < N; i++)
        pr[i] = i, sz[i] = 1;
    sort(qq.begin(), qq.end());
    for (auto i : qq)
        edges.push_back(
        edge(i.first.first, i.first.second,
        i.second.first, i.second.second)
        );
    curans = n;
    divide(0, k + 2, edges);
```

2.36 Берликамп

```
#define pb push_back
#define SZ 233333
const int MOD=1e9+7; //or any prime
ll qp(ll a,ll b)
{
ll x=1; a%=MOD;
while(b)
{
if(b&1) x=x*a%MOD;
a=a*a%MOD; b>>=1;
}
return x;
}
namespace linear_seq {
inline vector<int> BM(vector<int> x)
{
//ls: (shortest) relation sequence
    // (after filling zeroes) so far
//cur: current relation sequence
vector<int> ls,cur;
//lf: the position of ls (t')
//ld: delta of ls (v')
int lf,ld;
for(int i=0;i<int(x.size());++i)
{
ll t=0;
//evaluate at position i
for(int j=0;j<int(cur.size());++j)
t=(t+x[i-j-1]*(ll)cur[j])%MOD;
if((t-x[i])%MOD==0) continue; //good so far
//first non-zero position
if(!cur.size())
{
cur.resize(i+1);
lf=i; ld=(t-x[i])%MOD;
continue;
}
//cur=cur-c/ld*(x[i]-t)
ll k=-(x[i]-t)*qp(ld,MOD-2)%MOD/*1/ld*/;
vector<int> c(i-lf-1); //add zeroes in front
c.pb(k);
for(int j=0;j<int(ls.size());++j)
c.pb(-ls[j]*k%MOD);
if(c.size()<cur.size()) c.resize(cur.size());
for(int j=0;j<int(cur.size());++j)
c[j]=(c[j]+cur[j])%MOD;
//if cur is better than ls, change ls to cur
if(i-lf+(int)ls.size()>=(int)cur.size())
ls=cur,lf=i,ld=(t-x[i])%MOD;
cur=c;
}
for(int i=0;i<int(cur.size());++i)
cur[i]=(cur[i]%MOD+MOD)%MOD;
return cur;
}
int m; //length of recurrence
//a: first terms, h: relation
ll a[SZ],h[SZ],t_[SZ],s[SZ],t[SZ];
//calculate p*q mod f
inline void mull(ll*p,ll*q)
{
for(int i=0;i<m+m;++i) t_[i]=0;
for(int i=0;i<m;++i) if(p[i])
for(int j=0;j<m;++j)
t_[i+j]=(t_[i+j]+p[i]*q[j])%MOD;
for(int i=m+m-1;i>=m;--i)
        if(t_[i])
//miuns t_[i]x^{i-m}(x^m-\sum_{j=0}^{m-1}
        // x^{m-j-1}h_j)
for(int j=m-1;~j;--j)
t_[i-j-1]=(t_[i-j-1]+t_[i]*h[j])%MOD;
for(int i=0;i<m;++i) p[i]=t_[i];
}
inline ll calc(ll K)
{
for(int i=m;~i;--i)
s[i]=t[i]=0;
//init
s[0]=1; if(m!=1) t[1]=1; else t[0]=h[0];
//binary-exponentiation
while(K)
{
if(K&1) mull(s,t);
mull(t,t); K>>=1;
}
ll su=0;
for(int i=0;i<m;++i) su=(su+s[i]*a[i])%MOD;
return (su%MOD+MOD)%MOD;
}
inline int work(vector<int> x,ll n)
{
if(n<int(x.size())) return x[n];
vector<int> v=BM(x); m=v.size(); if(!m) return 0;
for(int i=0;i<m;++i) h[i]=v[i],a[i]=x[i];
return calc(n);
}
}
using linear_seq::work;
int main()
{
    vector<int> x = {1, 2, 4, 8, 16};
    for (int i = 0; i < 10; i++)
        cout << work(x, i) << endl;
}
```