



华南理工大学
South China University of Technology

ICPC TEMPLATE



BY

albertxwz

School of Computer Science & Engineering,
South China University of Technology

Published in
Dec 2021

This template is a supplementary version.

Contents

1	Standard Solution Template	1	B Theorem	16
1.1	support bits/stdc++.h	1	B.1 Lucas' Theorem	16
1.2	unsupport bits/stdc++.h	1	B.2 Betty's Theorem	16
1.3	Python Template	1	B.3 Bernoulli Number	17
2	Graph	2	C C++ STL: set	17
2.1	Network Flow	2	C.1 Basic Method	17
2.1.1	Dinic	2	C.2 Advanced Method	17
2.1.2	Edmonds-Karp	2		
2.2	DSU on Tree	3		
3	Mathematics	4		
3.1	Number Theory	4		
3.1.1	Linear Inverse Modulo	4		
3.1.2	Quick Power	4		
3.1.3	Baby-Step-Giant-Step Algorithm	4		
3.1.4	Möbius Inversion	4		
3.1.5	Dujiao Sieve	4		
3.1.6	Chinese Remainder Theory . . .	5		
3.1.7	Floor Sum	5		
3.1.8	Min_25 Sieve	6		
3.1.9	$\pi(x)$	6		
3.1.10	Lucas' Theorem	6		
3.2	Karatsuba Multiply	6		
3.3	Fast Fourier Transform	7		
3.4	Number Theory Transform	8		
3.5	Lagrange Insertion Value Method	9		
4	Data Structure	10		
4.1	Treap	10		
4.2	Splay Tree	10		
4.3	Two-dimensional Segment Tree	10		
4.4	Mo's Algorithm	10		
5	String	12		
5.1	KMP	12		
5.2	Trie	12		
6	Computational Geometry	13		
6.1	2D Template	13		
6.2	Smallest Covering Circle	13		
6.3	Half Plane Intersection	13		
A	Env Conf	15		

1 Standard Solution Template

1.1 support bits/stdc++.h

```
#include <bits/stdc++.h>
#define lc (o<<1)
#define rc ((o<<1)|1)
#define PB push_back
#define MK make_pair
using namespace std;
#define DebugP(x) cout << "Line" << __LINE__
    << " " << #x << "=" << x << endl

const int maxn = 3000 + 5;
const int modu = 998244353; // 1e9 + 7
const int inf = 0x3f3f3f3f;
const double eps = 1e-5;
const int dx[4] = {1, 0, -1, 0};
const int dy[4] = {0, 1, 0, -1};

typedef long long LL;

void read(LL &x) {
    x=0;int f=0;char ch=getchar();
    while(ch<'0' || ch>'9') {f|=(ch=='-');ch=
        getchar();}
    while(ch>='0' && ch<='9'){x=(x<<1)+(x<<3)+(
        ch^48);ch=getchar();}
    x=f?-x:x;
    return;
}

void read(int &x) { LL y; read(y); x = (int)y
    ; }
void read(char &ch) { char s[3]; scanf("%s",
    s); ch = s[0]; }
void read(char *s) { scanf("%s", s); }
template<class T, class ...U> void read(T &x,
    U& ... u) { read(x); read(u...); }

int main() {
    // freopen("my.txt", "w", stdout);
    ios::sync_with_stdio(0);
    cin.tie(0);
    return 0;
}
```

1.2 unsupport bits/stdc++.h

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <cmath>
#include <iterator>
#include <set>
#include <map>
#include <queue>
#include <stdlib.h>
#include <string>

using namespace std;

int main() {
    // freopen("my.txt", "w", stdout);
    return 0;
}
```

1.3 Python Template

```
for T in range(0, int(input())): #T组数据
    N= int(input())
    n,m= map(int, input().split())
    s= input()
    s=[ int(x) for x in
        input().split()] #一行输入的数组
    a[1:]=[ int(x) for x in
        input().split()] #从下标1开始读入一行
    for i in range(0, len(s)):
        a,b= map(int, input().split())

while True: #未知多组数据
    try:
        #n,m=map(int,input().split())
        #print(n+m,end="\n")
    except EOFError: #捕获到异常
        break
```

2 Graph

2.1 Network Flow

2.1.1 Dinic

```
struct Edge {
    int from, to, cap, flow;
    Edge(int from=0, int to=0, int cap=0, int
        flow=0):
        from(from), to(to), cap(cap), flow(
            flow) {}
};

vector<Edge> edges;
vector<int> g[maxn];
int d[maxn], cur[maxn];

void addedge(int u, int v, int cap) {
    edges.push_back(Edge(u, v, cap));
    g[u].push_back(edges.size()-1);
    edges.push_back(Edge(v, u, 0));
    g[v].push_back(edges.size()-1);
}

/*
 * 时间复杂度:  $O(n^2*m)$ 
 * 对于特殊的图, 所有容量为1的:  $O(\min(n^{0.67}, m^{0.5})*m)$ 
 * 二分图最大匹配:  $O(n^{0.5}*m)$ 
 */

bool BFS(int s, int t) {
    memset(d, -1, sizeof(d));
    queue<int> Q;
    Q.push(s);
    d[s] = 0;
    while (!Q.empty()) {
        int x = Q.front(); Q.pop();
        for (int i = 0; i < g[x].size(); ++i) {
            Edge &e = edges[g[x][i]];
            if (d[e.to] == -1 && e.cap > e.
                flow) {
                d[e.to] = d[x] + 1;
                Q.push(e.to);
            }
        }
    }
}
```

```
    }
}

return d[t] > -1;
}

int DFS(int x, int a, int t) {
    if (x == t || a == 0) return a;
    int flow = 0, f;
    for (int &i = cur[x]; i < g[x].size(); ++
        i) {
        Edge &e = edges[g[x][i]];
        if (d[x] + 1 == d[e.to] && (f = DFS(e
            .to, min(a, e.cap-e.flow), t)) >
            0) {
            flow += f;
            e.flow += f;
            edges[g[x][i]^1].flow -= f;
            a -= f;
            if (a == 0) break;
        }
    }
    return flow;
}

int MaxFlow(int s, int t) {
    int res = 0;
    while (BFS(s, t)) {
        for (int i = 0; i <= n; ++i) cur[i] =
            0;
        res += DFS(s, inf, t);
    }
    return res;
}
```

2.1.2 Edmonds-Karp

```
queue<int> Q;
int imp[maxn], p[maxn];

// 不断寻找增广路增广

int MaxFlow(int s, int t) {
    int res = 0;
    for (;;) {
        memset(imp, 0, sizeof(imp));
        while (!Q.empty()) Q.pop();
        Q.push(s);
```

```

    imp[s] = inf;
    while (!Q.empty()) {
        int x = Q.front(); Q.pop();
        for (int i = 0; i < g[x].size();
            ++i) {
            Edge &e = edges[g[x][i]];
            if (!imp[e.to] && e.cap > e.
                flow) {
                p[e.to] = g[x][i];
                imp[e.to] = min(imp[x], e.
                    cap-e.flow);
                Q.push(e.to);
            }
        }
        if (imp[t]) break;
    }
    if (!imp[t]) break;
    for (int u = t; u != s; u = edges[p[u]
        ].from) {
        edges[p[u]].flow += imp[t];
        edges[p[u]^1].flow -= imp[t];
    }
    res += imp[t];
}
return res;
}

```

2.2 DSU on Tree

3 Mathematics

3.1 Number Theory

3.1.1 Linear Inverse Modulo

```
const int maxn = 2e5 + 5;
const int modu = 1e9 + 7;
long long inv[maxn]; // k在模modu的意义下的逆
元是inv[k]
inv[1] = 1;
for (int i = 2; i < maxn; ++i)
    inv[i] = (modu - (modu/i)) * inv[modu%i] %
    modu;
```

3.1.2 Quick Power

```
LL powmod(LL a, LL b, LL modu) {
    LL res = 1;
    for (a %= modu; b; b >>= 1, a = a*a%modu)
        if (b&1) res = res * a % modu;
    return res;
}
```

3.1.3 Baby-Step-Giant-Step Algorithm

$$a^x \equiv b \pmod{n}$$

```
// no solution = -1
int log_mod(int a, int b, int n) {
    LL m, v, e = 1, i;
    m = S; // S = int(sqrt(n) + 0.5)
    v = powmod(powmod(a, m), n-2);
    unordered_map<int, int> x;
    x[1] = 0;
    for (i = 1; i < m; ++i) {
        e = e*a %n;
        if (!x.count(e)) x[e] = i;
    }
    for (i = 0; i < m; ++i) {
        if (x.count(b)) return (LL)i*m + x[b];
        b = (LL)b*v%n;
    }
    return -1;
}
```

3.1.4 Möbius Inversion

3.1.5 Dujiao Sieve

常见积性函数:

$$\epsilon(n) = [n = 1]$$

$$I(n) = 1$$

$$id(n) = n$$

$$d(x) = \sum_{i|n} 1$$

$$\sigma(x) = \sum_{i|n} i$$

$$\phi(i) = \sum_{i|n} [gcd(x, i) = 1]$$

$$\mu(x) = \begin{cases} 1 & x = 1 \\ (-1)^k & \prod_{i=1}^k q_i = 1 \\ 0 & \max\{q_i\} > 1 \end{cases}$$

狄利克雷卷积: 设 f, g 是两个数论函数, 则有 $(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$

常见性质:

$$1. \mu * I = \epsilon$$

$$2. \phi * I = id$$

$$3. \mu * id = \phi$$

最后要有形如:

$$g(1)S(1) = \sum_{i=1}^n (f * g)(i) - \sum_{i=2}^n g(i)S(\lfloor \frac{n}{i} \rfloor)$$

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <map>
using namespace std;
const int maxn = 2000010;
typedef long long ll;
ll T, n, pri[maxn], cur, mu[maxn], sum_mu[
    maxn];
bool vis[maxn];
map<ll, ll> mp_mu;
ll S_mu(ll x) {
    if (x < maxn) return sum_mu[x];
    if (mp_mu[x]) return mp_mu[x];
    ll ret = 1ll;
    for (ll i = 2, j; i <= x; i = j + 1) {
        j = x / (x / i);
```

```

    ret -= S_mu(x / i) * (j - i + 1);
}
return mp_mu[x] = ret;
}
ll S_phi(ll x) {
    ll ret = 0ll;
    for (ll i = 1, j; i <= x; i = j + 1) {
        j = x / (x / i);
        ret += (S_mu(j) - S_mu(i - 1)) * (x / i)
            * (x / i);
    }
    return ((ret - 1) >> 1) + 1;
}
int main() {
    scanf("%lld", &T);
    mu[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (!vis[i]) {
            pri[++cur] = i;
            mu[i] = -1;
        }
        for (int j = 1; j <= cur && i * pri[j] <
            maxn; j++) {
            vis[i * pri[j]] = true;
            if (i % pri[j])
                mu[i * pri[j]] = -mu[i];
            else {
                mu[i * pri[j]] = 0;
                break;
            }
        }
    }
    for (int i = 1; i < maxn; i++) sum_mu[i] =
        sum_mu[i - 1] + mu[i];
    while (T--) {
        scanf("%lld", &n);
        printf("%lld %lld\n", S_phi(n), S_mu(n));
    }
    return 0;
}

```

3.1.6 Chinese Remainder Theory

```

// x mod a[i] == b[i] mod a[i]
// gcd(a[i], a[j]) == 1 (i != j)
LL crt(LL *a, LL *b, int n) {
    LL res = 0;

```

```

    LL tota = 1;
    for (int i = 0; i < n; ++i) tota *= a[i];
    for (int i = 0; i < n; ++i) {
        LL m = tota / a[i];
        LL r, tmp;
        exgcd(m, a[i], r, tmp);
        r = (r*a[i]+a[i])%a[i];
        res = (res + b[i]*m%tota*r%tota) %
            tota;
    }
    return res;
}

// extended version
// x mod a[i] == b[i] mod a[i]
// gcd(a[i], a[j]) >= 1
LL excrt(LL *a, LL *b, int n) {
    for (int i = 1; i < n; ++i) {
        LL x, y;
        LL g = exgcd(a[0], a[i], x, y);
        x %= a[i];
        a[0] /= g;
        x = ((__int128)(b[i] - b[0])%a[i]*x%a
            [i] + a[i])%a[i];
        LL lcm = a[0]*a[i];
        b[0] = ((__int128)x*a[0] + b[0])%lcm;
        a[0] = lcm;
    }
    return b[0];
}

```

3.1.7 Floor Sum

```

// sigma(floor(a*i+b/c)) 0 <= i <= n
LL floor_sum(LL a, LL b, LL c, LL n) {
    if (a == 0) return (n+1)*(b/c);
    if (n == 0) return b/c;
    if (n < 0) return 0;
    LL res = (a/c)*n*(n+1)/2 + (n+1)*(b/c);
    a %= c;
    b %= c;
    LL m = (a*n+b)/c;
    return res + n*m - floor_sum(c, c-b-1, a,
        m-1);
}

// atcoder

```



```

using ll = long long;
ll floor_sum(ll n, ll m, ll a, ll b) {
    ll ans = 0;
    if (a >= m) {
        ans += (n - 1) * n * (a / m) / 2;
        a %= m;
    }
    if (b >= m) {
        ans += n * (b / m);
        b %= m;
    }

    ll y_max = (a * n + b) / m, x_max = (
        y_max * m - b);
    if (y_max == 0) return ans;
    ans += (n - (x_max + a - 1) / a) * y_max;
    ans += floor_sum(y_max, a, m, (a - x_max
        % a) % a);
    return ans;
}

```

3.1.8 Min_25 Sieve

3.1.9 $\pi(x)$

$$\pi(x) \sim \frac{x}{\ln x}$$

```

LL n;
vector<LL> p, val;
bool vis[maxn+1];
LL f[maxn+1];
int id1[maxn+1], id2[maxn+1];

inline int getid(LL k) {
    if (k <= n/k) return id1[k];
    return id2[n/k];
}

int main() {
    scanf("%lld", &n);
    memset(vis, 0, sizeof(vis));
    int m = sqrt(n);
    for (LL i = 2; i <= m; ++i) {
        if (!vis[i]) p.PB(i);
        for (int j = 0; j < SZ(p) && i*p[j]
            <= m; ++j) {
            vis[i*p[j]] = 1;
            if (i % p[j] == 0) break;

```

```

        }
    }
    for (LL L = 1, R; L <= n; L = R+1) {
        R = n / (n/L);
        val.PB(R);
        if (R <= n/R) id1[R] = SZ(val)-1;
        else id2[n/R] = SZ(val)-1;
    }
    for (int i = 0; i < SZ(val); ++i) f[i] =
        val[i] - val[i]/2 - (val[i] == 1);
    for (int i = 1; i < SZ(p); ++i)
        for (int j = SZ(val)-1; j >= 0 && val
            [j] >= p[i]*p[i]; --j)
            f[j] += - f[getid(val[j]/p[i])] +
                i;
    printf("%lld\n", f[SZ(val)-1]);
    return 0;
}

```

3.1.10 Lucas' Theorem

```

long long Lucas(long long n, long long m,
    long long p) {
    if (m == 0) return 1;
    return (C(n % p, m % p, p) * Lucas(n / p,
        m / p, p)) % p;
}

```

3.2 Karatsuba Multiply

```

int *karatsuba_polymul(int n, int *a, int *b)
{
    if (n <= 32) {
        // 规模较小时直接计算，避免继续递归带来的效率损失
        int *r = new int[n * 2 + 1]();
        for (int i = 0; i <= n; ++i)
            for (int j = 0; j <= n; ++j) r[i + j]
                += a[i] * b[j];
        return r;
    }

    int m = n / 2 + 1;
    int *r = new int[m * 4 + 1]();
    int *z0, *z1, *z2;

```

```

z0 = karatsuba_polymul(m - 1, a, b);
z2 = karatsuba_polymul(n - m, a + m, b + m);
;

// 计算 z1
// 临时更改, 计算完毕后恢复
for (int i = 0; i + m <= n; ++i) a[i] += a[i + m];
for (int i = 0; i + m <= n; ++i) b[i] += b[i + m];
z1 = karatsuba_polymul(m - 1, a, b);
for (int i = 0; i + m <= n; ++i) a[i] -= a[i + m];
for (int i = 0; i + m <= n; ++i) b[i] -= b[i + m];
for (int i = 0; i <= (m - 1) * 2; ++i) z1[i] -= z0[i];
for (int i = 0; i <= (n - m) * 2; ++i) z1[i] -= z2[i];

// 由 z0、z1、z2 组合获得结果
for (int i = 0; i <= (m - 1) * 2; ++i) r[i] += z0[i];
for (int i = 0; i <= (m - 1) * 2; ++i) r[i + m] += z1[i];
for (int i = 0; i <= (n - m) * 2; ++i) r[i + m * 2] += z2[i];

delete[] z0;
delete[] z1;
delete[] z2;
return r;
}

// 计算a*b=c, 时间复杂度是O(n^1.585)
void karatsuba_mul(int a[], int b[], int c[])
{
    int *r = karatsuba_polymul(LEN - 1, a, b);
    memcpy(c, r, sizeof(int) * LEN);
    for (int i = 0; i < LEN - 1; ++i)
        if (c[i] >= 10) {
            c[i + 1] += c[i] / 10;
            c[i] %= 10;
        }
    delete[] r;
}

```

3.3 Fast Fourier Transform

```

// f是系数数组, 处理完后, f表示:
// rev=1,是点表示法
// rev=-1,除N后是系数
// N=2^n
typedef complex<double> Comp; // 先导入头文件
complex

void DFT(Comp *f, int N, int rev) {
    if (N == 1) return;
    for (int i = 0; i < N; ++i) tmp[i] = f[i];
    for (int i = 0; i < N; ++i)
        if (i%2) f[i/2+N/2] = tmp[i];
        else f[i/2] = tmp[i];
    Comp *g = f, *h = f + N/2;
    DFT(g, N/2, rev); DFT(h, N/2, rev);
    // c[N]=cos(2*pi/N), s[N]=sin(2*pi/N)
    Comp w(c[N], s[N]*rev), cur(1, 0);
    for (int k = 0; k < N/2; ++k) {
        tmp[k] = g[k] + cur*h[k];
        tmp[k+N/2] = g[k] - cur*h[k];
        cur *= w;
    }
    for (int i = 0; i < N; ++i) f[i] = tmp[i];
}

// Important !!!!
//for (N = 1; N < n+m; N <= 1) {
//    s[N] = sin(2*pi/N);
//    c[N] = cos(2*pi/N);
//}
//s[N] = sin(2*pi/N);
//c[N] = cos(2*pi/N);
// no recursion
inline void change(Comp y[], int len) {
    int i, j, k;
    for (int i = 1, j = len / 2; i < len - 1; i++) {
        if (i < j) swap(y[i], y[j]);
        // 交换互为小标反转的元素, i<j 保证交换一次
        // i 做正常的 +1, j 做反转类型的 +1, 始终保持 i 和 j 是反转的
        k = len / 2;
        while (j >= k) {
            j = j - k;

```

```

        k = k / 2;
    }
    if (j < k) j += k;
}
/*
 * 做 FFT
 * len 必须是 2^k 形式
 * on == 1 时是 DFT, on == -1 时是 IDFT
 */
inline void fft(Comp y[], int len, int on) {
    change(y, len);
    for (int h = 2; h <= len; h <= 1) {
        Comp wn(c[h], on*s[h]);
        for (int j = 0; j < len; j += h) {
            Comp w(1, 0);
            for (int k = j; k < j + h / 2; k
                ++){
                Comp u = y[k];
                Comp t = w * y[k + h / 2];
                y[k] = u + t;
                y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if (on == -1) {
        for (int i = 0; i < len; ++i)
            y[i] = Comp(floor(y[i].real()/len
                +0.5), 0);
    }
}

```

3.4 Number Theory Transform

```

// convolution for F(x) * G(x)
/* N = 2^k
 * init: f[i] = ... 0 <= i < N
 *       g[i] = ... 0 <= i < N
 * call :
 * NTT(f, N, 1);
 * NTT(g, N, 1);
 * for (int i = 0; i < N; ++i) f[i] = (LL)f[i]*
 *     g[i];
 * NTT(f, N, -1);
 */

```

```

inline void change(int y[], int len) {
    for (int i = 1, j = len / 2, k; i < len -
        1; i++) {
        if (i < j) swap(y[i], y[j]);
        k = len / 2;
        while (j >= k) {
            j = j - k;
            k = k / 2;
        }
        if (j < k) j += k;
    }
}

// g is the primitive root of modu
inline void NTT(int y[], int len, int on) {
    int g = 3; // modu = 998244353 or
               // 1004535809
    change(y, len);
    for (int h = 2; h <= len; h <= 1) {
        LL wn = powmod(g, (modu-1)/h);
        if (on == -1) wn = powmod(wn, modu-2);
        ;
        for (int j = 0; j < len; j += h) {
            LL w = 1;
            for (int k = j; k < j + h / 2; k
                ++){
                LL u = y[k];
                LL t = (w * 111 * y[k + h /
                    2])%modu;
                y[k] = (u + t)%modu;
                y[k + h / 2] = (u - t + modu)
                    %modu;
                w = w * wn % modu;
            }
        }
    }
    if (on == -1) {
        LL INV = powmod(len, modu-2);
        for (int i = 0; i < len; ++i)
            y[i] = 111 * y[i] * INV % modu;
    }
}

```

3.5 Lagrange Insertion Value Method

$$f(k) = \sum_{i=1}^n y_i \prod_{j \neq i} \frac{k - x_j}{x_i - x_j}$$

```
#include <stdio>

const int maxn = 2010;
using ll = long long;
ll mod = 998244353;
ll n, k, x[maxn], y[maxn], ans, s1, s2;

ll powmod(ll x, ll n) {
    ll ret = 1ll;
    while (n) {
        if (n & 1) ret = ret * x % mod;
        x = x * x % mod;
        n >>= 1;
    }
    return ret;
}

ll inv(ll x) { return powmod(x, mod - 2); }

int main() {
    scanf("%lld%lld", &n, &k);
    for (int i = 1; i <= n; i++) scanf("%lld%lld", x + i, y + i);
    for (int i = 1; i <= n; i++) {
        s1 = y[i] % mod;
        s2 = 1ll;
        for (int j = 1; j <= n; j++)
            if (i != j) s1 = s1 * (k - x[j]) % mod,
                s2 = s2 * (x[i] - x[j]) % mod;
        ans += s1 * inv(s2) % mod;
    }
    printf("%lld\n", (ans % mod + mod) % mod);
    return 0;
}

// Lagrange interpolation O(n^2)
// n is the highest degree of polynomial
// return f(x) where x = n
// x[i] = i-1, x[i] sorted from low to high
LL lagrange(LL n, LL *x, LL *y, int m) {
    LL res = 0;
    for (int i = 0; i < m; ++i) {
        LL tmp = 1;
```

```
for (int j = 0; j < m; ++j)
    if (i != j) {
        tmp = (n - x[j]) % mod * inv
            [abs(x[i] - x[j])] % mod
            * tmp % mod;
    }
    if ((m-i-1)%2) tmp *= -1;
    res = (res + tmp*y[i]%mod) %mod;
}
return res;
}
```

4 Data Structure

4.1 Treap

```

struct Treap
{
    struct Node
    {
        Node *ch[2];
        int r, v, cnt;

        Node()
        {
            *this = NULL;
        }

        Node(int v) : v(v)
        {
            r = rand()/10*rand();
            ch[0] = ch[1] = NULL;
            cnt = 1;
        }

        bool operator < (const Node &rhs)
            const
        {
            return r < rhs.r;
        }

        int cmp(int x) const
        {
            if (x == v) return -1;
            return x > v;
        }
    } *root;

    void rotate(Node *&o, int d)
    {
        Node *k = o->ch[d^1];
        o->ch[d^1] = k->ch[d];
        k->ch[d] = o;
        o = k;
    }

    void insert(Node *&o, int x)
    {

```

```

        if (o == NULL)
        {
            o = new Node(x);
            return;
        }
        int k = o->cmp(x);
        if (k == -1) o->cnt++;
        else
        {
            insert(o->ch[k], x);
            if (o->ch[k]->r > o->r) rotate(o,
                k^1);
        }
    }

    void remove(Node *&o, int x)
    {
        int d = o->cmp(x);
        if (d == -1)
        {
            if (o->cnt == 0 || (--o->cnt) ==
                0)
            {
                if (o->ch[0] == NULL)
                    o = o->ch[1];
                else if (o->ch[1] == NULL)
                    o = o->ch[0];
                else
                {
                    int d1 = o->ch[0]->r > o->
                        ch[1]->r;
                    rotate(o, d1);
                    remove(o->ch[d1], x);
                }
            }
            return;
        }
        remove(o->ch[d], x);
    }

    int finds(Node *o, int x)
    {
        if (o == NULL) return INF;
        int d = o->cmp(x);
        if (d == -1) return 0;
        if (o->ch[d] == NULL) return abs(x-o
            ->v);
    }

```

```

        else return min(abs(x-o->v), finds(o
            ->ch[d], x));
    }

    int findmin()
    {
        Node *k = root;
        while (k->ch[0] != NULL) k = k->ch
            [0];
        return k->v;
    }
}

```

4.2 Splay Tree

4.3 Two-dimensional Segment Tree

4.4 Mo's Algorithm

```

struct Interval {
    int l, r, t, id;
    int k;
    Interval(int l=0, int r=0, int t=0, int
        id=0, int k=0): l(l), r(r), t(t), id(id), k(k) {}
};

struct UpdateOp {
    int p, x;
    UpdateOp(int p=0, int x=0): p(p), x(x) {}
};

int n, q, S, st, ed, qt; // S is the size of
    one block
vector<Interval> intervals;
vector<UpdateOp> up;

bool cmp(Interval A, Interval B) {
    return (A.l-1)/S < (B.l-1)/S ||
        ((A.l-1)/S == (B.l-1)/S && (A.r
            -1)/S < (B.r-1)/S) ||
        ((A.l-1)/S == (B.l-1)/S && (A.r
            -1)/S == (B.r-1)/S && A.t < B
            .t);
    // 没有更新操作时需要修改
}

```

```

void add(int loc) {
}

void del(int loc) {
}

int main() {
    // freopen("input.txt", "r", stdin);
    ios::sync_with_stdio(false);
    cin.tie(0);
    read(n); read(q);
    for (int i = 1; i <= n; ++i) {
        read(a[i]);
        tmpa[i] = a[i];
    }
    for (int i = 0; i < q; ++i) {
        int cmd;
        read(cmd);
        if (cmd == 1) {
            int l, r, k;
            read(l); read(r); read(k);
            intervals.push_back(Interval(l, r
                , up.size(), i, k));
        }
        else {
            ans[i] = -2;
            int p, x;
            read(p); read(x);
            up.push_back(UpdateOp(p, x));
        }
    }
    S = (int)pow(2.0*n*n, 1.0/3); // without
        update operation, S=(int)(n/sqrt(q))
    sort(intervals.begin(), intervals.end(),
        cmp);
    tot = 0;
    qt = 0;
    st = 1; ed = 1;
    add(1);
    for (auto qj: intervals) {
        if (ed <= qj.r) {
            while (ed < qj.r) add(++ed);
            while (ed > qj.r) del(ed--);
            while (st < qj.l) del(st++);
            while (st > qj.l) add(--st);
        }
    }
}

```

```

else {
    while (st < qj.l) del(st++);
    while (st > qj.l) add(--st);
    while (ed < qj.r) add(++ed);
    while (ed > qj.r) del(ed--);
}
if (qt > qj.t) { // recover the update
    operation
}
for (; qt < qj.t; ++qt) { // update
    operation
    if (st <= up[qt].p && up[qt].p <=
        ed)
        del(up[qt].p);
    a[up[qt].p] = up[qt].x;
    if (st <= up[qt].p && up[qt].p <=
        ed)
        add(up[qt].p);
}
ans[qj.id] = solve();
}
for (int i = 0; i < q; ++i)
    if (ans[i] > -2) cout << ans[i] << "\n";
return 0;
}

```

5 String

5.1 KMP

```
int f[maxn];

void getfail(char *P, int *f, int m) {
    f[0] = 0;
    f[1] = 0;
    for (int i = 1; i < m; ++i) {
        int j = f[i];
        while (j && P[i] != P[j]) j = f[j];
        if (P[i] == P[j]) f[i+1] = j+1;
        else f[i+1] = 0;
    }
}

int find(char *T, char *P, int *f, int n, int
m) {
    int res = 0;
    getfail(P, f);
    for (int i = 0, j = 0; i < n; ++i) {
        while (j && P[j] != T[i]) j = f[j];
        if (P[j] == T[i]) j++;
        if (j == m) { // 出现一次
            res++;
            j = f[m];
        }
    }
    return res;
}
```

5.2 Trie

```
struct Trie {
    vector<int> ch[26], val; // val可以存储节
                           // 点的信息
    Trie() { // 这里表示字符串
              // 的编号
        newNode();
    }

    void newNode() {
        for (int i = 0; i < 26; ++i)
            ch[i].push_back(-1);
        val.push_back(-1);
    }
}
```

```
void insert(const string &str, int v) {
    int i, u;
    for (i = 0, u = 0; i < str.length();
        ++i) {
        int c = str[i] - 'a';
        if (ch[c][u] == -1) {
            newNode();
            ch[c][u] = ch[c].size() - 1;
        }
        u = ch[c][u];
    }
    val[u] = v;
}

int find(const string &str) {
    int i, u;
    for (i = 0, u = 0; i < str.length();
        ++i) {
        int c = str[i] - 'a';
        if (ch[c][u] == -1) return -1;
        u = ch[c][u];
    }
    return val[u];
}

};
```


6 Computational Geometry

6.1 2D Template

```

using Point = pair<double, double>;
Point operator+(Point &A, Point &B) { return
    Point(A.X+B.X, A.Y+B.Y); }
Point operator-(Point &A, Point &B) { return
    Point(A.X-B.X, A.Y-B.Y); }
Point operator*(double a, Point A) { return
    Point(a*A.X, a*A.Y); }
Point operator*(Point A, double a) { return a
    *A; }
double Dot(Point A, Point B) { return A.X*B.X
    + A.Y*B.Y; }
double Cross(Point A, Point B) { return A.X*B
    .Y - A.Y * B.X; }
double Distance(Point A, Point B) { return
    sqrt(Dot(A-B, A-B)); }

bool Circle(Point A, Point B, Point C, Point
    &cir) {
    double a1 = 2*(B.X - A.X), b1 = 2*(B.Y -
        A.Y), c1 = Dot(B, B) - Dot(A, A);
    double a2 = 2*(C.X - A.X), b2 = 2*(C.Y -
        A.Y), c2 = Dot(C, C) - Dot(A, A);
    // a1*x + b1*y = c1
    // a2*x + b2*y = c2
    double D = a1 * b2 - a2 * b1;
    if (dcmp(D) == 0) return false;
    cir = Point(c1 * b2 - c2 * b1, a1 * c2 -
        a2 * c1) * (1 / D);
    return true;
}

struct Line {
    Point A, B;
    double rad;
    Line(Point A=MK(0, 0), Point B=MK(0, 0)):
        A(A), B(B) { rad = getAngle(); }
    double getAngle() { return atan2((B-A).Y,
        (B-A).X); }
    Point getdir() { return B-A; }
    bool operator<(const Line &op) const {
        return dcmp(rad-op.rad) < 0;
    }
};

```

```

bool onLeft(Line l, Point p) { return dcmp(
    Cross(l.getdir(), p-l.A)) >= 0; }

bool intersect(Line l1, Line l2, Point &res)
{
    double a1 = l1.getdir().X, b1 = -l2.
        getdir().X, c1 = l2.A.X - l1.A.X;
    double a2 = l1.getdir().Y, b2 = -l2.
        getdir().Y, c2 = l2.A.Y - l1.A.Y;
    double D = a1 * b2 - a2 * b1;
    if (dcmp(D) == 0) return false;
    double t1 = (c1 * b2 - c2 * b1) / D;
    res = l1.A + l1.getdir()*t1;
    return true;
}

```

6.2 Smallest Covering Circle

```

for (int i = 0; i < n; ++i) swap(p[rand()%n],
    p[rand()%n]);
r = 0;
cir = p[0];
for (int i = 0; i < n; ++i) {
    if (dcmp(Distance(cir, p[i]) - r) <= 0)
        continue;
    cir = (p[0] + p[i]) * 0.5;
    r = Distance(p[0], p[i]) * 0.5;
    for (int j = 1; j < i; ++j) {
        if (dcmp(Distance(cir, p[j]) - r) <=
            0) continue;
        cir = (p[i] + p[j]) * 0.5;
        r = Distance(p[i], p[j]) * 0.5;
        for (int k = 0; k < j; ++k) {
            if (dcmp(Distance(cir, p[k]) - r)
                <= 0) continue;
            Circle(p[i], p[j], p[k], cir);
            r = Distance(cir, p[i]);
        }
    }
}

```

6.3 Half Plane Intersection

```

/*
 * input: Array of lines

```

```

* output: the points of intersection
* Time: O(nlogn)
*/
deque<Line> Q;
deque<Point> Qp;
vector<Point> halfPlaneCross(vector<Line> &ls
) {
    while (!Q.empty()) Q.pop_front();
    while (!Qp.empty()) Qp.pop_front();
    sort(ALL(ls), cmp);
    Q.push_back(ls[0]);
    for (int i = 1; i < SZ(ls); ++i) {
        if (dcmp(Cross(ls[i].getdir(), ls[i-1].getdir())) != 0) {
            while (!Qp.empty() && !onLeft(ls[i], Qp.back())) Qp.pop_back(),
                Qp.pop_back();
            while (!Qp.empty() && !onLeft(ls[i], Qp.front())) Qp.pop_front(),
                Qp.pop_front();
            Point tmp;
            intersect(Q.back(), ls[i], tmp);
            Q.push_back(ls[i]);
            Qp.push_back(tmp);
        }
    }
    while (!Qp.empty() && !onLeft(Q.front(), Qp.back())) Qp.pop_back(), Qp.pop_back();
    vector<Point> ansp;
    if (Qp.size() <= 1) return ansp;
    ansp.PB(Point());
    intersect(Q.front(), Q.back(), ansp[0]);
    while (!Qp.empty()) ansp.PB(Qp.front()),
        Qp.pop_front();
    return ansp;
}

```

A Env Conf

```
" user configuration
"set cindent
"set autoindent
"set nu rnu
"set tabstop=4
"set backspace=2
"set vb t_vb=
"se shiftwidth=4
" set background=dark
" inoremap ( (<LEFT>
" inoremap [ [<LEFT>
" colorscheme evening
syntax on
color molokai
se ai nu rnu bs=2 ts=4 sw=4
se guifont=Consolas:b:h12
imap {<CR> {<CR>}<ESC>O
imap <F5> <ESC>:call Run()<CR>
imap <C-S> <ESC>:w<CR>

map <C-A>c ggvG$"+y
map <C-A>v ggvG$"+p
map <C-S> :w<CR>
map <F5> :call Run()<CR>
" if os is Linux, replace %<.exe with ./%<
func! Run()
    exec "w"
    exec "!g++ -Wall -g % -o %<.exe"
    exec "silent !%<.exe < my.in > my.out"
endfunc

map <F10> :call CaR()<CR>
func! CaR()
    exec "w"
    exec "!g++ -Wall -g % -o %<.exe"
    exec "!%<.exe"
endfunc
" Non
map <F4> :call PY()<CR>
func! PY()
    exec "w"
    exec "!python %"
endfunc

map <F6> :call Debug()<CR>
```

```
func! Debug()
    exec "w"
    exec "silent !g++ -Wall -g % -o %<.exe"
    exec "!gdb %<.exe"
endfunc
map <F7> :call Finderror()<CR>
func! Finderror()
    exec "w"
    exec "silent !g++ -Wall -g test.cpp -o
        test.exe"
    exec "silent !g++ -Wall -g % -o %<.exe"
    exec "silent !%<.exe < my.in > my.out"
    exec "silent !test.exe < my.in > ans.txt"
    exec "!fc ans.txt my.out"
endfunc
```

```
// launch
{
    // 使用 IntelliSense 了解相关属性。
    // 悬停以查看现有属性的描述。
    // 欲了解更多信息，请访问: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "g++.exe build and debug
                active file",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}\\${
                fileBasenameNoExtension}.exe"
            ,
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "miDebuggerPath": "C:\\mingw64\\
                bin\\gdb.exe",
            "setupCommands": [
                {
                    "description": "Enable
                        pretty-printing for
                        gdb",
                    "text": "-enable-pretty-
                        printing",
```

```

        "ignoreFailures": true
    }
},
"preLaunchTask": "g++.exe build
    active file"
}
]
}

```

```

// tasks
{
// 有关 tasks.json 格式的文档, 请参见
// https://go.microsoft.com/fwlink/?LinkId
// =733558
"version": "2.0.0",
"tasks": [
    {
        "type": "shell",
        "label": "g++.exe build active
            file",
        "command": "C:\\mingw64\\bin\\g
            ++.exe",
        "args": [
            "-Wall",
            "-g",
            "${file}",
            "-o",
            "${fileDirname}\\${
                fileBasenameNoExtension}.
                exe"
        ],
        "options": {
            "cwd": "C:\\mingw64\\bin"
        },
        "problemMatcher": [
            "$gcc"
        ],
        "group": {
            "kind": "build",
            "isDefault": true
        }
    }
]
}

```

```

// properties
{

```

```

"configurations": [
    {
        "name": "Win32",
        "includePath": [
            "${workspaceFolder}/**"
        ],
        "defines": [
            "_DEBUG",
            "UNICODE",
            "_UNICODE"
        ],
        "compilerPath": "C:\\mingw64\\bin
            \\g++.exe",
        "cStandard": "gnu17",
        "cppStandard": "gnu++14",
        "intelliSenseMode": "gcc-x64"
    }
],
"version": 4
}

```

```

cmake_minimum_required(VERSION 3.16)
project(code)
set(CMAKE_CXX_STANDARD 14)

file (GLOB files *.cpp */*.cpp)
foreach (file ${files})
    string (REGEX REPLACE ".+/(.+)/(.+)|\\..*"
        "|1-|2" exe ${file})
    add_executable (${exe} ${file})
endforeach ()

```

B Theorem

B.1 Lucas' Theorem

对于质数 p , 有

$$\binom{n}{m} \bmod p = \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \cdot \binom{n \bmod p}{m \bmod p} \bmod p$$

B.2 Betty's Theorem

如果两个无理数 a, b 满足:

$$\frac{1}{a} + \frac{1}{b} = 1$$

那么对于两个集合 A, B :

$$A = \{[na]\}, B = \{[nb]\}$$

有下面两个结论:

$$A \cap B = \emptyset, A \cup B = \mathbb{N}^+$$

B.3 Bernoulli Number

定义: $S(n, m) = \sum_{i=1}^n i^m$ ($m > 0, n > 0$)
 则必有: $S(n, m) = \frac{1}{m+1} \sum_{i=0}^m \binom{m+1}{i} B_i n^{m+1-i}$
 其中: $B_n = -\frac{1}{n+1} \sum_{i=0}^{n-1} \binom{n+1}{i} B_i$, $B_0 = 1$, $B_1 = \pm \frac{1}{2}$, $B_2 = \frac{1}{6}$, $B_3 = 0$, $B_4 = -\frac{1}{30}$
 一般来说, 自然数幂求和要 B_1 取正数。另外, 伯努利数和 $\zeta(s)$ 有关,

$$B_{2m} = \frac{2(-1)^{m-1}(2m)!}{(2\pi)^{2m}} \zeta(2m)$$

使用多项式求逆或者分治 fft 可以快速计算伯努利数, 因为伯努利数满足

$$\frac{x}{e^x - 1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = \frac{1}{\frac{e^x - 1}{x}}$$

C C++ STL: set

set 与 unordered_set 的区别在于有无在内部存储时有无顺序。

C.1 Basic Method

begin() 返回 set 容器的第一个元素
 end() 返回 set 容器的最后一个元素
 clear() 删除 set 容器中的所有元素
 empty() 判断 set 容器是否为空
 max_size() 返回 set 容器可能包含的元素最大个数
 size() 返回当前 set 容器中的元素个数
 rbegin() 返回的值和 end() 相同
 rend() 返回的值和 rbegin() 相同
 count() 用来查找 set 中某个键值出现的次数。(在 set 中只有 0 或 1 次)
 equal_range() 返回一对定位器, 分别表示第一个大于或等于给定键值的元素和第

一个大于给定键值的元素, 这个返回值是一个 pair 类型, 如果这一对定位器中哪个返回失败, 就会等于 end() 的值

erase(iterator) 删除定位器 iterator 指向的值

erase(first, second) 删除定位器 first 和 second 之间的值

erase(key_value) 删除键值 key_value 的值

insert(key_value) 将 key_value 插入到 set 中, 返回值是 pair<set<int>::iterator, bool>, bool 标志着插入是否成功, 而 iterator 代表插入的位置, 若 key_value 已经在 set 中, 则 iterator 表示的 key_value 在 set 中的位置

lower_bound(key_value) 返回第一个大于或等于 key_value 的定位器

upper_bound(key_value) 返回最后一个大于或等于 key_value 的定位器

C.2 Advanced Method

注: 必须导入 algorithm 头文件
 set_intersection(first1, last1, first2, last2, d_first, comp)
 first1, last1 - 要检验的第一元素范围
 first2, last2 - 要检验的第二元素范围
 d_first - 输出范围的起始
 comp - 比较函数对象 (即满足比较 (Compare) 概念的对象), 若第一参数小于 (即先序于) 第二参数则返回 true

Example:

```
std::set_intersection(v1.begin(), v1.
    end(), v2.begin(), v2.end(), std::
    back_inserter(v_intersection));
// v_intersection 就是交集,
    back_inserter () 用于 vector
```

set_union(first1, last1, first2, last2, d_first, comp)
 同 intersection