ACM/XCPC Template Sweat Boys

浙江工商大学

糖果超甜

October 27, 2021

Contents

0 text

0.1 o2,o3

```
#pragma GCC optimize(3,"Ofast","inline")

#pragma GCC optimize(2)
```

0.2 测试模板

```
signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    signed test_index_for_debug = 1;
    char acm_local_for_debug = 0;
    do {
        if (acm_local_for_debug == '$') exit(0);
        if (test_index_for_debug > 20)
            throw runtime_error("Check the stdin!!!");
        auto start_clock_for_debug = clock();
        solve();
        auto end_clock_for_debug = clock();
        cout << "Test " << test_index_for_debug << " successful" << endl;
        cerr << "Test " << test_index_for_debug++ << " Run Time: "
            << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC << "s" << endl;
        cout << "--------------------------------------------------" << endl;
    } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
#else
    solve();
#endif
    return 0;
}
```

0.3 读入输出模板

```
inline int read(){
   int s=0,w=1;
   char ch=getchar();
   while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
   while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
   return s*w;
}

----------------------------------------------------------------------------
namespace StandardIO {
    inline char nc() {
        static char buf[1000000],*p1=buf,*p2=buf;
        return p1==p2&&(p2=(p1=buf)+fread(buf,1,1000000,stdin),p1==p2)?EOF:*p1++;
    }
    template <typename _Tp> inline void read(_Tp&sum) {
        char ch=nc();sum=0;
        while(!(ch>='0'&&ch<='9')) ch=nc();
        while(ch>='0'&&ch<='9') sum=(sum<<3)+(sum<<1)+(ch-48),ch=nc();
    }

    template<typename T>
    inline void write(T x) {
        if (x < 0) putchar('-'), x *= -1;
        if (x >= 10) write(x / 10);
        putchar(x % 10 + '0');
    }
}

-------------------------------------------------------------------------------
inline __int128 read()
{
   int X=0,w=0; char ch=0;
   while(!isdigit(ch)) {w|=ch=='-';ch=getchar();}
   while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48),ch=getchar();
   return w?-X:X;
}
inline void print(__int128 x)
{
   if(x<0){putchar('-');x=-x;}
   if(x>9) print(x/10);
   putchar(x%10+'0');
}
```

0.4

```
/***□ □ ļ□ ***/
/*SiberianSquirrel*//*CuteKiloFish*/
#include <bits/stdc++.h>
//#include<bits/extc++.h>
#include<ext/rope>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_cxx;
using namespace __gnu_pbds;
#define Inv(x) quick_pow(x, mod - 2)
#define Polynomial vector<int>
#define DEBUG(x, y) cout << x << ": " << y << '\n';
#define mem(a, x) memset(a, x, sizeof a)
#define right_1_pos(x) __builtin_ffs(x)
#define left_0_num(x) __builtin_clz(x)
#define right_0_num(x) __builtin_ctz(x)
#define num_of_1(x) __builtin_popcount(x)
#define Pii pair<int, int>
#define mp_(x, y) make_pair(x, y)
#define all(v) (v).begin(), (v).end()
using ld = long double;
using ll = long long;
//using ill = __int128;
using ull = unsigned long long;
using i16 = short;
const ld pi = acos(-1.0);
const ld eps = 1e-8;
const ll mod = 998244353, mod_g = 3, img = 86583718;
int inv2, inv3;
```

```
tree<ll, null_type, less<ll>, rb_tree_tag, tree_order_statistics_node_update> tr;
```

0.5 main

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<ll, ll> PII;
typedef unsigned long long ull;
const ll inf = 1e18;
const int N = 2e6 + 10;
const int M = 1e6 + 10;
const double eps = 1e-8;
const int mod = 1e9 + 7;

#define fi first
#define se second
#define re register
#define lowbit (-x&x)

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("input", "r", stdin);
    freopen("output", "w", stdout);
#endif

#ifdef ACM_LOCAL
    auto start = clock();
#endif
    int t = 1;
//    cin >> t;
    while (t--)
        solve();
#ifdef ACM_LOCAL
    auto end = clock();
    cerr << "Run Time: " << double(end - start) / CLOCKS_PER_SEC << "s" << endl;
#endif
    return 0;
}
```

1 dp（动态规划）

1.1 二维费用

```
时间复杂度 O(n^3)

f[i][j] = max(f[i][j], f[i-a]+f[j-b] + w);
```

1.2 LIS 问题 nlogn

```
/*

void solve() {
    while (cin >> a[++n]);
    n--;
    for (int i = 1; i <= n; i++) {
        if (a[i] > b[len1]) b[++len1] = a[i];
        else {
            int pos = lower_bound(b + 1, b + len1 + 1, a[i]) - b;
            b[pos] = a[i];
        }
    }

    reverse(a+1, a+n+1);
    for (int i = 1; i <= n; i++) {
        if (a[i] >= b[len2]) b[++len2] = a[i];
        else {
            int pos = upper_bound(b + 1, b + len2 + 1, a[i]) - b;
            b[pos] = a[i];
        }
    }
    cout << len2 << endl;
    cout << len1 << endl;
}


*/
```

1.3 SOS dp

```

//
// Created by SANZONG on 2021/7/21.
//
#include "bits/stdc++.h"

#define int long long
using namespace std;
const int N = 1 << 20;
const int mod = 998244353;
int a[N], b[N];
int c[N];
int d1[N], d2[N];
int d3[N], d4[N];

signed main() {
    ios::sync_with_stdio(false);
    int T;
    cin >> T;
    while (T--) {
        int n;
        cin >> n;
        for (int i = 0; i < n; ++i) {
            cin >> a[i];d1[i] = a[i];d2[i] = a[i];
        }
        for (int i = 0; i < n; ++i) {
            cin >> b[i];d3[i] = b[i];d4[i] = b[i];
        }
        c[n] = -(1ll << 62);
        for (int i = n - 1; i >= 0; i--) {          //枚举每个数
            for (int j = 0; (1 << j) < n; j++) {
                if (i & (1 << j)) {                 //属于子集
                    d1[i ^ (1<<j)] = max(d1[i ^ (1<<j)], d1[i ]);           //求k<=i中k能对应的最大的a[i]，注意转移
                    d2[i ^ (1<<j)] = min(d2[i ^ (1<<j)], d2[i]);
                    d3[i ^ (1<<j)] = max(d3[i ^ (1<<j)], d3[i ]);
                    d4[i ^ (1<<j)] = min(d4[i ^ (1<<j)], d4[i ]);
                }
            }
        }
        int ans = 0;
        for (int i = n - 1; i >= 0; --i) {
            c[i] = max({c[i + 1], d2[i] * d4[i], d1[i] * d3[i], d1[i] * d4[i], d2[i] * d3[i]});
            ans = (ans + c[i]) % mod;
        }
        cout << (ans + mod) % mod << endl;
    }
}
```

1.4 SOSDP

```
const int bit = 20;
const int mx = 1 << bit | 1;
void sosdp(vector<int> &a, vector<int> &dp, int n) {
//    for(int i = 0; i <= n; ++ i) dp[i] = 0;
    for(int i = n; i >= 0; ++ i) {
        for(int j = 0; j < bit; ++ j) {
            if((i | (1 << j)) > n) continue;
            dp[i] = (dp[i] + dp[i | (1 << j)]) % mod;
        }
    }
}
```

1.5 SOSDP2

```
const int bit = 19;
const int mx = 1 << bit | 1;
ll A[mx], F[mx], B[mx];
ll maxx[2][mx], minn[2][mx];

void solve() {
    int o; cin >> o; while(o--) {
        int n; cin >> n;
        for (int i = 0; i < n; ++ i) {
            cin >> A[i];
            maxx[0][i] = minn[0][i] = A[i];
        }
        for (int i = 0; i < n; ++ i) {
            cin >> B[i];
            maxx[1][i] = minn[1][i] = B[i];
        }

        for(int j = mx; j >= 0; -- j) {
            for(int i = 0; i < bit; ++ i) {
                if((j | (1 << i)) >= n) continue;
                maxx[0][j] = max(maxx[0][j], maxx[0][j | (1 << i)]);
                maxx[1][j] = max(maxx[1][j], maxx[1][j | (1 << i)]);
                minn[0][j] = min(minn[0][j], minn[0][j | (1 << i)]);
                minn[1][j] = min(minn[1][j], minn[1][j | (1 << i)]);
            }
        }
        ll maxxx = -2e18, res = 0;
        for(int i = n - 1; i >= 0; -- i) {
            maxxx = max(maxxx, maxx[0][i] * maxx[1][i]);
            maxxx = max(maxxx, maxx[0][i] * minn[1][i]);
            maxxx = max(maxxx, minn[0][i] * maxx[1][i]);
            maxxx = max(maxxx, minn[0][i] * minn[1][i]);
            res = (res + maxxx) % mod;
            while(res < 0) res += mod;
        }
        cout << res << '\n';
    }
}
```

1.6 背包问题方案

```
如果要求字典序最小

n --> 1 做一遍普通的背包

1 --> n 取一遍方案，如果f[i][cur] = f[i+1][cur-v]+w，那么i就是方案

字典序最大，反过来即可

时间复杂度O(nm)
```

1.7 背包最优方案计数问题

```
可以不装满背包，保证总价值最大

f[N]记录价值，cnt[N]记录个数

f[j] = max(f[j], f[j-v]+w);

如果当前f[j-v] + w > f[j]，说明是f[j-v]+w更新的答案，因此cnt[j] = cnt[j-v];

如果当前f[j-v] + w = f[j]，说明两者都可，cnt[j] += cnt[j-v];

别忘了cnt初始化为 1，因为背包可以空着也算一种方案

时间复杂度O(nm)
```

1.8 单调队列 dp

```
//
// Created by SANZONG on 2021/9/17.
//

//
// Created by SANZONG on 2020/8/10.
//
//lca模板

#include "bits/stdc++.h"
#define int long long
const int maxn = 1e6;
using namespace std;
int dp[maxn];
int sum[maxn];
int q[maxn];
int a[maxn];
signed main() {
    int n,k;
    cin >> n >> k;
    for (int i = 1; i <= n; ++i) {
        int s;cin >> s;
        sum[i] = sum[i-1] + s;
    }
    int head = 1;
    int tail = 0;
    q[++tail] = 0;
    for (int i = 1; i <= n; ++i) {
        dp[i] = a[q[head]] + sum[i];
        a[i] = dp[i-1]-sum[i];
        while (head <= tail && i - q[head] + 1 > k)
            head++;
        while (head <= tail && a[q[tail]] < a[i])
            tail--;
        q[++tail] = i;
    }
    cout << dp[n]  << endl;
}
```

1.9 多重背包

```
1.朴素算法

时间复杂度O(n^3)

for (int i = 1; i <= n; i++)
    for (int j = m; j >= v[i]; j--)
        for (int k = 1; k <= s[i]; k++)
            f[j] = max(f[j], f[j - k*v[i]] + k*w[i]);

核心 f[j] = max(f[j], f[j-v] + w, f[j-v*2] + 2*w, f[j-3*v] + 3*w ....)

2.二进制优化

时间复杂度O(nm*log(n))

```

```
例如: 10 = 1 + 2 + 4 + 3

将问题转换为01背包问题

3.单调队列优化

时间复杂度O(nm)

f[0] = max(f[0], f[v] + w, f[2*v] + 2*w, f[3*v] + 3*w...)
f[1] = max(f[1], f[1+v] + w, f[1+2*v] + 2*w, f[1+3*v] + 3*w...)
f[2] = max(f[2], f[2+v] + w, f[2+2*v] + 2*w, f[2+3*v] + 3*w...)
...
f[v-1] = max(f[v-1], f[v-1+v] + w, f[v-1+2*v] + 2*w, f[v-1+3*v] + 3*w...)

所有的 m 都可以转换为 c + a * x, 而且是互不影响的

f[j] = f[j]
f[j+v] = max f[j+v], f[j] + w
f[j+2*v] = max f[j+2*v], f[j+v] + w, f[j] + 2*w

修改一下变成
f[j] = f[j]
f[j+v] = max(f[j+v] - w, f[j]) + w
f[j+2*v] = max(f[j+2*v] - 2*w, f[j+v] - w, f[j]) + 2*w

将 f[j + k*w] - k*w 存入单调队列中, 取值时用加上 k*w

#include <bits/stdc++.h>
using namespace std;

const int N = 20010;

int f[N], g[N], n, m;
int q[N];

int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
        memcpy(g, f, sizeof f);
        for (int j = 0; j < v; j++) {
            int hh = 1, tt = 0;
            for (int k = j; k <= m; k += v) {
                while (hh <= tt && k - q[hh] > s*v) hh++; //队列中最多只能有s+1个值
                if (hh <= tt) f[k] = max(f[k], g[q[hh]] + ( k - q[hh]) / v * w);
                while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v * w) tt--;
                q[++tt] = k;
            }
        }
    }
    cout << f[m] << endl;
}
```

1.10 分组背包

```
v[0....n]代表组
```

```

f[j] = max(f[j], f[j-v[0]]+w[0], f[j-v[1]]+w[1], f[j-v[2]]+w[2].....)

枚举三层循环即可
```

1.11 混合背包

```
假设存在无限个，一个，有限个的物品

将多重背包用二进制优化转换为01背包

接着按照完全背包和01背包做就行
```

1.12 区间 dp

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define ACM_LOCAL

const int N = 200 + 5;
const int Mod = 1e9 + 7;
int n, s[N], f1[N][N], f2[N][N];

void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> s[i], s[i+n] = s[i];
    for (int i = 1; i <= 2*n; i++) s[i] += s[i-1];

    for (int len = 2; len <= n; len++) {
        for (int i = 1; i + len - 1 <= 2*n; i++) {
            int j = i + len - 1;
            f1[i][j] = 1e8;
            for (int k = i; k < j; k++) {
                f1[i][j] = min(f1[i][j], f1[i][k] + f1[k+1][j] + s[j] - s[i-1]);
                f2[i][j] = max(f2[i][j], f2[i][k] + f2[k+1][j] + s[j] - s[i-1]);
            }
        }
    }
    int Max = 0, Min = 1e8;
    for (int i = 1; i + n - 1 <= 2*n; i++) {
        Max = max(Max, f2[i][i+n-1]);
        Min = min(Min, f1[i][i+n-1]);
    }
    cout << Min << endl;
    cout << Max << endl;
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
```

```
    solve();
    return 0;
}
```

1.13 数位 dp

```
ll dfs(int pos, int pre, bool limit, bool lead) {
    if (pos == -1) return 1;
    if (!limit && !lead && dp[pos][...][...] != -1) return dp[pos][...][...];
    int End = limit ? a[pos] : 9;
    ll ans = 0;
    for (int i = 0; i <= End; i++) {
        ******//操作1
    }
    if (!limit && !lead) dp[pos][...][...] = ans;
    return ans;
}
ll calc(ll x) {
    int pos = 0;
    while (x) {
        a[pos++] = x % 10;
        x /= 10;
    }
    return dfs(pos-1, 0, true, true);
}
```

1.14 四边形优化

```
//
// Created by acer on 2020/11/23.
//

//m(i,j)=min{m(i,k-1),m(k,j)}+w(i,j)(i≤k≤j)
//满足对于 i≤i'<j≤j'，有 w(i',j)≤w(i,j')，且对于 i≤i'<j≤j'，有 w(i,j)+w(i',j')≤w(i',j)+w(i,j')
//则m函数具有相同性质，故可以简单的推出m的最优决策点有s(i,j)≤s(i,j+1)≤s(i+1,j+1)
int main() {
    int f[100][100];
    int m[100][100];
    int n, INF = 100;
    for (int i = 1; i <= n; ++i) {
        m[i][i] = i;
    }
    for (int len = 2; len <= n; ++len)  // 枚举区间长度
        for (int l = 1, r = len; r <= n; ++l, ++r) {  // 枚举长度为len的所有区间
            f[l][r] = INF;
            for (int k = m[l][r - 1]; k <= m[l + 1][r]; ++k)
                if (f[l][r] > f[l][k] + f[k + 1][r] + w(l, r)) {
                    f[l][r] = f[l][k] + f[k + 1][r] + w(l, r);  // 更新状态值
                    m[l][r] = k;  // 更新（最小）最优决策点
                }
        }
}
```

1.15 完全背包

```
f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], .....)

f[i][j - v[i]] + w[i] = max(f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], ......)

f[i][j] = max(f[i-1][j], f[i-1][j-v[i]]+w[i])

==> for (int i = v[i]; i <= m; i++)

因此 j 使用当前的状态更新自己， 所以正序
```

1.16 悬线法

```
#include<cstdio>

#define N 2005
#define max(a, b) a>b?a:b
#define min(a, b) a<b?a:b
using namespace std;
int up[N][N], left[N][N], right[N][N], ansa, ansb, a[N][N], m, n;

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            up[i][j] = 1, left[i][j] = j, right[i][j] = j, scanf("%d", &a[i][j]);// up初值，读入，left/right 最初值

    for (int i = 1; i <= n; i++)
        for (int j = 2; j <= m; j++)
            if (a[i][j] ^ a[i][j - 1])
                left[i][j] = left[i][j - 1];
    for (int i = 1; i <= n; i++)
        for (int j = m; j > 1; j--)
            if (a[i][j] ^ a[i][j - 1])
                right[i][j - 1] = right[i][j];//left/right初值，即（i，j）点向左/右的最大宽度

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++) {
            if (i > 1 && a[i][j] ^ a[i - 1][j])
                up[i][j] = up[i - 1][j] + 1, left[i][j] = max(left[i][j],left[i - 1][j]), right[i][j] = min(right[i][j],right[i - 1][j]);
            int a = right[i][j] - left[i][j] + 1;
            int b = min(a, up[i][j]);
            ansa = max(ansa, b * b);
            ansb = max(ansb, a * up[i][j]);
        }
    printf("%d\n%d", ansa, ansb);
}
```

1.17 有依赖的背包 (树形 dp)

```
类似分组背包的思想

for (int j = m; j >= v[i])
    for (int k = 0; k <= j; k++)
        f[x][j] = max(f[x][j], f[x][j-k] + f[y][k]);
```

```

for (int i = m; i >= v[x]; i--) f[x][i-v[x]] + w[x];
for (int i = 0; i < v[x]; i++) f[x][i] = 0;
每个子节点为一个组，组内不同的背包容量转移为组内的不同元素
```

1.18 01 背包

```
f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i])

因为 i 只用到前后的两次状态，因此可以压缩成一维的空间

f[j] = max(f[j], f[j-v[i]] + w[i])

并且 for (int j = m; j >= v[i]; j--)

因为f[j[v[i]]] + w[i]要用到i-1维的数据更新当前背包空间，所以j倒序
```

2 树形结构

2.1 lca(倍增)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
vector<int> g[N];

int f[N][30], dep[N];

void dfs(int u, int fa) {
    dep[u] = dep[fa] + 1; f[u][0] = fa;
    for (int i = 1; i <= 19; i++) f[u][i] = f[f[u][i-1]][i-1];
    for (auto &v : g[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int x, int y) {
    if (dep[x] > dep[y]) swap(x, y);
    for (int i = 19; i >= 0; i--) if (dep[y] - dep[x] >= (1 << i)) y = f[y][i];

    if (x == y) return x;
    for (int i = 19; i >= 0; i--) {
        if (f[y][i] != f[x][i])
            y = f[y][i], x = f[x][i];
    }
    return f[x][0];
}
```

2.2 lca（树链剖分）

```
int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (d[top[u]] > d[top[v]])
            u = fa[top[u]];
        else
            v = fa[top[v]];
    }
    return d[u] > d[v] ? v : u;
}
```

2.3 次小生成树

```
#include <bits/stdc++.h>
using namespace std;
#define INF 1e16
const int N = 1e5 + 200;
const int M = 6 * 1e5 + 300;
int head[M], edge[M], Next[M], ver[M], tot, fa[M], n, m, father[N][32], deep[N];
long long dp[2][N][32], val1, val2, ans_max, ans;

struct node
{
    int x, y, z, vis;
```

```
} s[M];

int cmp(node a, node b)
{
    return a.z < b.z;
}

struct Edge {
    void init2() {
        memset(head, 0, sizeof(head));
        tot = 0;
    }
    void add_edge(int a, int b, int c) {
        edge[++tot] = b;
        ver[tot] = c;
        Next[tot] = head[a];
        head[a] = tot;
    }
    int find(int x) {
        return x == fa[x] ? x : fa[x] = find(fa[x]);
    }
    void Kruskal() {
        sort(s + 1, s + 1 + m, cmp);
        for (int i = 1; i <= m; i++)
        {
            int a = find(s[i].x), b = find(s[i].y);
            if (a == b)
                continue;
            s[i].vis = 1;
            fa[a] = b;
            ans += s[i].z;
            add_edge(s[i].x, s[i].y, s[i].z);
            add_edge(s[i].y, s[i].x, s[i].z);
        }
    }
    void bfs(int root) {
        deep[root] = 0;
        queue<int> q;
        q.push(root);
        while (q.size()) {
            int x = q.front(), len = (int)log2(deep[x] + 1);
            q.pop();
            for (int i = head[x]; i; i = Next[i]) {
                int y = edge[i];
                if (y == father[x][0])
                    continue;
                deep[y] = deep[x] + 1;
                father[y][0] = x, dp[0][y][0] = ver[i], dp[1][y][0] = -INF;
                q.push(y);
                for (int t = 1; t <= len; t++) {
                    father[y][t] = father[father[y][t - 1]][t - 1];
                    if (dp[0][y][t - 1] != dp[0][father[y][t - 1]][t - 1]) {
                        dp[0][y][t] = max(dp[0][y][t - 1], dp[0][father[y][t - 1]][t - 1]);
                        dp[1][y][t] = min(dp[0][y][t - 1], dp[0][father[y][t - 1]][t - 1]);
                    }
                    else {
                        dp[0][y][t] = dp[0][y][t - 1];
```

```cpp
                        dp[1][y][t] = max(dp[1][y][t - 1], dp[1][father[y][t - 1]][t - 1]);
                    }
                }

            }
        }
    }
    inline void update2(int x) {
        if (x > val1)
            val2 = val1, val1 = x;
        else if (x > val2 && x != val1)
            val2 = x;
    }
    inline void update(int x, int t) {
        update2(dp[0][x][t]);
        update2(dp[1][x][t]);
    }
    inline void Lca(int x, int y) {
        val1 = val2 = -INF;
        if (deep[x] < deep[y])
            swap(x, y);
        while (deep[x] > deep[y]) {
            int t = (int)log2(deep[x] - deep[y]);
            update(x, t), x = father[x][t];
        }
        if (x == y)
            return;
        for (int t = (int)log2(deep[x]); t >= 0; t--) {
            if (father[x][t] != father[y][t]) {
                update(x, t), update(y, t);
                x = father[x][t];
                y = father[y][t];
            }
        }
        update(x, 0), update(y, 0);
    }
} g1;

void solve() {
    scanf("%d%d", &n, &m);
    g1.init2();
    for (int i = 1; i <= m; i++) {
        int a, b, c;
        scanf("%d%d%d", &a, &b, &c);
        s[i].x = a, s[i].y = b, s[i].z = c;
        fa[i] = i;
    }
    g1.Kruskal();
    g1.bfs(1);
    ans_max = INF;
    for (int i = 1; i <= m; i++) {
        if (!s[i].vis) {
            g1.Lca(s[i].x, s[i].y);
            if (val1 != s[i].z)
                ans_max = min(ans_max, ans - val1 + s[i].z);
            else
                ans_max = min(ans_max, ans - val2 + s[i].z);
        }
    }
    printf("%lld\n", ans_max);
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    signed test_index_for_debug = 1;
    char acm_local_for_debug = 0;
    do {
        if (acm_local_for_debug == '$') exit(0);
        if (test_index_for_debug > 20)
            throw runtime_error("Check the stdin!!!");
        auto start_clock_for_debug = clock();
        solve();
        auto end_clock_for_debug = clock();
        cout << "Test " << test_index_for_debug << " successful" << endl;
        cerr << "Test " << test_index_for_debug++ << " Run Time: "
            << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC << "s" << endl;
        cout << "--------------------------------------------------" << endl;
    } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
#else
    solve();
#endif
    return 0;
}
```

2.4 点分治

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>
#include <stack>
#include <cmath>
#include <bitset>
#include <map>

using namespace std;
typedef long long ll;
const int N = 1e4 + 5, M = 2e5 + 5, INF = 0x3f3f3f3f;

struct Edge {
    int to, next, vi;
} e[N << 1];

int h[N], cnt;

void add(int u, int v, int w) {
    e[cnt].to = v;
    e[cnt].vi = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

ll ans = 0;
int root, sum, n;
int sz[N], mx[N], vis[N];

void getroot(int x, int fa) {
    sz[x] = 1, mx[x] = 0;
    for (int i = h[x]; ~i; i = e[i].next) {
        int y = e[i].to;
        if (vis[y] || y == fa) continue;
        getroot(y, x);
        sz[x] += sz[y];
        mx[x] = max(mx[x], sz[y]);
    }
    mx[x] = max(mx[x], sum - sz[x]);
    if (mx[x] < mx[root]) root = x;
}

int d[N], dep[N];

void getd(int x, int fa) {
    d[++d[0]] = dep[x];
    for (int i = h[x]; ~i; i = e[i].next) {
        int y = e[i].to;
        if (vis[y] || y == fa) continue;
        dep[y] = dep[x] + e[i].vi;
        getd(y, x);
    }
}

int K;

int cal(int x, int now) {
    dep[x] = now, d[0] = 0;
    getd(x, -1);
    sort(d + 1, d + d[0] + 1);
    int res = 0, l = 1, r = d[0];
    while (l < r) {
        if (d[l] + d[r] > K) r--;
        else res += r - l, l++;
    }
    return res;
}

void work(int x) {
    ans += cal(x, 0);
    vis[x] = 1;
    for (int i = h[x]; ~i; i = e[i].next) {
        int y = e[i].to;
        if (vis[y]) continue;
        ans -= cal(y, e[i].vi);
        sum = sz[y], root = 0;
        getroot(y, -1);
        work(root);
    }
}

```

```
int main() {
    while (scanf("%d %d", &n, &K), n) {
        cnt = 0;
        memset(h, -1, sizeof h);
        memset(vis, 0, sizeof vis);
        for (int i = 1; i <= n - 1; i++) {
            int u, v, w;
            scanf("%d %d %d", &u, &v, &w);
            add(u, v, w), add(v, u, w);
        }
        root = 0, ans = 0, sum = n, mx[0] = INF;
        getroot(1, -1);
        work(root);
        printf("%lld\n", ans);
    }
}
```

2.5 树的半径

```
#include <bits/stdc++.h>
using namespace std;
const int N = 5000 + 5;
int dis[N], n, m, h[N], cnt, maxd1[N], maxd2[N], mv[N], st, r;

struct edge {
    int u, to, next, vi;
}e[N<<1];

void add(int u, int v, int w) {
    e[cnt].u = u;
    e[cnt].to = v;
    e[cnt].vi = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

void dp(int u, int fa) {
    maxd1[u] = 0, maxd2[u] = 0;
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa) continue;
        dp(v, u);
        int tmp = maxd1[v] + e[i].vi;
        if (tmp > maxd1[u]) maxd2[u] = maxd1[u], maxd1[u] = tmp, mv[u] = v;
        else if (tmp > maxd2[u]) maxd2[u] = tmp;
    }
    st = max(st, maxd1[u] + maxd2[u]);
}

void dfs(int u, int fa, int fr) {
    r = min(r, max(fr, maxd1[u]));
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa) continue;
        if (mv[u] == v) dfs(v, u, max(fr + e[i].vi, maxd2[u] + e[i].vi));
        else dfs(v, u, max(maxd1[u] + e[i].vi, fr + e[i].vi));
    }
}
```

```

int main() {
    cin >> n;
    memset(h, -1, sizeof h);
    for (int i = 1; i <= n - 1; i++) {
        int x, y, z;
        cin >> x >> y >> z;
        add(x, y, z), add(y, x, z);
    }
    dp(1, 0);
    r = 0x3f3f3f3f;
    dfs(1, 0, 0);
    cout << r << endl;
}
```

2.6 树的直径

```
//--------------------------树形dp
int f[N];
int ans;
vector<int> g[N];
void dp(int u, int fa) {
    for (auto v : g[u]) {
        if (v == fa) continue;
        dp(v, u);
        ans = max(ans, f[u] + f[v] + 1);
        f[u] = max(f[u], f[v] + 1);
    }
}


//--------------------------两次dfs

int dfs(int u, int fa) {
    int maxpos = u;
    for (auto v : g[u]) {
        if (vis[v] || v == fa) continue;
        dis[v] = dis[u] + 1;
        pre[v] = u;
        int z = dfs(v, u);
        if (dis[z] > dis[maxpos]) maxpos = z;
    }
    return maxpos;
}
```

2.7 树哈希

```

vector<int> g[N];
int prime[N], siz[N], mx[N], cnt, rt, n;
bool is_prime[N];
ull Hash[N];
void get_prime(int n) {
    memset(is_prime, true, sizeof is_prime);
    for (int i = 2; i <= n; i++) {
        if (is_prime[i]) prime[++cnt] = i;
        for (int j = 1; j <= cnt && prime[j] * i <= n; j++) {
```

```cpp
            is_prime[prime[j] * i] = false;
            if (i % prime[j] == 0) break;
        }
    }
}
void getRt(int x, int fa) {
    siz[x] = 1, mx[x] = 0;
    for (auto v : g[x]) {
        if (v == fa) continue;
        getRt(v, x);
        siz[x] += siz[v];
        mx[x] = max(mx[x], siz[v]);
    }
    mx[x] = max(mx[x], n - siz[x]);
    if (mx[x] < mx[rt]) rt = x;
}
ull getTreeHash(int x, int fa) {
    siz[x] = 1;
    ull res = 1;
    for (auto v : g[x]) {
        if (v == fa) continue;
        res += getTreeHash(v, x) * prime[siz[v]];
        siz[x] += siz[v];
    }
    return res;
}
```

2.8 树链剖分

```cpp
const int N = 1e5 + 10;
vector<int> g[N];
int son[N], siz[N], dep[N], fat[N], dfn[N], rnk[N], top[N], tot;
void dfs1(int u, int fa) {
    son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;
    for (auto v : g[u]) {
        if (v == fa) continue;
        dfs1(v, u);
        siz[u] += siz[v];
        if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
    }
}

void dfs2(int u, int t) {
    rnk[dfn[u] = ++tot] = u; top[u] = t;
    if (son[u] == -1) return;
    dfs2(son[u], t);
    for (auto v : g[u]) {
        if (v != son[u] && v != fat[u]) dfs2(v, v);
    }
}
// ----------结合线段树操作
int querymax(int x, int y) {
    int ans = -1;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        ans = max(ans, st.query1(1, dfn[top[x]], dfn[x]));
        x = fa[top[x]];
    }
    if (dfn[x] > dfn[y]) swap(x, y);
    ans = max(ans, st.query1(1, dfn[x], dfn[y]));//在点权转化成边权时，dfn[x]+1开始
    return ans;
}

int querysum(int x, int y) {
    int ans = 0;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        ans += st.query2(1, dfn[top[x]], dfn[x]);
        x = fa[top[x]];
    }
    if (dfn[x] > dfn[y]) swap(x, y);
    ans += st.query2(1, dfn[x], dfn[y]);
    return ans;
}

void update(int x, int y, int c) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        st.update(1, dfn[top[x]], dfn[x], c);
        x = fa[top[x]];
    }
    if (dfn[x] > dfn[y]) swap(x, y);
    st.update(1, dfn[x], dfn[y], c);
}
```

2.9 树上 k-th 祖先 O(q+nlogn)

```
//
// Created by SANZONG on 2021/7/6.
//

#include "bits/stdc++.h"


using namespace std;
const int maxn = 6e5;
int head[maxn << 1];
int cnt;
#define ui unsigned int
ui s;

inline ui get(ui x) {
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    return s = x;
}

#define int long long
struct node {
    int next;
    int to;
} a[maxn << 1];

void add(int u, int v) {
    a[++cnt].next = head[u];
    a[cnt].to = v;
    head[u] = cnt;
}

int f[maxn][33];
int dep[maxn];
int son[maxn];
int dlen[maxn];     //每个点所在的长链长度，画一下就知道
int g[maxn];       //log

void dfs(int u) {
    dlen[u] = dep[u] = dep[f[u][0]] + 1;
    for (int i = 1; i <= 30; ++i) {
        f[u][i] = f[f[u][i - 1]][i - 1];
    }
    for (int i = head[u]; i; i = a[i].next) {
        int v = a[i].to;
        if (v == f[u][0]) continue;
        dlen[v] = dep[v] = dep[u] + 1;    //向下时先等于深度
        dfs(v);
        dlen[u] = max(dlen[u], dlen[v]);    //逆推时一直取子树最大深度
        if (dlen[v] > dlen[son[u]]) son[u] = v;
    }
}

int U[maxn], D[maxn], id[maxn], cnt_t, top[maxn];

void dfs_t(int u, int root) {
    id[u] = ++cnt_t;     //一条相同长链上的id是连续的。
    D[cnt_t] = u;          //dfs序为cnt_t的点的连顶向下数组。
    U[cnt_t] = root;       //dfs序为cnt_t的点的连顶向上数组。
    if (son[u]) {
        top[son[u]] = top[u];
        dfs_t(son[u], f[root][0]);
    }
    for (int i = head[u]; i; i = a[i].next) {
        int v = a[i].to;
        if (v == f[u][0] || v == son[u]) continue;
        top[v]  =v;                 //短链
        dfs_t(v, v);
    }
}

int getkfa(int x, int k) {
    if (k == 0) return x;
    x = f[x][g[k]];     //直接跳到链上
    k -= 1 << g[k];      //跳到f[x][g[k]]的消耗
    k -= dep[x] - dep[top[x]];  //x 从x跳到顶点
    x = top[x];
    return k >= 0 ? U[id[x]+k] : D[id[x]-k];
}

//int getkfa(int x,int k)
//{
//    if (k == 0) return x;
//    int d = dep[x] - k;
//    for (int i = 30; i >= 0; --i) {
//        if (dep[f[x][i]] > d) x = f[x][i];
//    }
//    return f[x][0];
//}
signed main() {
    ios::sync_with_stdio(0);
//    freopen("in.txt","r",stdin);
    int n, q, root;;
    cin >> n >> q >> s;
    g[0] = -1;
    for (int i = 1; i <= n; ++i) { g[i] = g[i >> 1] + 1; }   //求log
    for (int i = 1; i <= n; ++i) {
        cin >> f[i][0];
        if (f[i][0] == 0) {
            root = i;
        } else {
            add(f[i][0], i);
            add(i, f[i][0]);
        }
    }
    dep[root] = 1;
    dfs(root);
    top[root] = root;
    dfs_t(root, root);
//    for (int i = 1; i <= n; ++i) {
//        cout << id[i] << endl;
//    }
    int ans = 0;
    int res = 0;
    for (int i = 1; i <= q; ++i) {

        int xi = ((get(s) ^ ans) % n) + 1;
//        cout << (get(s) ^ ans) << endl;
        int ki = (get(s) ^ ans) % dep[xi];
        ans = getkfa(xi, ki);
//        cout << xi << ' ' << ki << ' ' << ans << endl;
        res ^= (i * ans);
    }
    cout << res << endl;
}
```

2.10 树上差分 (边差分)

```
边差分：  c[x] += val, c[y] += val, c[lca(x, y)] -= 2*val;

点差分：  c[x] += val, c[y] += val, c[lca(x, y)] -= val, c[fa[lca(x, y)]] -= val;
```

2.11 树上距离 (lca tarjan)

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
vector<int> query[N], query_id[N];

struct edge {
    int to, next, vi;
}e[N*2];

int T, n, q, h[N], cnt, ans[N], d[N], fa[N], vis[N];
```

```

void add_query(int u, int v, int id) {
    query[u].push_back(v), query_id[u].push_back(id);
    query[v].push_back(u), query_id[v].push_back(id);
}

void add(int u, int v, int w) {
    e[cnt].to = v;
    e[cnt].vi = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

int find(int x) {
    if (fa[x] == x) return x;
    else return fa[x] = find(fa[x]);
}

void tarjan(int x) {
    vis[x] = 1;
    for (int i = h[x]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (vis[v]) continue;
        d[v] = d[x] + e[i].vi;
        tarjan(v);
        fa[v] = x;
    }

    for (int i = 0; i < query[x].size(); i++) {
        int y = query[x][i], id = query_id[x][i];
        if (vis[y] == 2) {
            int lca = find(y);
            ans[id] = min(ans[id], d[x] + d[y] - 2 * d[lca]);
        }
    }
    vis[x] = 2;
}

int main() {
    scanf("%d", &T);
    while (T--) {
        scanf("%d %d", &n, &q);
        for (int i = 1; i <= n; i++) {
            h[i] = -1;
            d[i] = 0;
            query[i].clear(), query_id[i].clear();
            fa[i] = i;
            vis[i] = 0;
        }
        cnt = 0;
        for (int i = 1; i <= n - 1; i++) {
            int x, y, z;
            scanf("%d %d %d", &x, &y, &z);
            add(x, y, z), add(y, x, z);
        }
        for (int i = 1; i <= q; i++) {
            int x, y;
            scanf("%d %d", &x, &y);
            if (x == y) ans[i] = 0;
            else {
                add_query(x, y, i);
                ans[i] = 999999999;
            }
        }
        tarjan(1);
        for (int i = 1; i <= q; i++) {
            printf("%d\n", ans[i]);
        }
    }
}
```

2.12 树上距离（lca 倍增）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 4e4 + 10;

int h[N], dis[N], d[N], f[N][30];
int T, n, m, cnt, s;

struct edge {
    int to, next, vi;
}e[N<<1];

void add(int u, int v, int vi) {
    e[cnt].to = v;
    e[cnt].vi = vi;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

void dfs(int x, int fa) {
    f[x][0] = fa, d[x] = d[fa] + 1;
    for (int i = 1; i <= 20; i++) f[x][i] = f[f[x][i-1]][i-1];
    for (int i = h[x]; ~i; i = e[i].next) {
        int y = e[i].to;
        if (y != fa) dfs(y, x);
    }
}

int lca(int x, int y) {
    if (d[x] > d[y]) swap(x, y);
    for (int i = 20; i >= 0; i--) if (d[f[y][i]] >= d[x]) y = f[y][i];

    if (x == y) return y;
    for (int i = 20; i >= 0; i--) {
        if (f[y][i] != f[x][i]) {
            y = f[y][i];
            x = f[x][i];
        }
    }
    return f[x][0];
}

int main() {
    memset(h, -1, sizeof h);
    scanf("%d %d %d", &n, &m, &s);
    for (int i = 1; i <= n - 1; i++) {
        int a, b, c;
        scanf("%d %d", &a, &b);
        add(a, b, 1), add(b, a, 1);
    }
    dfs(s, 0);
    for (int i = 1; i <= m; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        printf("%d\n", lca(a, b));
    }
}
```

2.13 树上启发式合并

```
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const int INF = 0x3f3f3f3f;
const int N = 2e5 + 10;
const int M = 1e6 + 10;
const int MOD = 1e9 + 7;
int sz[N], son[N], h[N], tot, col[N], cnt[N], vis[N], mx;
ll sum, ans[N];
struct Edge {
    int to, next;
}e[M];

void add(int u, int v) {
    e[tot].to = v;
    e[tot].next = h[u];
    h[u] = tot++;
}

void dfs(int u, int fa) {
    sz[u] = 1;
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa) continue;
        dfs(v, u);
        sz[u] += sz[v];
        if (!son[u] || sz[v] > sz[son[u]])
            son[u] = v;
    }
}

void count(int u, int fa, int k) {
    //统计子树信息


    //
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa || vis[v]) continue;
        count(v, u, k);
    }
}

void dsu(int u, int fa, int keep) {
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa || son[u] == v) continue;
        dsu(v, u, 0);//查询轻儿子
    }
    if (son[u]) dsu(son[u], u, 1), vis[son[u]] = 1;//查询重儿子
    count(u, fa, 1);//统计子树信息
    //统计答案
    if (son[u]) vis[son[u]] = 0;
    if (!keep) count(u, fa, -1), mx = sum = 0;//撤回信息
}
```

2.14 k 级祖先

```
vector<int> g[N];
int dep[N], f[N][30];
void dfs(int x, int fa) {
    f[x][0] = fa; dep[x] = dep[fa] + 1;
    for (int i = 1; i <= 20; i++) {
        f[x][i] = f[f[x][i-1]][i-1];
    }
    for (auto &v : g[x]) {
        if (v == fa) continue;
        dfs(v, x);
    }
}
int find_Kth(int x, int k) {
    int bit = 0;
    while (k) {
        if (k & 1) x = f[x][bit];
        k >>= 1;
        bit++;
    }
    return x;
}
```

3 数据结构

3.1 Max XOR(tire)

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
const int M = 3000010;
int n, a[N];
int trie[M][2];
int cnt;
void insert(int x){
    int p = 0;
    for (int i = 30; ~i; i--){
        int w = x >> i & 1;
        if (!trie[p][w]) trie[p][w] = ++cnt;
        p = trie[p][w];
    }
}

int query(int x){
    int p = 0;
    int res = 0;
    for (int i = 30; ~i; i--){
        int w = x >> i & 1;

        if (trie[p][w ^ 1]){
            res += (1 << i);
            p = trie[p][w ^ 1];
        }
        else p = trie[p][w];
    }
    return res;
}
```

3.2 splay(待完善)

```cpp
#include <bits/stdc++.h>

#define ACM_LOCAL
#define fi first
#define se second
#define pb push_back
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int n, m, k, cnt;
int a[N];

struct SplayTree {
     int sz[N], ch[N][2], cnt[N], fa[N], rev[N], val[N], ad[N];
     int tot, rt;
#define rt ch[anc][0]
#define keynode ch[ch[rt][1]][0]
     const int anc = 0;
     void clear(int x) {sz[x] = cnt[x] = ch[x][0] = ch[x][1] = val[x] = ad[N] = 0;}
     void init() {tot = 0, clear(0);}
     int get(int x) {return ch[fa[x]][1] == x;}

     int newnode(int k) {
         int x = ++tot; clear(x);
         val[x] = k;
         sz[x] = cnt[x] = 1;
         return tot;
     }

     void setc(int x, int y, int d) { //基本点操作：设置x的d号儿子为y
         ch[x][d] = y;
         if (y) fa[y] = x;
         if (x) push_up(x);
     }

    void reverse(int x) {
        if (x == 0) return;//Necessary
        swap(ch[x][0], ch[x][1]);
        rev[x] ^= 1;
    }
    void add(int x, int k) {
        if (x == 0)return;
        val[x] += k;
        ad[x] += k;
    }

    void push_up(int x) {
        sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x];
    }

    void push_down(int x) {
        if (!x) return;
        if (rev[x]) {
            reverse(ch[x][0]);
            reverse(ch[x][1]);
            rev[x] = 0;
        }
        if (ad[x]) {
            add(ch[x][0], ad[x]);
            add(ch[x][1], ad[x]);
            ad[x] = 0;
        }
    }
    void rotate(int x) {
        int f = fa[x];
        int ff = fa[f];
        bool d = get(x);
        bool dd = get(f);
        setc(f, ch[x][!d], d);  //第一步：把与x原父节点同方向的子树取作原父节点f的子树
        setc(x, f, !d);         //第二步：使与x原父节点同方向的子树连接原父节点f
        setc(ff, x, dd);        //第三步：使得x替代原父节点f，变为新父节点点ff的子树
    }

    void splay(int x, int anc = 0) {//双旋
        if (x == 0)return;
        while (fa[x] != anc) {
            push_down(fa[x]);
            push_down(x);
            if (fa[fa[x]] != anc)rotate(get(x) == get(fa[x]) ? fa[x] : x);
            rotate(x);
        }
    }
    int rank(int k) {//排名
        int x = rt;
        while (1) {
            if (x == 0)return 0;
            if (k == val[x]) { splay(x); return x; }
            bool d = k > val[x];
            x = ch[x][d];
        }
        splay(x);
    }

    void ins(int k) {
        int x = rank(k);
        if (x) {cnt[x]++; sz[x]++; return;}
        int fa = anc; x = rt;
        bool d = 0;
        while (x) {
            fa = x;
            d = k > val[x];
            x = ch[x][d];
        }
        x = newnode(k);
        setc(fa, x, d);
        splay(x);
    }

    int Min(int x) {//子树中最小节点
        push_down(x);
        while (ch[x][0]) x = ch[x][0], push_down(x);
        return x;
    }
    int Max(int x) {//子树中最大节点
        push_down(x);
        while (ch[x][1]) x = ch[x][1], push_down(x);
        return x;
    }
    int kth(int k) {
        ++k;    //这里涉及到区间操作，我们在左右界各添加新节点，因此进入时要++k
        int x = rt;
        while (1) {
            push_down(x);
            if (sz[ch[x][0]] >= k) x = ch[x][0];
            else if (sz[ch[x][0]] + cnt[x] >= k) return x;
            else
            {
                k -= sz[ch[x][0]] + cnt[x];
                x = ch[x][1];
            }
        }
    }
    void del(int x) {
        splay(x);//转到根后便不再需要pushdown()
        if (ch[x][0] == 0)setc(anc, ch[x][1], 0);   //如果没有左子树，则直接把右子树放到树根
        else
        {
            setc(anc, ch[x][0], 0);                //第一步：把左子树放到树根
            splay(Max(ch[x][0]), anc);             //第二步：把左子树最大节点转到树根
            setc(rt, ch[x][1], 1);                 //第三步：把右子树接到树根上
        }
    }
    //查找操作：返回树中[l,r]区间段的根节点
    int segment(int l, int r) {
        splay(kth(l - 1), anc);
        splay(kth(r + 1), rt);
        return keynode;
    }
    //分离操作：把[l,r]区间段从树中分离
    int split(int l, int r) {
        int x = segment(l, r); fa[x] = 0;
        setc(ch[rt][1], 0, 0);
        push_up(rt);
        return x;
    }
    //合并操作，把子树x插入到pos位置的右侧
    void inspos(int x, int pos) {
        segment(pos + 1, pos);  //使得pos位于根，pos+1位于根的右子树
        setc(ch[rt][1], x, 0);
        push_up(rt);
    }
    //中序遍历
    void ldr(int x) {
        if (ch[x][0]) ldr(ch[x][0]);
        printf("%d ", x);
        if (ch[x][1]) ldr(ch[x][1]);
    }
    void add(int l, int r, int k) {//区间加
        int x = segment(l, r);
        add(x, k);
        splay(x);
    }

    void reverse(int l, int r) {//区间翻转
        int x = segment(l, r);
        reverse(x);
        splay(x);
    }



}spy;

void solve() {
    int n, opt, x;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);

}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("input", "r", stdin);
    freopen("output", "w", stdout);
#endif
    solve();
    return 0;
}
```

3.3 ST 表

```
namespace ST {
    int mi[N][21], ma[N][21], lg[N], a[N], gcd[N][21];
    int cmp1(int x, int y) {
        return a[x] < a[y] ? x : y;
    }
    int cmp2(int x, int y) {
        return a[x] > a[y] ? x : y;
    }
    void init(int n) {
        for (int i = 1; i <= n; i++) {
            cin >> a[i];
            gcd[i][0] = a[i];
            ma[i][0] = mi[i][0] = i, lg[i] = log2(i);
        }
        for (int i = 1; i <= 20; i++) {
            for (int j = 1; j + (1 << i) - 1 <= n; j++) {
                mi[j][i] = cmp1(mi[j][i - 1], mi[j + (1 << (i - 1))][i - 1]);
                ma[j][i] = cmp2(ma[j][i - 1], ma[j + (1 << (i - 1))][i - 1]);
                gcd[j][i] = __gcd(gcd[j][i - 1], gcd[j + (1 << (i - 1))][i - 1]);
            }
        }
    }
    int qry_mi(int l, int r) {
        int k = lg[r - l + 1];
        return a[cmp1(mi[l][k], mi[r - (1 << k) + 1][k])];
    }
    int qry_ma(int l, int r) {
        int k = lg[r - l + 1];
        return a[cmp2(ma[l][k], ma[r - (1 << k) + 1][k])];
    }
    int qry_pmi(int l, int r) {
        int k = lg[r - l + 1];
        return cmp1(mi[l][k], mi[r - (1 << k) + 1][k]);
    }
    int qry_pma(int l, int r) {
        int k = lg[r - l + 1];
        return cmp2(ma[l][k], ma[r - (1 << k) + 1][k]);
    }
    int qry_gcd(int l, int r) {
        int k = lg[r - l + 1];
        return __gcd(gcd[l][k], gcd[r - (1 << k) + 1][k]);
    }
}
```

3.4

```
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
```

```cpp
typedef long long ll;
typedef pair<int, int> PII;
const int N = 1e5 + 100, M = 5e5 + 5, INF = 0x3f3f3f3f;
mt19937 rnd(233);
struct Treap {
    int lc[N], rc[N], val[N], key[N], sz[N];
    int cnt, root;

    inline int newnode(int v) {
        val[++cnt] = v;
        key[cnt] = rnd();
        sz[cnt] = 1;
        return cnt;
    }

    inline void update(int now) {
        sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
    }

    void split(int now, int v, int &x, int &y) {
        if (!now) x = y = 0;
        else {
            if (val[now] <= v) {
                x = now;
                split(rc[now], v, rc[now], y);
            } else {
                y = now;
                split(lc[now], v, x, lc[now]);
            }
            update(now);
        }
    }

    int merge(int x, int y) {
        if (!x || !y) return x + y;
        if (key[x] > key[y]) {
            rc[x] = merge(rc[x], y);
            update(x);
            return x;
        } else {
            lc[y] = merge(x, lc[y]);
            update(y);
            return y;
        }
    }

    int x, y, z;

    inline void insert(int val) {
        split(root, val, x, y);
        root = merge(merge(x, newnode(val)), y);
    }

    inline void del(int val) {
        split(root, val, x, z);
        split(x, val - 1, x, y);
        y = merge(lc[y], rc[y]);
        root = merge(merge(x, y), z);
    }

    inline int getrank(int v) { //查询v的排名
        split(root, v - 1, x, y);
        int rank = sz[x] + 1;
        root = merge(x, y);
        return rank;
    }

    inline int getnum(int rank) {//查询排名第rank的数
        int now = root;
        while (now) {
            if (sz[lc[now]] + 1 == rank)
                break;
            else if (sz[lc[now]] >= rank)
                now = lc[now];
            else {
                rank -= sz[lc[now]] + 1;
                now = rc[now];
            }
        }
        return val[now];
    }

    inline int pre(int v) {
        split(root, v - 1, x, y);
        int now = x;
        while (rc[now])
            now = rc[now];
        int pre = val[now];
        root = merge(x, y);
        return pre;
    }

    inline int nxt(int v) {
        split(root, v, x, y);
        int now = y;
        while (lc[now])
            now = lc[now];
        int nxt = val[now];
        root = merge(x, y);
        return nxt;
    }
}fhq;
```

3.5 Treap(文艺平衡树)

```
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const int N = 1e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
mt19937 rnd(233);
struct Treap {
    int key[N], lc[N], rc[N], val[N], sz[N];
    bool reverse[N];
    ll sum[N], tag[N];
    int cnt, rt;

    inline int newnode(int v) {
        val[++cnt] = v;
        sz[cnt] = 1;
        key[cnt] = rnd();
        reverse[cnt] = false;
        return cnt;
    }
    inline void push_up(int now) {
        sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
        sum[now] = sum[lc[now]] + sum[rc[now]] + val[now];
    }
    inline void push_down(int now) {
        if (reverse[now]) {
            swap(lc[now], rc[now]);
            reverse[lc[now]] ^= 1;
            reverse[rc[now]] ^= 1;
            reverse[now] = false;
        }
        if (tag[now]) {
            sum[lc[now]] += sz[lc[now]] * tag[now];
            val[lc[now]] += tag[now];
            tag[lc[now]] += tag[now];

            sum[rc[now]] += sz[rc[now]] * tag[now];
            val[rc[now]] += tag[now];
            tag[rc[now]] += tag[now];
        }
        tag[now] = 0;
    }
    void split(int now, int siz, int &x, int &y) {
        if (!now) x = y = 0;
        else {
            push_down(now);
            if (sz[lc[now]] < siz) {
                x = now;
                split(rc[now], siz-sz[lc[now]]-1, rc[now], y);
            } else {
                y = now;
                split(lc[now], siz, x, lc[now]);
            }
            push_up(now);
        }
    }
    int merge(int x, int y) {
        if (!x || !y) return x + y;
        if (key[x] < key[y]) {
            push_down(x);
            rc[x] = merge(rc[x], y);
            push_up(x);
            return x;
        } else {
            push_down(y);
            lc[y] = merge(x, lc[y]);
            push_up(y);
            return y;
        }
    }
    void rev(int l, int r) {
        int x, y, z;
        split(rt, l-1, x, y);
        split(y, r-l+1, y, z);
        reverse[y] ^= 1;
        rt = merge(merge(x, y), z);
    }
    void ldr(int now) {
        if (!now) return;
        push_down(now);
        ldr(lc[now]);
        printf("%d ", val[now]);
        ldr(rc[now]);
    }
    void add(int l, int r, int v) {
        int x, y, z;
        split(rt, l-1, x, y);
        split(y, r-l+1, y, z);
        tag[y] += v;
        sum[y] += sz[y] * v;
        val[y] += v;
        rt = merge(merge(x, y), z);
    }
    ll query(int l, int r) {
        ll ans = 0;
        int x, y, z;
        split(rt, l-1, x, y);
        split(y, r-l+1, y, z);
        ans = sum[y];
        rt = merge(merge(x, y), z);
        return ans;
    }
    void insert(int pos, int v) {
        int x, y, z;
        split(rt, pos-1, x, y);
        z = newnode(v);
        rt = merge(merge(x, z), y);
    }
}fhq;
```

3.6 并查集

```
//带权并查集
const int N = 1e5 + 10;
int fa[N], d[N];
int find(int x) {
    int par = fa[x];
    fa[x] = find(fa[x]);
    d[x] += d[par];
}

//合并x和y，并计算路径长度
void merge(int x, int y, int w) {//w代表 x->y 路径长
    int fx = find(x), fy = find(y);
    if (fx != fy) fa[fx] = fy, d[fx] = d[y] - d[x] + w;
    else {
        int dis = d[x] - d[y]; // 如果x和y已经存在关系，dis代表x到y的距离
    }
}

//种类并查集

find(x) --- x的同类
find(x+n) ---
find(x+n+n....) ---

//建立虚点
void merge(int x, int y) {//正常的合并操作
    int fx = find(x), fy = find(y);
    ans[fy] += ans[fx];
    num[fy] += num[fx];
    fa[fx] = fy;
}

void move(int x) {//id[x] 代表x目前的编号
    int fx = find(id[x]);
    ans[fx] -= x;//原来集合中减去x的贡献
    num[fx]--;
    id[x] = ++n;//建立新的节点
    ans[id[x]] = x;
    num[id[x]] = 1;
    fa[id[x]] = id[x];
}

//可撤销并查集
struct Undo_dsu {
    stack<PII> st;
    int fa[N], siz[N];
    void init() {
        while (st.size()) st.pop();
        for (int i = 1; i <= n; i++) fa[i] = i, siz[i] = 1;
    }
    int find(int x) {return fa[x] == x ? x : find(fa[x]);}
    bool merge(int x, int y) {
        int fx = find(x), fy = find(y);
        if (fx == fy) return false;
        if (siz[fx] > siz[fy]) swap(fx, fy), swap(x, y);
        siz[fy] += siz[fx], fa[fx] = fy;
        st.push({fx, fy});
        return true;
    }
    void undo() {
        PII now = st.top();
        fa[now.fi] = now.fi;
        siz[now.se] -= siz[now.fi];
        st.pop();
    }
}dsu;
```

3.7 单调队列

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000010;
int q[N], a[N], n, k;
void deque_max() {
    int hh = 1, tt = 0;
    for (int i = 1; i <= n; i ++) {
        while(hh <= tt && q[hh] + k <= i) hh ++;
        while(hh <= tt && a[q[tt]] < a[i]) tt --;
        q[++tt] = i;
        if (i >= k) cout << a[q[hh]] << " ";
    }
}

void deque_min() {
    int hh = 1, tt = 0;
    for (int i = 1; i <= n; i ++) {
        while(hh <= tt && q[hh] + k <= i) hh ++;
        while(hh <= tt && a[q[tt]] > a[i]) tt --;
        q[++tt] = i;
        if (i >= k) cout << a[q[hh]] << " ";
    }
}
```

3.8 单调栈

```
#include <bits/stdc++.h>
using namespace std;
const int N = 3e6 + 10;
int a[N], n, idx[N];
stack<int> stk;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) {
        while (stk.size() && a[stk.top()] < a[i]) {
            idx[stk.top()] = i;
            stk.pop();
        }
        stk.push(i);
    }
    for (int i = 1; i <= n; i++) cout << idx[i] << " ";
}
```

3.9 笛卡尔树

```
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const int N = 3e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int n, m, k;
struct Car_tree {
    int lc[N], rc[N], stk[N], top, val[N];
    void init(int n) {
        for (int i = 0; i <= n; i++) lc[i] = rc[i] = 0;
    }
    int build(int n) {
        int rt;
        for (int i = 1; i <= n; i++) {
            scanf("%d", &val[i]);
            while (top && val[stk[top]] > val[i]) {
                lc[i] = stk[top], top--;
            }
            if (top) rc[stk[top]] = i;
            stk[++top] = i;
        }
        while (top) rt = stk[top--];
        return rt;
    }
}tr;
```

3.10 静态主席树

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5 + 5;

int a[N], b[N], rt[N];
int n, m;
struct Hash {
    int b[N], tot;
    void init() {tot = 0;}
    void insert(int x) {b[++tot] = x;}
    void build() {
        sort(b+1, b+1+tot);
        tot = unique(b+1, b+tot+1) - (b+1);
    }
    int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
}ha;
struct {
    int t[N << 5], lc[N << 5], rc[N << 5];
    int NodeNum = 0;
    ll sum[N << 5];
    int build(int l, int r) {
        int num = ++NodeNum;
        if (l != r) {
            int mid = (l + r) >> 1;
            lc[num] = build(l, mid);
            rc[num] = build(mid + 1, r);
        }
        return num;
    }
    int update(int pre, int l, int r, int x) {
        int num = ++NodeNum;
        lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
        if (l != r) {
            int mid = (l + r) >> 1;
            if (x <= mid) lc[num] = update(lc[pre], l, mid, x);
            else rc[num] = update(rc[pre], mid + 1, r, x);
        }
        return num;
    }
    int Qry_Kth_Num(int u, int v, int l, int r, int k) {
        if (l == r) return ha.b[l];
        int mid = (l + r) >> 1, num = t[lc[v]] - t[lc[u]];
        if (num >= k) return Qry_Kth_Num(lc[u], lc[v], l, mid, k);
        else return Qry_Kth_Num(rc[u], rc[v], mid + 1, r, k-num);
    }
    ll Qry_Kth_Sum(int u, int v, int l, int r, int k) {//k大数之和
        if (l == r) return 1ll*ha.b[l] * k;
        int mid = (l + r) >> 1, num = t[rc[v]] - t[rc[u]];
        if (num >= k) return Qry_Kth_Sum(rc[u], rc[v], mid+1, r, k);
        else return sum[rc[v]] - sum[rc[u]] + Qry_Kth_Sum(lc[u], lc[v], l, mid, k-num);
    }
    int Binary_Search(int left, int right, int val) {//查找小于等于val的个数
        int l = 1, r = right - left + 1;
        while (l <= r) {
            int mid = l + r >> 1;
            int num = Qry_Kth_Num(rt[left - 1], rt[right], 1, ha.tot, mid);
            if (num > val) r = mid - 1;
            else l = mid + 1;
        }
        return r;
    }
}hjt;
```

3.11 珂朵莉树

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <stack>
#include <queue>
#include <map>
#include <vector>
#include <cstdio>
#include <cmath>
#include <ctime>
#include <bitset>
#include <unordered_map>
#include <string>
#include <set>
using namespace std;
#define ACM_LOCAL
#define  IT set<node> ::iterator
inline int read(){
   int s=0,w=1;
   char ch=getchar();
   while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
   while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
   return s*w;
}
const int N = 3e5 + 10;
const int INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
typedef pair<int, int> PII;
typedef long long ll;
typedef unsigned long long ull;
int ans;
struct node {
    int l, r;
    mutable ll val;
    node(int L, int R = -1, ll V = 0):l(L), r(R), val(V) {}
    bool operator < (const node &rhs) const {
        return l < rhs.l;
    }
};

set<node> s;
IT split(int pos) {
    IT it = s.lower_bound(node(pos));
    if (it != s.end() && it->l == pos) return it;
    it--;
    int L = it->l, R = it->r;
    ll val = it->val;
    s.erase(it);
    s.insert(node(L, pos-1, val));
    return s.insert(node(pos, R, val)).first;
}

void push_down(int l, int r, ll val) {
    IT itr = split(r+1), itl = split(l);
    s.erase(itl, itr);
    s.insert(node(l, r, val));
}

void add(int l, int r, ll val) {
    IT itr = split(r+1), itl = split(l);
    for (; itl != itr; ++itl)
        itl->val += val;
}

ll rank(int l, int r, int k) {
    vector<pair<ll, int>> vp;
    IT itr = split(r+1), itl = split(l);
    vp.clear();
    for (; itl != itr; ++itl) {
        vp.push_back(pair<ll, int>(itl->val, itl->r - itl->l + 1));
    }
    sort(vp.begin(), vp.end());
    for (vector<pair<ll,int> >::iterator it = vp.begin(); it != vp.end(); ++it) {
        k -= it->second;
        if (k <= 0)
            return it->first;
    }
}

ll ksm(ll a, ll b, ll mod) {
    ll res = 1, ans = a % mod;
    while (b) {
        if (b & 1)
            res = res * ans % mod;
        ans = ans * ans % mod;
        b >>= 1;
    }
    return res;
}

ll sum(int l, int r, int ex, int mod) {
    IT itr = split(r+1), itl = split(l);
    ll res = 0;
    for (; itl != itr; ++itl)
        res = (res + (ll)(itl->r - itl->l + 1) * ksm(itl->val, ll(ex), ll(mod))) % mod;
    return res;
}
```

3.12 可持久化并查集

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <queue>
#include <stack>
#include <string>
#include <set>
#include <map>
#include <bitset>
using namespace std;
#define ACM_LOCAL

const int INF = 0x3f3f3f3f;
const int N = 2e5 + 5;
typedef pair<int, int> PII;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

struct node {
    int l, r;
    int val;
}t[N * 40 * 2];

int cnt, tot, rootdep[N], rootfa[N], n, m;

void build(int l, int r, int &now) {
    now = ++cnt;
    if (l == r) {
        t[now].val = ++tot;
        return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, t[now].l);
    build(mid + 1, r, t[now].r);
}

void update(int l, int r, int ver, int &now, int pos, int val) {
    t[now = ++cnt] = t[ver];
    if (l == r) {
        t[now].val = val;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid) update(l, mid, t[ver].l, t[now].l, pos, val);
    else update(mid + 1, r, t[ver].r, t[now].r, pos, val);
}

int query(int l, int r, int &now, int pos) {
    if (l == r) return t[now].val;
    int mid = (l + r) >> 1;
    if (pos <= mid) return query(l, mid, t[now].l, pos);
    else return query(mid + 1, r, t[now].r, pos);
}

int find(int ver, int x) {
    int fx = query(1, n, rootfa[ver], x);
    return fx == x ? x : find(ver, fx);
}

void merge(int ver, int x, int y) {
    x = find(ver-1, x);
    y = find(ver-1, y);
    if (x == y) {
        rootfa[ver] = rootfa[ver-1];
        rootdep[ver] = rootdep[ver-1];
    }
    else {
        int depx = query(1, n, rootdep[ver-1], x);
        int depy = query(1, n, rootdep[ver-1], y);
        if (depx < depy) {
            update(1, n, rootfa[ver-1], rootfa[ver], x, y);
            rootdep[ver] = rootdep[ver-1];
        }
        else if (depx > depy) {
            update(1, n, rootfa[ver-1], rootfa[ver], y, x);
            rootdep[ver] = rootdep[ver-1];
        }
        else {
            update(1, n, rootfa[ver-1], rootfa[ver], x, y);
            update(1, n, rootdep[ver-1], rootdep[ver], y, depy+1);
        }
    }
}
```

3.13 可持久化数组

```
#include <bits/stdc++.h>
using namespace std;
#define ACM_LOCAL
const int N = 1e6 + 5;
typedef pair<int, int> PII;
typedef long long ll;

inline int read(){
   int s=0,w=1;
   char ch=getchar();
   while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
   while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
   return s*w;
}
int NodeNum, n, m, a[N], rt[N];

struct Node{
    int t[N << 5], lc[N << 5], rc[N << 5];

    void build(int l, int r, int &now) {
        now = ++NodeNum;
        if (l == r) t[now] = a[l];
        else {
            int mid = (l + r) >> 1;
            build(l, mid, lc[now]);
            build(mid + 1, r, rc[now]);
        }
    }
    void update(int pre, int l, int r, int &now, int pos, int val) {
        now = ++NodeNum;
        lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
        if (l == r) t[now] = val;
        else {
            int mid = (l + r) >> 1;
            if (mid >= pos) update(lc[pre], l, mid, lc[now], pos, val);
            else update(rc[pre], mid + 1, r, rc[now], pos, val);
        }
    }
    void query(int pre, int l, int r, int &now, int pos) {
        now = ++NodeNum;
        lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
        if (l == r) printf("%d\n", t[now]);
        else {
            int mid = (l + r) >> 1;
            if (mid >= pos) query(lc[pre], l, mid, lc[now], pos);
            else query(rc[pre], mid + 1, r, rc[now], pos);
        }
    }
}T;


void solve() {
    n = read(), m = read();
    for (int i = 1; i <= n; i++) a[i] = read();

    T.build(1, n, rt[0]);
    for (int i = 1; i <= m; i++) {
        int opt, id, x, y;
        id = read(), opt = read();
        if (opt == 1) {
            x = read(), y = read();
            T.update(rt[id], 1, n, rt[i], x, y);
        }
        else {
            x = read();
            T.query(rt[id], 1, n, rt[i], x);
        }
    }

}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    solve();
    return 0;
```

```
}
```

3.14 平衡树（pb$_ds$

```
#include "bits/stdc++.h"
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
#define int long long
tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> t;
signed main()
{
    int n;
    cin >> n;
    while (n--)
    {
        int k,s;
        cin >> k >> s;
        if (k == 1)
        {
            cout << t.order_of_key(s)+1 << endl;         //order_of_key有几个比s小
        }
        else if (k == 2){
            cout << *t.find_by_order(s-1)  << endl;       //order是有几个比s小，s-1个比s小
        } else if (k == 3)
        {
            if (t.find_by_order(t.order_of_key(s)-1) != t.end())
                cout << *t.find_by_order(t.order_of_key(s)-1) << endl;
            else
                cout <<  -2147483647 << endl;
        } else if (k == 4)
            if (t.upper_bound(s) != t.end())
                cout << *t.upper_bound(s) << endl;        //未找到为t.end()
            else
                cout << 2147483647 << endl;
        else if (k == 5)
            t.insert(s);
    }
}
```

3.15 扫描线

```
#include <bits/stdc++.h>
using namespace std;
#define ACM_LOCAL
#define lc u << 1
#define rc u << 1 | 1
const int N = 2e5 + 5;
const int INF = 0x3f3f3f3f;
typedef unsigned long long ull;
typedef pair<int, int> PII;
typedef long long ll;

int n, m, p;
int v[N << 1];
struct node {
    int l, r;
    ll len;
    int cover;
}t[N << 3];

struct L{
    int x;
    int y1, y2;
    int state;
    bool operator < (L &rhs) {
        return x < rhs.x;
    }
}line[N << 1];

void push_up(int u) {
    if (t[u].cover) t[u].len = t[u].r - t[u].l;
    else t[u].len = t[lc].len + t[rc].len;
}

void build(int u, int l, int r) {
    t[u].l = v[l], t[u].r = v[r];
    if (r - l <= 1) return;
    int mid = (l + r) >> 1;
    build(lc, l, mid);
    build(rc, mid, r);
    push_up(u);
}

void update(int u, int ql, int qr, int v) {
    if (ql <= t[u].l && qr >= t[u].r) {
        t[u].cover += v;
        push_up(u);
        return;
    }
    if (ql < t[rc].l) update(lc, ql, qr, v);
    if (qr > t[lc].r) update(rc, ql, qr, v);
    push_up(u);
}

void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        v[i] = y1, v[i+n] = y2;
        line[i] = {x1, y1, y2, 1};
        line[i + n] = {x2, y1, y2, -1};
    }
    sort(v+1, v+(n<<1)+1);
    sort(line+1, line+(n<<1)+1);
    build(1, 1, n<<1);
    ll ans = 0;
    for (int i = 1; i <= n<<1; i++) {
        ans += 1ll*t[1].len * (line[i].x - line[i-1].x);
        update(1, line[i].y1, line[i].y2, line[i].state);
    }
    cout << ans << endl;
}

```

```
signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    solve();
    return 0;
}
```

3.16 树状数组

```
//-------------------------------一维树状数组
struct BIT {
    int c[N];
    int lowbit(int x) {return x&-x;}
    void add(int x, int val) {for (int i = x; i < N; i += lowbit(i)) c[i] += val;}
    int ask(int x) {int res = 0; for (int i = x; i > 0; i -= lowbit(i)) res += c[i]; return res;}
};

//-----------------------------二维树状数组
int c[N][N];

//单点修改，区间查询
int lowbit(int x) {return x & -x;}
void add(int x, int y, int v) {
    for (int i = x; i <= N; i += lowbit(i))
        for (int j = y; j <= N; j += lowbit(j))
            c[i][j] += v;
}

ll query(int x, int y) {
    ll res = 0;
    for (int i = x; i > 0; i -= lowbit(i))
        for (int j = y; j > 0; j -= lowbit(j))
            res += c[i][j];
    return res;
}

//区间修改，单点查询
void add_s(int x1, int y1, int x2, int y2, int v) {
    add(x2+1, y2+1, v);
    add(x1, y1, v);
    add(x1, y2+1, -v);
    add(x2+1, y1, -v);
}
//区间修改，区间查询
void add(int x,int y,int v) {
    for(int i = x; i<= n;i += lowbit(i))
        for(int j=y;j<=m;j+=lowbit(j)) {
            t1[i][j] += v;
            t2[i][j] += v*x;
            t3[i][j] += v*y;
            t4[i][j] += v*x*y;
        }
}
void modify(int x1,int y1,int x2,int y2,int v) {
    add(x1,y1,v);
    add(x1,y2+1,-v);
    add(x2+1,y1,-v);
    add(x2+1,y2+1,v);
}
int ask(int x, int y) {
    int res = 0;
    for(int i = x;i;i -= lowbit(i))
        for(int j=y;j;j -= lowbit(j))
            res += (x+1)*(y+1)*t1[i][j]-(y+1)*t2[i][j]-(x+1)*t3[i][j]+t4[i][j];
    return res;
}
int query(int x1,int y1,int x2,int y2) {
    return ask(x2,y2)-ask(x2,y1-1)-ask(x1-1,y2)+ask(x1-1,y1-1);
}
```

3.17 树状数组套主席树

```
//单点修改动态区间k大
#include <bits/stdc++.h>
#define lowbit(i) i & -i
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const ll INF = 1e18;
const int N = 1e5 + 10;
const int M = 1e6 + 10;
const int MOD = 1e9 + 7;
struct Segtree {
    int ls, rs;
    int sum;
}t[N * 400];
int tot, n, m, rt[N], a[N], cnt[2], tmp[2][20], b[N];
void modify(int &now, int l, int r, int pos, int val) {
    if (!now) now = ++tot;
    if (l == r) {
        t[now].sum += val;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid) modify(t[now].ls, l, mid, pos, val);
    else modify(t[now].rs, mid + 1, r, pos, val);
    t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
}
void prepare_modify(int x, int val) {
    for (int i = x; i <= n; i += lowbit(i)) {
        modify(rt[i], 1, 10, a[x], val);//预处理出修改哪log棵主席树
    }
}
int query(int l, int r, int k) {
    if (l == r) return l;
    int mid = (l + r) >> 1, sum = 0;
    for (int i = 1; i <= cnt[1]; i++) sum += t[t[tmp[1][i]].ls].sum;
    for (int i = 1; i <= cnt[0]; i++) sum -= t[t[tmp[0][i]].ls].sum;
    if (k <= sum) {
        for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].ls;
        for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].ls;
        return query(l, mid, k);
    } else {
        for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].rs;
        for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].rs;
        return query(mid + 1, r, k - sum);
    }
}
int prepare_query(int l, int r, int k) {//处理出需要进行加减操作的log棵主席树
    memset(tmp, 0, sizeof tmp);
    memset(cnt, 0, sizeof cnt);
    for (int i = r; i > 0; i -= lowbit(i)) tmp[1][++cnt[1]] = rt[i];
    for (int i = l - 1; i > 0; i -= lowbit(i)) tmp[0][++cnt[0]] = rt[i];
    return query(1, 10, k);
}
void solve() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n; i++) prepare_modify(i, 1);
    while(m--) {
        char op; cin >> op;
        if (op == 'Q') {
            int l, r, k; cin >> l >> r >> k;
            cout << prepare_query(l, r, k) << endl;
        } else {
            int x, y; cin >> x >> y;
            prepare_modify(x, -1);
            a[x] = y;
            prepare_modify(x, 1);
        }
    }
}

//区间修改，区间K大
#include <bits/stdc++.h>
#define lowbit(i) i & -i
#define Debug(x) cout << x << endl
#define fi first
#define se second
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
const ll INF = 1e18;
const int N = 5e4 + 10;
const int M = 1e6 + 10;
const int MOD = 1e9 + 7;
struct Segment {
    int ls, rs;
    ll sum, lazt;
} t[N * 400];
int rt[N << 2], tot, n, m, b[N], len;

void push_down(int now, int l, int r) {
    if (t[now].lazt) {
        int mid = (l + r) >> 1;
        if (!t[now].ls) t[now].ls = ++tot;
        if (!t[now].rs) t[now].rs = ++tot;
        t[t[now].ls].lazt += t[now].lazt;
        t[t[now].rs].lazt += t[now].lazt;
        t[t[now].ls].sum += 1ll*(mid - l + 1) * t[now].lazt;
        t[t[now].rs].sum += 1ll*(r - mid) * t[now].lazt;
        t[now].lazt = 0;
    }
}

void update(int &now, int ql, int qr, int l, int r) {
    if (!now) now = ++tot;
    if (ql <= l && qr >= r) {
        t[now].sum += r - l + 1;
        t[now].lazt++;
        return;
    }
    push_down(now, l, r);
    int mid = (l + r) >> 1;
    if (ql <= mid) update(t[now].ls, ql, qr, l, mid);
    if (qr > mid) update(t[now].rs, ql, qr, mid + 1, r);
    t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
}

void add(int u, int ql, int qr, int pos, int l, int r) {
    update(rt[u], ql, qr, 1, n);
    if (l == r) return;
    int mid = (l + r) >> 1;
    if (pos <= mid) add(u << 1, ql, qr, pos, l, mid);
    else add(u << 1 | 1, ql, qr, pos, mid + 1, r);
}

ll getsum(int &now, int ql, int qr, int l, int r) {
    if (!now) return 0;
    if (ql <= l && qr >= r) return t[now].sum;
    push_down(now, l, r);
    int mid = (l + r) >> 1;
    ll ans = 0;
    if (ql <= mid) ans += getsum(t[now].ls, ql, qr, l, mid);
    if (qr > mid) ans += getsum(t[now].rs, ql, qr, mid + 1, r);
    return ans;
}

int query(int u, int ql, int qr, ll k, int l, int r) {
    if (l == r) return b[l];
    int mid = (l + r) >> 1;
    ll num = getsum(rt[u<<1|1], ql, qr, 1, n);
    if (k > num) return query(u<<1, ql, qr, k - num, l, mid);
    else return query(u<<1|1, ql, qr, k, mid + 1, r);
}

struct Query {
    ll op, l, r, c;
} q[N];

void solve() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        cin >> q[i].op;
        if (q[i].op == 1) {
            cin >> q[i].l >> q[i].r >> q[i].c;
            b[++len] = q[i].c;
        } else {
            cin >> q[i].l >> q[i].r >> q[i].c;
        }
    }
    sort(b + 1, b + len + 1);
    len = unique(b + 1, b + len + 1) - b - 1;
    for (int i = 1; i <= m; i++) {
        if (q[i].op == 1) {
            q[i].c = lower_bound(b + 1, b + len + 1, q[i].c) - b;
            add(1, q[i].l, q[i].r, q[i].c, 1, len);
        } else {
            printf("%d\n", query(1, q[i].l, q[i].r, q[i].c, 1, len));
        }
    }
}
```

3.18 替罪羊树（平衡树）

```
#include<bits/stdc++.h>
#define lc (son[p][0])
#define rc (son[p][1])
using namespace std;
//#define ACM_LOCAL
const int N = 4e5 + 10;
struct SGT {
    const double alpha = 0.75;
    int st[N], top, tot, rt, son[N][2], fa[N], val[N], siz[N], all[N];
    bool in[N];

    inline int get() { return top ? st[top--] : ++tot; }

    void del(int t) { st[++top] = t; }

    inline void pushup(int p) { siz[p] = siz[lc] + siz[rc] + in[p], all[p] = all[lc] + all[rc] + 1; }

    inline bool check(int p) { return (all[lc] >= all[p] * alpha) || (all[rc] >= all[p] * alpha); }

    inline int newnode(int v = 0, int pa = 0) {
        int p = get();
        lc = rc = 0, val[p] = v, siz[p] = all[p] = 1, in[p] = 1, fa[p] = pa;
        return p;
    }

    inline void getpos(int p, vector<int> &v) {
        if (!p)return;
        getpos(lc, v);
        if (in[p])v.push_back(p);
        else del(p);
        getpos(rc, v);
    }

    inline int build(int l, int r, vector<int> v) {
        if (l >= r)return 0;
        int mid = l + r >> 1, p = v[mid];
        lc = build(l, mid, v), rc = build(mid + 1, r, v), fa[lc] = fa[rc] = p;
        return pushup(p), p;
    }

    inline void rebuild(int &p) {
        static vector<int> v;
        v.clear();
        int pa = fa[p];
        getpos(p, v);
        fa[(p = build(0, v.size(), v))] = pa;
    }

    inline int rank(int v) {
        int p = rt, ret = 1;
        while (p) {
            if (v <= val[p])p = lc;
            else ret += siz[lc] + in[p], p = rc;
        }
        return ret;
    }

    inline int kth(int k) {
        int p = rt;
        while (p) {
            if (siz[lc] + 1 == k && in[p]) break;
            if (siz[lc] >= k)p = lc;
            else k -= siz[lc] + in[p], p = rc;
        }
        return val[p];
    }

    inline int insert(int &p, int v) {
        if (!p)return p = newnode(v), 0;
        ++siz[p], ++all[p];
        int ret;
        ret = insert(v <= val[p] ? lc : rc, v), pushup(p);
        if (check(p))ret = p;
        return ret;
    }

    inline void insert(int v) {
        int p = insert(rt, v);
        if (!p)return;
        if (p == rt)rebuild(rt);
        else {
            int f = fa[p];
            if (p == son[f][0])rebuild(son[f][0]);
            else rebuild(son[f][1]);
        }
    }

     void erase(int p, int k) {
        --siz[p];
        if (in[p] && k == siz[lc] + in[p]) {
            in[p] = 0;
            return;
        }
        if (k <= siz[lc])erase(lc, k);
        else erase(rc, k - siz[lc] - in[p]);
    }
```

```

     void erase(int v) {
        erase(rt, rank(v));
        if (siz[rt] < alpha * all[rt])rebuild(rt);
    }

     int pre(int v) { return kth(rank(v) - 1); }

     int suf(int v) { return kth(rank(v + 1)); }
} tzy;
```

3.19 线段树（区间最长连续段）

```
int rt[N], cnt;//主席树版本
struct Tree {
    int ls, rs;
    int lsum, rsum, tsum;
}hjt[N*50];
void push_up(int now, int l, int r) {
    int mid = (l + r) >> 1;
    hjt[now].lsum = hjt[hjt[now].ls].lsum;
    if (hjt[hjt[now].ls].lsum == mid - l + 1) hjt[now].lsum += hjt[hjt[now].rs].lsum;
    hjt[now].rsum = hjt[hjt[now].rs].rsum;
    if (hjt[hjt[now].rs].rsum == r - mid) hjt[now].rsum += hjt[hjt[now].ls].rsum;
    hjt[now].tsum = max(hjt[hjt[now].ls].tsum, hjt[hjt[now].rs].tsum);
    hjt[now].tsum = max(hjt[now].tsum, hjt[hjt[now].ls].rsum + hjt[hjt[now].rs].lsum);
}
void modify(int &now, int pre, int l, int r, int pos, int val) {
    now = ++cnt;
    hjt[now] = hjt[pre];
    if (l == r) {
        hjt[now].lsum = hjt[now].rsum = hjt[now].tsum = 1;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid) modify(hjt[now].ls, hjt[pre].ls, l, mid, pos, val);
    else modify(hjt[now].rs, hjt[pre].rs, mid+1, r, pos, val);
    push_up(now, l, r);
}
int query(int now, int l, int r, int ql, int qr) {
    if (ql <= l && qr >= r) return hjt[now].tsum;
    int mid = (l + r) >> 1;
    int ans = 0;
    if (ql <= mid) ans = max(ans, query(hjt[now].ls, l, mid, ql, qr));
    if (qr > mid) ans = max(ans, query(hjt[now].rs, mid+1, r, ql, qr));
    ans = max(ans, min(mid-ql+1, hjt[hjt[now].ls].rsum) + min(qr-mid, hjt[hjt[now].rs].lsum));
    return ans;
}
```

3.20 线段树 (最长子序和)

```
#include<bits/stdc++.h>
using namespace std;
const int N=5e5+1000;
int n,m,x,y;
int a[N];
```

```
struct Node {
    int l,r;
    int sum;
    int lmax;
    int rmax;
    int tmax;
}tr[N<<2];

void push_up(Node &u,Node &l,Node &r) {
    u.sum=l.sum+r.sum;
    u.lmax=max(l.lmax,l.sum+r.lmax);
    u.rmax=max(r.rmax,r.sum+l.rmax);
    u.tmax=max(max(l.tmax,r.tmax),l.rmax+r.lmax);
}

void push_up(int u) {
    push_up(tr[u],tr[u<<1],tr[u<<1|1]);
}

void build(int u,int l,int r) {
    tr[u].l = l, tr[u].r = r;
    if(l==r) {
        tr[u]={l,r,a[r],a[r],a[r],a[r]};
        return ;
    }
    int mid = (l+r)>>1;

    build(u<<1, l, mid);
    build(u<<1|1, mid+1, r);
    push_up(u);

}
void  update(int u,int x,int v) {
    if(tr[u].l==tr[u].r) {
        tr[u]={x,x,v,v,v,v};
        return ;
    }
    int mid = (tr[u].l + tr[u].r)>>1;
    if(x<=mid) update(u<<1, x, v);
    else update(u<<1|1 , x, v);
    push_up(u);
}
Node query(int u,int ql,int qr) {
    if(ql<=tr[u].l&&qr>=tr[u].r) return tr[u];
    int mid=tr[u].l+tr[u].r>>1;
    if(qr<=mid) return query(u<<1,ql,qr);
    else if(ql>mid) return query(u<<1|1,ql,qr);
    else
    {
        auto left=query(u<<1,ql,qr);
        auto right=query(u<<1|1,ql,qr);
        Node res;
        push_up(res,left,right);
        return res;
    }
}
```

3.21 线段树

```
namespace Seg {
#define ls (u<<1)
#define rs (u<<1|1)
#define mid ((l+r)>>1)
    ll sum[N<<2], tag[N<<2];
    void push_up(int u) {
        sum[u] = sum[u<<1] + sum[u<<1|1];
    }
    void push_down(int u, int l, int r) {
        if (tag[u]) {
            tag[u<<1] += tag[u];
            tag[u<<1|1] += tag[u];
            sum[u<<1] += 1ll*(mid-l+1)*tag[u];
            sum[u<<1|1] += 1ll*(r-mid)*tag[u];
            tag[u] = 0;
        }
    }
    void modify(int u, int ql, int qr, int l, int r, int val) {
        if (ql <= l && qr >= r) {
            sum[u] += 1ll*(r-l+1)*val;
            tag[u] += val;
            return;
        }
        push_down(u, l, r);
        if (ql <= mid) modify(ls, ql, qr, l, mid, val);
        if (qr > mid) modify(rs, ql, qr, mid+1, r, val);
        push_up(u);
    }

#undef mid
#undef ls
#undef rs
}

一些技巧

1)找到最左端满足条件的位置:

1. 二分+线段树 复杂度是nlogn^2

2. 直接在线段树上找 复杂度是nlogn

int query(int u, int L, int R, int val) {
    if (t[u].mi > val) return 2e9;
    if (t[u].l == t[u].r)return t[u].l;
    int mid = t[u].l + t[u].r >> 1;
    if (L <= t[u].l && R >= t[u].r) {
        if (t[u<<1].mi <= val) return query(u<<1, t[u].l, mid, val);
        else return query(u<<1|1, mid + 1, t[u].r, val);
    }
    else {
        if (R <= mid)return query(u<<1, L, R, val);
        else if (L > mid)return query(u<<1|1, L, R, val);
        else return min(query(u<<1, L, mid, val), query(u<<1|1, mid + 1, R, val));
    }
}
```

```
2)势能线段树(均摊时间复杂度)

1. 区间与，单点修改，区间max

struct node {
    int l, r;
    int Or, And, Max, tag;
}t[N<<2];
int a[N];
void push_up(int u) {
    t[u].Or = t[u<<1].Or | t[u<<1|1].Or;
    t[u].And = t[u<<1].And & t[u<<1|1].And;
    t[u].Max = max(t[u<<1].Max, t[u<<1|1].Max);
}
void push_down(int u) {
    if (t[u].tag != 0) {
        t[u<<1].tag += t[u].tag;
        t[u<<1|1].tag += t[u].tag;
        t[u<<1].Max += t[u].tag;
        t[u<<1|1].Max += t[u].tag;
        t[u<<1].And += t[u].tag;
        t[u<<1|1].And += t[u].tag;
        t[u<<1].Or += t[u].tag;
        t[u<<1|1].Or += t[u].tag;
        t[u].tag = 0;
    }
}
void build(int u, int l, int r) {
    t[u].l = l, t[u].r = r;
    if (l == r) {
        t[u].Or = t[u].And = t[u].Max = a[l];
        return;
    }
    int mid = (l + r) >> 1;
    build(u<<1, l, mid);
    build(u<<1|1, mid+1, r);
    push_up(u);
}
void modify(int u, int ql, int qr, int val) {
    if ((t[u].Or & val) == t[u].Or) return;
    if (ql <= t[u].l && qr >= t[u].r && (t[u].Or & val) - t[u].Or == (t[u].And & val) - t[u].And) {
        int tmp = (t[u].Or & val) - t[u].Or;
        t[u].And += tmp;
        t[u].Or += tmp;
        t[u].Max += tmp;
        t[u].tag += tmp;
        return;
    }
    push_down(u);
    int mid = (t[u].l + t[u].r) >> 1;
    if (ql <= mid) modify(u<<1, ql, qr, val);
    if (qr > mid) modify(u<<1|1, ql, qr, val);
    push_up(u);
}
void update(int u, int pos, int val) {
    if (t[u].l == t[u].r) {
        t[u].Max = t[u].Or = t[u].And = val;
        return;
    }
    push_down(u);
    int mid = (t[u].l + t[u].r) >> 1;
    if (pos <= mid) update(u<<1, pos, val);
    else update(u<<1|1, pos, val);
    push_up(u);
}
int query(int u, int ql, int qr) {
    if (ql <= t[u].l && qr >= t[u].r) return t[u].Max;
    push_down(u);
    int mid = (t[u].l + t[u].r) >> 1;
    int ans = 0;
    if (ql <= mid) ans = max(ans, query(u<<1, ql, qr));
    if (qr > mid) ans = max(ans, query(u<<1|1, ql, qr));
    return ans;
}

2. 区间开根号，区间加，区间和

势能=max-min

3. 区间质因子问题
```

3.22 线段树 +dfs 序

```
int in[N], out[N];
vector<int> g[N];
void dfs(int u) {
    in[u] = ++tot;
    for (auto i : g[u]) {
        dfs(i);
    }
    out[u] = tot;
}

修改子树内信息[in[u], out[u]]
```

3.23 线段树标记永久化

```
typedef long long ll;
const int N = 1e5 + 5, M = 5e5 + 5, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;

int n, q, a[N];

struct Segment_tree {
#define ls u << 1
#define rs u << 1 | 1
    int L[N << 2], R[N << 2];
    ll sum[N << 2], lazy[N << 2];

    void push_up(int u) {
        sum[u] = sum[ls] + sum[rs] + (R[u] - L[u] + 1) * lazy[u];
    }

    void build(int u, int l, int r) {
        L[u] = l, R[u] = r;
        if (l == r) {
            sum[u] = a[l];
            return;
        }
        int m = (l + r) >> 1;
        build(ls, l, m);
        build(rs, m + 1, r);
        push_up(u);
    }

    void update(int u, int ql, int qr, ll k) {
        if (ql <= L[u] && R[u] <= qr) {
            sum[u] += (R[u] - L[u] + 1) * k;
            lazy[u] += k;
            return;
        }
        int m = (L[u] + R[u]) >> 1;
        if (ql <= m) update(ls, ql, qr, k);
        if (qr > m) update(rs, ql, qr, k);
        push_up(u);
    }

    ll query(int u, int ql, int qr, ll tg) {
        if (ql <= L[u] && R[u] <= qr) return sum[u] + (R[u] - L[u] + 1) * tg;
        int m = (L[u] + R[u]) >> 1;
        ll ans = 0;
        if (ql <= m) ans += query(ls, ql, qr, tg + lazy[u]);
        if (qr > m) ans += query(rs, ql, qr, tg + lazy[u]);
        return ans;
    }

#undef ls
#undef rs
} tr;
```

3.24 线段树合并 + 动态开点

```
#pragma GCC optimize (2)
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
const int N = 1e5 + 5;
const int M = 6e6 + 5;
struct edge {
    int to, next;
}e[2000010];

struct tree {
    int l, r;
    int maxx, id;
}t[M];

struct query {
    int x, y, z;
}q[N];

int tmp[N], n, m, h[N], cnt, son[N], root[N], siz[N], top[N], fa[N], tot, d[N], idx,
    all, ans[N], k;
```

```

void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

void dfs1(int u) {
    son[u] = -1;
    siz[u] = 1;
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (!d[v]) {
            d[v] = d[u] + 1;
            fa[v] = u;
            dfs1(v);
            siz[u] += siz[v];
            if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
        }
    }
}

void dfs2(int u, int t) {
    top[u] = t;
    if (son[u] == -1) return;
    dfs2(son[u], t);
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v != son[u] && v != fa[u]) dfs2(v, v);
    }
}

int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (d[top[u]] > d[top[v]])
            u = fa[top[u]];
        else
            v = fa[top[v]];
    }
    return d[u] > d[v] ? v : u;
}

void push_up(int u) {
    int ls = t[u].l, rs = t[u].r;
    if (t[ls].maxx >= t[rs].maxx) t[u].maxx = t[ls].maxx, t[u].id = t[ls].id;
    else t[u].maxx = t[rs].maxx, t[u].id = t[rs].id;
}

void update(int &now, int l, int r, int pos, int val) {
    if (!now) now = ++idx;
    if (l == r && l == pos){
        t[now].maxx += val;
        t[now].id = l;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid) update(t[now].l, l, mid, pos, val);
    else update(t[now].r, mid + 1, r, pos, val);
    push_up(now);
```

```
}

void merge(int &x, int &y, int l, int r) {
    if (!x) return;
    if (!y) { y = x;return; }
    if (l == r) {
        t[y].maxx += t[x].maxx;
        return;
    }
    int mid = (l + r) >> 1;
    merge(t[x].l, t[y].l, l, mid);
    merge(t[x].r, t[y].r, mid + 1, r);
    push_up(y);
}

void dfs_merge(int u, int fa) {
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v == fa) continue;
        dfs_merge(v, u);
        merge(root[v], root[u], 1, k);
    }
    if (t[root[u]].maxx == 0) ans[u] = 0;
    else {
        ans[u] = tmp[t[root[u]].id];
    }
}
void solve() {
    memset(h, -1, sizeof h);
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n - 1; i++) {
        int x, y;
        scanf("%d %d", &x, &y);
        add(x, y), add(y, x);
    }
    for (int i = 1; i <= m; i++) {
        scanf("%d %d %d", &q[i].x, &q[i].y, &q[i].z);
        tmp[++k] = q[i].z;
    }
    d[1] = 1;
    dfs1(1);
    dfs2(1, 1);
    sort(tmp + 1, tmp + 1 + k);
    k = unique(tmp + 1, tmp + 1 + k) - (tmp + 1);
    for (int i = 1; i <= m; i++) {
        q[i].z = lower_bound(tmp + 1, tmp + 1 + k, q[i].z) - tmp;
        update(root[q[i].x], 1, k, q[i].z, 1);
        update(root[q[i].y], 1, k, q[i].z, 1);
        update(root[lca(q[i].x, q[i].y)], 1, k, q[i].z, -1);
        if (fa[lca(q[i].x, q[i].y)]) update(root[fa[lca(q[i].x, q[i].y)]], 1, k, q[i].z, -1);
    }
    dfs_merge(1, 0);
    for (int i = 1; i <= n; i++) printf("%d\n", ans[i]);
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    signed test_index_for_debug = 1;
    char acm_local_for_debug = 0;
    do {
        if (acm_local_for_debug == '$') exit(0);
        if (test_index_for_debug > 20)
            throw runtime_error("Check the stdin!!!");
        auto start_clock_for_debug = clock();
        solve();
        auto end_clock_for_debug = clock();
        cout << "Test " << test_index_for_debug << " successful" << endl;
        cerr << "Test " << test_index_for_debug++ << " Run Time: "
            << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC << "s" << endl;
        cout << "--------------------------------------------------" << endl;
    } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
#else
    solve();
#endif
    return 0;
}
```

3.25 Hash 表

```
//手写哈希表
struct HashSet {
    const int mod = 1000009;
    struct node {
        int k, v, nex;
    } buf[N];
    int h[N], tot;
    void ins(int x) {
        int pos = x % mod;
        for (int i=h[pos]; i; i=buf[i].nex) {
            if (buf[i].k == x) { buf[i].v++; return ; }
        }
        buf[++tot] = (node){x, 1, h[pos]};
        h[pos] = tot;
    }
    int find(int x) {
        int pos = x % mod;
        for (int i=h[pos]; i; i=buf[i].nex) {
            if (buf[i].k == x) return buf[i].v;
        }
        return 0;
    }
} H;

//支持单点修改的hash
const int N = 2e6 + 10, M = 1e6 + 10;
const int mod = 1e9 + 7;
char s[N];
int n, m;
ll f[N];
struct BIT {
    ll c[N];
    int lowbit(int x) {return x&-x;}
    void add(int x, ll val) {
        for (int i = x; i <= n; i += lowbit(i)) c[i] = (c[i] + val) % mod;
    }
    ll qry(int x) {
        ll res = 0;
        for (int i = x; i > 0; i -= lowbit(i)) res = (res + c[i]) % mod;
        return res;
    }
    ll query(int l, int r) {
        return (qry(r) - qry(l-1) + mod) % mod;
    }
    bool comp(int l1, int r1, int l2, int r2) {
        return query(l1, r1)*f[l2-l1] % mod == query(l2, r2);
    }
    bool check(int l1, int r1, int l2, int r2) {
        if (l1 == r1) return true;
        int mid1 = (l1 + r1) >> 1;
        int mid2 = (l2 + r2) >> 1;
        bool f1 = comp(l1, mid1, l2, mid2);
        bool f2 = comp(mid1+1, r1, mid2+1, r2);
        if (!f1 && !f2) return false;
        if (!f1) return check(l1, mid1, l2, mid2);
        else return check(mid1+1, r1, mid2+1, r2);
    }
}bit;
void init() {
    f[0] = 1;
    for (int i = 1; i <= n; i++) f[i] = f[i-1] * 2333 % mod;
    for (int i = 1; i <= n; i++) bit.add(i, s[i] * f[i] % mod);
}

修改操作：将x位置的改为c
bit.add(x, (c-s[x]+mod)%mod*f[x]%mod);
s[x] = c;
```

4 数学

4.1 EX$_CRT$

```cpp
/***)□ □ й□ ȼ□ □ □ □ ***/
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int MAXN = 2e5 + 5;
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if(b == 0) {
        x = 1; y = 0;
        return a;
    }
    ll r = exgcd(b, a % b, x, y);
    ll t = x; x = y; y = t - a / b * y;
    return r;
}

int n;
ll a[MAXN], b[MAXN];

ll solve() {
    ll A = a[1], B = b[1], t, d, x, y;
    for(int i = 2; i <= n; i++) {
        d = exgcd(A, a[i], x, y);
        if((b[i] - B) % d) return -1;
        x = x * (b[i] - B) / d, t = a[i] / d, x = (x % t + t) % t;
        B = A * x + B;
        A = A / d * a[i];
        B %= A;
    }
    while(B < 0) B += A;
    return (B + A) % A;
}
int main() {
    scanf("%d",&n);
    for(int i = 1; i <= n; i++)
        scanf("%lld%lld", &a[i], &b[i]);
    printf("%lld\n",solve());

    return 0;
}
```

4.2 EX$_Lucas$

```cpp

// p不为质数，利用中国剩余定理结合求解

#include <iostream>
using namespace std;

typedef long long ll;

const int N = 1e5 + 10;

ll quick_pow(ll a, ll b, ll P) {
    ll ans = 1;
    while(b) {
        if(b & 1)
            ans = ans * a % P;
        a = a * a % P;
        b >>= 1;
    }
    return ans % P;
}

ll ex_gcd(ll a, ll b, ll &x, ll &y) {
    ll res, t;
    if(!b) {
        x = 1;
        y = 0;
        return a;
    }
    res = ex_gcd(b, a % b, x, y);
    t = x;
    x = y;
    y = t - (a / b) * y;
    return res;
}

ll INV(ll a, ll mod) {
    ll x, y;
    ll d = ex_gcd(a, mod, x, y);
    return d ? (x % mod + mod) % mod : -1;
}

ll fac(ll n, ll P, ll pk) {// 阶乘除去质因子后模质数幂 (n / p^a) % pk
    if(!n) return 1;
    ll ans = 1;
    for(int i = 1;i < pk; i++) {// 第三部分：n!与p互质的乘积
        if(i % P)
            ans = ans * i % pk;
    }
    ans = quick_pow(ans, n / pk, pk) % pk; // 第三部分：n!与p互质的乘积,ans循环的次数为n/pk
    for(int i = 1;i <= n % pk; i++) {// 第四部分：循环过后n!剩下的部分
        if(i % P) ans = ans * i % pk;
    }
    return ans * fac(n / P, P, pk) % pk;  // 第一部分，p的幂，个数为n/p;   第二部分：(n/p)!
}

ll C(ll m, ll n, ll P, ll pk) {// 组合数模质数幂
    if(n < 0 || m < 0 || n > m) return 0;
    ll f1 = fac(m, P, pk), f2 = fac(n, P, pk), f3 = fac(m - n, P, pk), tmp = 0; // tmp = pk1 - pk2 - pk3
    for(ll i = m; i ; i /= P)     tmp += i / P;
    for(ll i = n; i ; i /= P)     tmp -= i / P;
    for(ll i = m - n; i ; i /= P) tmp -= i / P;
    return f1 * INV(f2, pk) % pk * INV(f3, pk) * quick_pow(P, tmp, pk) % pk;
}

ll p[N], a[N];
int cnt;

ll CRT() {
    ll M = 1, ans = 0;
    for(int i = 1;i <= cnt; i++)  M *= p[i];
    for(int i = 1;i <= cnt; i++) {
        ll m = M / p[i];
        ans = (ans + a[i] * m % M * INV(m, p[i]) % M) % M;
    }
    return (ans % M + M) % M;
}

ll EX_Lucas(ll m, ll n, ll P) {
    for(int i = 2;i * i <= P; i++) {
        if(P % i == 0) {
            ll tmp = 1;
            while(P % i == 0) {
                tmp *= i;
                P /= i;
            }
            p[++cnt] = tmp;
            a[cnt] = C(m, n, i, tmp);
        }
    }
    if(P > 1) {
        p[++cnt] = P;
        a[cnt] = C(m, n, P, P);
    }
    return CRT();
}
int main()
{
    ll m, n, P;
    cin >> m >> n >> P;
    cnt = 0;
    cout << EX_Lucas(m, n, P) << endl;
}
```

4.3 Lucas

```
// p一定为质数

#include <iostream>
using namespace std;

typedef long long ll;

const int N = 1e7 + 10;

ll p; // C(n,m) % p

ll f[N], inv[N], invF[N];

void Init()
{
    f[0] = f[1] = inv[0] = inv[1] = invF[0] = invF[1] = 1;
    for(int i = 2;i <= p; i++)
    {
        f[i] = f[i - 1] * i % p;
        inv[i] = (p - (p / i)) * inv[p % i] % p;
        invF[i] = invF[i - 1] * inv[i] % p;
    }
}
```

```

ll C(ll m, ll n)
{
    if(m < 0 || n < 0 || n > m)
        return 0;
    ll ans = f[m];
    ans = ans * invF[n] % p;
    ans = ans * invF[m - n] % p;
    return ans;
}

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

int main()
{
    ll m, n;
    cin >> m >> n >> p;
    Init();
    cout << Lucas(m, n) << endl;
}
```

4.4 Miller$_Rabin$

```
// 二次探测定理：对素数p,满足x^2≡1(modp)的小于p的正整数解x只有1或p-1.

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 7;
const int times = 10;

ll ksc(ll a, ll b, ll mod)  {
    ll ans = 0;
    while(b > 0) {
        if(b & 1) {
            ans = (ans + a) % mod;
        }
        a = (a << 1) % mod;
        b >>= 1;
    }
    return ans;
}

ll quick_pow(ll a, ll b, ll mod) {
    ll ans = 1, base = a;
    while(b != 0) {
        if(b & 1) {
            ans = ans * base % mod;
        }
        base = base * base % mod;
        b >>= 1;
    }
    return ans;
}

bool Miller_Pabin(ll n)//Miller测试的主体结构
{
    if(n < 2) return false;
    if(n == 2) return true;
    if(n & 1 == 0) return false;//对于偶数的优化
    ll k = 0,u = n - 1;//p为Miller测试的k，u为Miller测试的m

    while(u & 1 == 0){ // 把x拆成u*2^k
        u >>= 1;
        k++;
    }
    srand(time(NULL));

    ll x, pre; // pre为上次探测的x的值

    for(int i = 1;i <= times; i++) {
        x = rand() % (n - 1) + 1;
        x = quick_pow(x, u, n); // 先求出x^u(mod n)
        pre = x;
        for(int j = 1;j <= k; j++) {
            x = ksc(x, x, n);
            if(x == 1 && pre != 1 && pre != n - 1)
                return false;
            pre = x;
        }
        if(x != -1)
        return false;
    }
    return true;
}

int main() {
    ll n; cin >> n;
    cout << (Miller_Pabin(n) ? "Prime" : "Not a Prime") << endl;
}
```

4.5 Min25 筛

```
#include <iostream>

using namespace std;

typedef long long ll;

const int N = 1e5 + 10;


namespace Min25 {
    int prime[N], id1[N], id2[N], flag[N], ncnt, m;

    ll g[N], sum[N], a[N], T, n;

    inline int ID(ll x) {
        return x <= T ? id1[x] : id2[n / x];
    }

    inline ll calc(ll x) {
        return x * (x + 1) / 2 - 1;
    }

    inline ll f(ll x) {
        return x;
    }

    inline void init() {
        ncnt = 0, m = 0;
        T = sqrt(n + 0.5);
        for (int i = 2; i <= T; i++) {
            if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
            for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {
                flag[i * prime[j]] = 1;
                if (i % prime[j] == 0) break;
            }
        }
        for (ll l = 1; l <= n; l = n / (n / l) + 1) {
            a[++m] = n / l;
            if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
            g[m] = calc(a[m]);
        }
        for (int i = 1; i <= ncnt; i++)
            for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)
                g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
    }

    inline ll Solve(ll x) {
        if (x <= 1) return x;
        return n = x, init(), g[ID(n)];
    }

}
```

4.6 杜教筛

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
const int N = 1e6 + 10;

unordered_map<int, ll> smu, sphi;
bool isPrime[N];
int prime[N], num;
ll mu[N], phi[N];

void makeMobiusAndEuler(int siz) {
    mu[1] = phi[1] = 1;
    for (int i = 2; i <= siz; i++) {
        if (!isPrime[i]) prime[++num] = i, mu[i] = -1, phi[i] = i - 1;
        for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
            isPrime[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else {
                phi[i * prime[j]] = phi[prime[j]] * phi[i];
                mu[i * prime[j]] = -mu[i];
            }
        }
    }
    for (int i = 1; i <= siz; i++) mu[i] += mu[i - 1], phi[i] += phi[i - 1];
}

ll getSmu(int n) {
    if (n < N) return mu[n];
    if (smu[n]) return smu[n];
    ll res = 1;
    for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
        r = n / (n / l);
        res -= 1ll * (r - l + 1) * getSmu(n / l);
    }
    return smu[n] = res;
}

ll getSphi(int n) {
    if (n < N) return phi[n];
    if (sphi[n]) return sphi[n];
    ll res = 1ll * n * (n + 1) / 2;
    for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
        r = n / (n / l);
        res -= 1ll * (r - l + 1) * getSphi(n / l);
    }
    return sphi[n] = res;
}
```

4.7 杜教筛

```
//筛大范围前缀和
ll prime[N], mu[N], k;
ll phi[N];
bool is_prime[N];
inline void init(int n) {
    memset(is_prime, true, sizeof is_prime);
    mu[1] = 1; phi[1] = 1;
    for (re int i = 2; i < n; ++i) {
        if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
        for (re int j = 1; j <= k && i * prime[j] < n; ++j) {
            is_prime[i * prime[j]] = false;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            } else {
                mu[i * prime[j]] = -mu[i];
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            }
        }
    }
    for (re int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
}
unordered_map<ll, ll> sum_mu, sum_phi;
```

```
inline ll GetSum_mu(ll n) {
    if (n <= 3e7) return mu[n];
    if (sum_mu[n]) return sum_mu[n];
    ll ans = 1;
    for (re ll l = 2, r; l <= n; l = r + 1) {
        r = min(n, n / (n / l));
        ans -= (r - l + 1) * GetSum_mu(n / l);
    }
    return sum_mu[n] = ans;
}

inline ll GetSum_phi(ll n) {
    if (n <= 3e7) return phi[n];
    if (sum_phi[n]) return sum_phi[n];
    ll ans = n * (n + 1) / 2;
    for (re ll l = 2, r; l <= n; l = r + 1) {
        r = min(n, n / (n / l));
        ans -= (r - l + 1) * GetSum_phi(n / l);
    }
    return sum_phi[n] = ans;
}
```

4.8 二次剩余

```
#include <iostream>
#include <ctime>

using namespace std;

typedef long long ll;


typedef struct{
    ll x, y; // 把求出来的w作为虚部，则为a + bw
}num;

ll quick_pow(ll a, ll b, ll p) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return ans % p;
}


num num_mul(num a, num b, ll w, ll p) {// 复数乘法
    num ans = {0, 0};
    ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
    ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
    return ans;
}

ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
    num ans = {1, 0};
    while(b) {
        if(b & 1)
```

```
            ans = num_mul(ans, a, w, p);
        a = num_mul(a, a, w, p);
        b >>= 1;
    }
    return ans.x % p;
}

ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
    return quick_pow(a, (p - 1) >> 1, p);
}

ll Cipolla(ll n, ll p) {// 输入a和p，是否存在x使得x^2 = a (mod p)，存在二次剩余返回x，存在二次非剩余返回-1      注意：p是奇质数
    n %= p;
    if(n == 0)
        return 0;
    if(p == 2)
        return 1;
    if(legendre(n, p) + 1 == p) // 二次非剩余
        return -1;

    ll a, w;

    while(true) {// 找出a，求出w，随机成功的概率是50%，所以数学期望是2
        a = rand() % p;
        w = ((a * a - n) % p + p) % p;
        if(legendre(w, p) + 1 == p) // 找到w，非二次剩余条件
            break;
    }
    num x = {a, 1};
    return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x,一个解是x，另一个解是p-x，这里的w其实要开方，但是由拉格朗日定理可知虚部为0，所以最终答案就是对x的实部用快速幂求解
}

int main()
{
    ll n, p;
    cin >> n >> p;
    srand((unsigned)time(NULL));
    cout << Cipolla(n, p) << endl;
    return 0;
}
```

4.9 反演相关

```
/*
莫比乌斯反演
g[n] = \sum_{d | n} f[d]
f[d] = \sum_{d | n} g[d] * mu[n / d]
二项式反演
g[n] = \sum{i = 1}^{n} C(n, i) * f[i]
f[n] = \sum{i = 1}^{n} C(n, i) * g[i] * (-1)^{n - i}
子集反演
f(S) = \sum_{T \belong S} g(T)
g(S) = \sum_{T \belong S} f(T) * (-1) ^ {|S| - |T|}
*/
```

4.10 斐波那契

```
gcd(f[n], f[m]) = f[gcd(n, m)]

斐波那契前n项和S[n] = f[n+2] - 1;
```

4.11 光速幂

```

ll v_pow(ll a, ll b) {
    ll ans = 1;
    ll base = 65536, k = 1;
    while(1) {
        if((b % (k * base)) / k == 0) break;
        ans = ans * quick_pow(a, (b % (k * base)) / k) % mod;
        a = quick_pow(a, base) % mod;
        k = k * base;
    }
    return ans;
}
```

4.12 康托展开

```
#include <bits/stdc++.h>//康拓展开
using namespace std;
typedef long long ll;
const int N = 1e6 + 5;
const int Mod = 998244353;
int T, n;
ll a[N], fac[N], c[N];
inline int lowbit(int x) { return x & -x; }
void add(int x, int v) {
    for (int i = x; i <= n; i += lowbit(i)) c[i] += v;
}
int query(int x) {
    int res = 0;
    for (int i = x; i > 0; i -= lowbit(i)) res += c[i];
    return res;
}
void get_fac() {
    fac[1] = 1;
    for (int i = 2; i <= N; i++) fac[i] = fac[i - 1] * i % Mod;
}
ll cantor() {
    for (int i = n; i >= 1; i--) cin >> a[i];
    ll res = 0;
    for (int i = 1; i <= n; i++) {
        res += 1ll*query(a[i]) * fac[i-1];
        res %= Mod;
        add(a[i], 1);
    }
    return (res + 1) % Mod;
}

int main() {
    get_fac();
    cin >> n;
    cout << cantor() << endl;
}
/*

ans = sum(a[i] * (i-1)!.....) + 1
a[i] 代表第i个数比i到n的数之中大的个数

*/

//逆康托展开
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6 + 5;
const int Mod = 998244353;
int T, n, order;
ll fac[N];
vector<int> a, ans;
void get_fac() {
    fac[1] = fac[0] = 1;
    for (int i = 2; i <= N; i++) fac[i] = fac[i - 1] * i % Mod;
}
void decantor(int order, int n) {
    for (int i = 1; i <= n; i++) a.push_back(i);

    for (int i = n; i >= 1; i--) {
        int r = order % fac[i - 1];
        int t = order / fac[i - 1];
        order = r;
        ans.push_back(a[t]);
        a.erase(a.begin() + t);
    }
}
int main() {
    get_fac();
    cin >> order >> n;
    order--;
    for (auto x : ans) cout << x << " ";
}
```

4.13 快速幂

```
ll mul(ll a, ll b) {
    ll z = (long double) a / mod * b;
    ll res = (unsigned long long) a * b - (unsigned long long) z * mod;
    return (res + mod) % mod;
}
// O(1) quick_mul, use long double
inline ll quick_pow(ll ans, ll p, ll res = 1) {
    for(; p; p >>= 1, ans = mul(ans, ans) % mod)
        if(p & 1) res = mul(res, ans) % mod;
    return res % mod;
}
double gcd(double a,double b) {
    if(fabs(b) < eps) return a;
    if(fabs(a) < eps) return b;
    return gcd(b, fmod(a,b));
}
```

```
int gcd(int a, int b) { return __gcd(a, b); }
ll gcd(ll a, ll b) { return __gcd(a, b); }
```

4.14 莫比乌斯反演

```
ll prime[N], mu[N], k;
ll phi[N];
bool is_prime[N];
inline void init(int n) {
    memset(is_prime, true, sizeof is_prime);
    mu[1] = 1; phi[1] = 1;
    for (int i = 2; i < n; ++i) {
        if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
        for (int j = 1; j <= k && i * prime[j] < n; ++j) {
            is_prime[i * prime[j]] = false;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            } else {
                mu[i * prime[j]] = -mu[i];
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
            }
        }
    }
    for (int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
}

ll cal(int n, int m) {
    ll ans = 0;
    n /= d, m /= d;
    int mx = min(n, m);
    for (ll l = 1, r; l <= mx; l = r + 1) {
        r = min(n / (n / l), m / (m / l));
        ans += 1ll * (mu[r] - mu[l-1]) * (n / l) * (m / l);
    }
    return ans;
}
```

4.15 逆元

```
/***□ □ □ ***/
namespace INV {
    typedef long long ll;
    const int N = 2e5 + 10;
    const int mod = 1e9 + 7;
    int inv[N];
    ll x, y;
    ll gcd(ll a, ll b) {
        return b ? gcd(b, a % b) : a;
    }
    void get_inv(int n) {
        inv[0] = inv[1] = 1;
        for(int i = 2; i <= n; ++i)
            inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
    }
    ll quick_pow(ll ans, ll p, ll res = 1) {
        ans %= mod;
        p %= mod - 1;
        for(; p; p >>= 1, ans = ans * ans % mod)
            if(p & 1) res = res * ans % mod;
        return res % mod;
    }
    ll inv1(ll ans) {
        return quick_pow(ans, mod - 2);
    }
    ll ex_gcd(ll a, ll b, ll &x, ll &y) { //)□ ý¾¸□ □ □ □ 熢
        if(!b) {
            x = 1, y = 0;
            return a;
        }
        ll r = ex_gcd(b, a % b, y, x);
        ll t = x;
        x = y;
        y = t - a / b * y;
        return r;
    }
    ll inv2(ll a) { //□ □ a□ □ mod□ µ□ □ □ □ £¬□ □ □ ²»´□ □ «µ□ □ -1
        ll d = ex_gcd(a, mod, x, y);
        return d == 1 ? (x % mod + mod) % mod : -1;
    }
}
```

4.16 欧拉函数

```
int prime[N], phi[N], k;
bool is_prime[N];
void get_phi(int n) {
    memset(is_prime, true, sizeof is_prime);
    phi[1] = 1;
    for (int i = 2; i < n; i++) {
        if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
        for (int j = 1; j <= k && i * prime[j] < n; j++) {
            is_prime[i * prime[j]] = false;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}

ll init(ll n) {
    ll m = (int)sqrt(n + 0.5);
    ll ans = n;
    for (ll i = 2; i <= m; ++ i) {
        if (n % i == 0) {
            ans = ans / i * (i - 1);
            while(n % i == 0) n /= i;
        }
    }
    if (n > 1) ans = ans / n * (n - 1);
    return ans;
}
```

4.17 欧拉降幂

```
void init(int n) {
    memset(is_prime, true, sizeof is_prime);
    phi[1] = 1;
    for (int i = 2; i < n; i++) {
        if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
        for (int j = 1; j <= k && i * prime[j] < n; j++) {
            is_prime[i * prime[j]] = false;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}
ll mod(ll a, ll mm) {return a >= mm ? a % mm + mm : a;}
ll ksm(ll a, ll b, ll mm) {
    ll res = 1, base = a;
    while (b) {
        if (b & 1) res = mod(res * base, mm);
        base = mod(base * base, mm);
        b >>= 1;
    }
    return res;
}
ll calc(ll a, ll b, ll mm) {
    if (b == 0 || mm == 1) return 1;
    else return ksm(a, calc(a, b-1, phi[mm]), mm);
}
ll f(ll a, ll b, ll mm) {
    if (a == 0) return 0;
    ll ans = calc(a, b, mm) % mm;
    return ans;
}
```

4.18 筛法树

```
ll fa_prime[N], fa_prime_edge[N];
ll prime[N], cnt;
bool is_prime[N];
void prime_table(int n) {
    memset(is_prime, true, sizeof is_prime);
    cnt = 0;
    for (int i = 2; i < n; i++) {
        if (is_prime[i]) prime[++cnt] = i, fa_prime[i] = 1, fa_prime_edge[i] = i;
        for (int j = 1; j <= cnt && i * prime[j] < n; j++) {
            is_prime[i * prime[j]] = false;
            fa_prime[i * prime[j]] = i;
            fa_prime_edge[i * prime[j]] = prime[j];
            if (i % prime[j] == 0) break;
        }
    }
}
```

4.19 算术基本定理

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

ll get_Count(ll n) {
    ll ans = 1;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            int a = 0;
            while(n % i == 0) {
                a++;
                n /= i;
            }
            ans *= (a + 1);
        }
    }
    if(n > 1)  ans *= 2;
    return ans;
}

ll get_Sum(ll n) {
    ll ans = 1;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            ll a = 1;
            while(n % i == 0) {
                n /= i;
                a *= i;
            }
            ans = ans * (a * i - 1) / (i - 1);
        }
    }
    if(n > 1) ans *= (n + 1);
    return ans;
}
```

4.20 质数、积性函数

```cpp
/***□ □ □ □ /»□ □ □ ¯□ □ ***/
//2147483647□ □ □ □ □ □ □ □ ¸□ □ □ £°105097565, 1e8□ □ □ □  5e6 ¸□ □ □ □ □
void get_all(int n) {
    phi[1] = 1;
    for (int i = 2; i <= n; ++ i) {
        if (!vis[i]) prime[++ cnt] = i;
        for (int j = 1; j <= cnt && prime[j] * i <= n; ++ j) {
            vis[i * prime[j]] = true;
//          p_num[i * prime[j]] = p_num[i] + p_num[prime[j]];
            if (i % prime[j] == 0) {
                mu[i * prime[j] = 0;
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            } else {
                mu[i * prime[j]] = -mu[i];
                phi[i * prime[j]] = phi[i] * phi[prime[j]];
            }
        }
    }
}

ll Euler(ll n) {
    ll res = n;
    for(int i = 2; i * i <= n; ++i) {
        if(n % i == 0) res = res / i * (i - 1);
        while(n % i == 0) n /= i;
    }
    if(n > 1) res = res / n * (n - 1);
    return res;
}
```

4.21 线性筛

```
--------------------------------------------------------//筛质数
const int N = 1e6 + 5;
int prime[N];
bool is_prime[N];
void get_prime(){
    int k = 0;
    memset(is_prime, true, sizeof is_prime);
    is_prime[0] = is_prime[1] = false;
    for(int i = 2 ; i <= N;i++){
        if (is_prime[i]) prime[++k] = i;
        for(int j = 1; j <= k && i * prime[j] <= N;j++){
            is_prime[i * prime[j]] = false;
            if(i % prime[j] == 0) break;
        }
    }
}
--------------------------------------------------------//约数和
int prime[N], cnt;
bool is_prime[N];
ll sum[N], e[N];

void init() {
    memset(is_prime, true, sizeof is_prime);
    sum[1] = 1;
    for (int i = 2; i < N; ++i) {
        if (is_prime[i]) {
            prime[++cnt] = i;
            sum[i] = i + 1;
            e[i] = 1;
        }
        for (int j = 1; j <= cnt && 1ll*i * prime[j] < N; ++j) {
            is_prime[prime[j] * i] = false;
            if (i % prime[j] == 0) {
                sum[i * prime[j]] = sum[i] * prime[j] + e[i];
                e[i * prime[j]] = e[i];
                break;
            }
            sum[i * prime[j]] = sum[i] * (prime[j] + 1);
            e[i * prime[j]] = sum[i];
        }
    }
}
```

4.22 线性同余方程

```
/***□ □ □ □ ʳ□ □ ³□ ***/
void RemainderEquation(int a, int b, int n) {
    ll X, Y, d, res;
    ll min_res;
    d = gcd(a,n);
    exgcd(a, n, X, Y);
    if(b%d == 0) {
        X = X * (b / d) % n;//µõ½ʳ□ ³½□ □ ɦ½□
        for(int i = 0 ; i < d; i++) {
            res = (X + (i * (b/d))) % n;
            cout << res << '\n';    //□ □ □ □ □ □ □ □
        }
        min_res=(X%(n/d)+(n/d))%(n/d);
        cout<<min_res<<endl;        //□ □ □ □ □ C½□
    } else {
        cout << No Sulutions! << '\n';
    }
}
```

4.23 原根

```

#include <iostream>
#include <vector>

using namespace std;

typedef long long ll;

vector<ll> YG;
ll p, n; // p是模数，n是p的欧拉函数值

ll gcd(ll a, ll b) {
    return b ? gcd(b, a % b) : a;
}

ll quick_pow(ll a, ll b, ll p) ;

ll phi(ll n) {
    ll ans = n;
    for(int i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            ans = ans - ans / i;
            while(n % i == 0) {
                n /= i;
            }
        }
    }
    if(n > 1)
        ans = ans - ans / n;
    return ans;
}

vector<ll> PrimeFac(ll n) { // n的素因子
    vector<ll> fac;
    fac.clear();
    for(ll i = 2;i * i <= n; i++) {
        if(n % i == 0) {
            fac.push_back(i);
            while(n % i == 0)
                n /= i;
        }
    }
    if(n > 1)
        fac.push_back(n);
    return fac;
}

bool is_Protogen(ll p) { // 原根p = 2、4、p^k、2*p^k(p为非2的质数，k为任意数)
    if(p == 2 || p == 4) return true;
    if(p <= 1 || p % 4 == 0) return false;
    ll num = 0;
    while(p % 2 == 0) // 2的倍数先筛掉
        p /= 2;
    for(int i = 3;i * i <= p; i++) { // p只能是一个非2的素数的倍数构成，否则没有原根
        if(p % i == 0) {
            num++;
            while(p % i == 0)
                p /= i;
        }
    }
    if(p > 1) num++;
    if(num == 1) return true;
    return false;
}

ll Protogen(ll p) {
    if(!is_Protogen(p)) // 先判断是否存在原根
        return -1;
    n = phi(p);
    if(p == 2) return 1;
    if(p == 3) return 2;
    if(p == 4) return 3;
    vector<ll> fac = PrimeFac(n); // f(p)的素因子
    for(int i = 2;i <= p - 1; i++) {
        if(gcd(i, p) != 1) // n是模p的欧拉函数值，i要和n互质
            continue;
        bool flag = true;
        for(ll j = 0;j < fac.size(); j++) {
            if(quick_pow(i, n / fac[j] , p) == 1)
                flag = 0;
        }
        if(flag) // i就是原根
            return i;
    }
    return -1;
}

void Sum_Protogen(ll k) { // 找出n的所有原根
    YG.push_back(k);
    for(int i = 2;i < n; i++) {
        if(gcd(i, n) == 1) // i要与f(n)互质
            YG.push_back(quick_pow(k, i, p));
    }
}
```

```cpp

int main() {
    cin >> p;
    ll k = Protogen(p); // p的原根
    cout << k << endl;
    Sum_Protogen(k);
    for(int i = 0;i < YG.size(); i++) {
        cout << YG[i] << " ";
    }
    cout << endl;
    return 0;
}
```

4.24 原根表

```
mod                                         原根
r*2^k+1   r               k              g
3   1   1   2
5   1   2   2
17  1   4   3
97  3   5   5
193 3   6   5
257 1   8   3
7681    15  9   17
12289   3   12  11
40961   5   13  3
65537   1   16  3
786433  3   18  10
5767169 11  19  3
7340033 7   20  3
23068673    11  21  3
104857601   25  22  3
167772161   5   25  3
469762049   7   26  3
998244353   119 23  3   这个数常用
1004535809  479 21  3   加起来不会爆int
2013265921  15  27  31
2281701377  17  27  3    这个数平方刚好不会爆ll
3221225473  3   30  5
75161927681 35  31  3
77309411329 9   33  7
206158430209    3   36  22
2061584302081   15  37  7
2748779069441   5   39  3
6597069766657   3   41  5
39582418599937  9   42  5
79164837199873  9   43  5
263882790666241 15  44  7
1231453023109121    35  45  3
1337006139375617    19  46  3
3799912185593857    27  47  5
4222124650659841    15  48  19
7881299347898369    7   50  6
31525197391593473   7   52  3
180143985094819841  5   55  6
1945555039024054273 27  56  5
4179340454199820289 29  57  3
```

4.25 整除分块

```
int calc(int n, int m) {
    //\sum_{i = 1} ^ {m} n / i
    //向下取整
    for (int l = 1, r; l <= m; l = r + 1) {
        if (n / l) r = min(m, n / (n / l));
        else r = m;
        //[l, r]之间的 n / l 都相等
    }

    //向上取整
    for (int l = 1, r; l <= m; l = r + 1) {
        int t = (n + l - 1) / l;
        if (t == 1) r = m;
        else r = min(m, (n - 1) / (t - 1));
        //[l, r]之间的 (n + l - 1) / l 都相等
    }

}
```

4.26 整数拆分

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

//递归
ll PartitionCount(ll n, ll m)
{
    if(n == 1 || m == 1)
        return 1;
    else if(n < m)
        return PartitionCount(n , n);
    else if(n == m)
        return PartitionCount(n , n - 1) + 1;
    else
     return PartitionCount(n - m , m) + PartitionCount(n , m - 1);
}

//DP
ll dp[10005][10005];

void Partition_DP(ll n, ll m)
{
    for(ll i = 1;i <= n + 1; i++)
    {
        for(ll j = 1;j <= m + 1; j++)
        {
            if(i == 1 || j == 1)
                dp[i][j] = 1;
            else if(i == j)
                dp[i][j] = 1 + dp[i][j - 1];
            else if(i < j)
                dp[i][j] = dp[i][i];
            else
                dp[i][j] = dp[i - j][j] + dp[i][j - 1];
        }
    }
}
```

4.27 整数分块

```
void cal(ll a) {
    for (ll l = 1, r; l <= a; l = r + 1) {
        r = min(a, (a / (a / l)));
        ans = (ans + (r - l + 1) % MOD * (a / l));
    }
}
```

4.28 BSGS

```
/***BSGS***/
int A,B,C; // A^x = B (mod C)
struct Hashmap {//¹▯ ▯ ±▯ ▯ ▯ ▯ map
    static const int Ha = 999917, maxe = 46340;
    int E,lnk[Ha],son[maxe+5],nxt[maxe+5],w[maxe+5];
    int top,stk[maxe+5];
    void clear() {E=0;while (top) lnk[stk[top--]]=0;}
    void Add(int x,int y) {son[++E]=y;nxt[E]=lnk[x];w[E]=((1<<30)-1)*2+1;lnk[x]=E;}
    bool count(int y) {
        int x=y%Ha;
        for (int j=lnk[x];j;j=nxt[j])
            if (y==son[j]) return true;
        return false;
    }
    int& operator [] (int y) {
        int x=y%Ha;
        for (int j=lnk[x];j;j=nxt[j])
            if (y==son[j]) return w[j];
        Add(x,y);stk[++top]=x;return w[E];
    }
} f;
int exgcd(int a,int b,int &x,int &y) {
    if (!b) { x = 1; y = 0; return a; }
    int r = exgcd(b,a % b,x,y), t = x; x = y; y = t - a / b * y;
    return r;
}
int BSGS(int A,int B,int C) {
    if (C==1) if (!B) return A!=1; else return -1;
    if (B==1) if (A) return 0; else return -1;
    if (A%C==0) if (!B) return 1; else return -1; //¼¸▯ ▯ ▯ ▯ ▯ ▯
    int m=ceil(sqrt(C)),D=1,Base=1;f.clear();
    for (int i=0;i<=m-1;i++) {//▯ Ō▯ A^j´▯ ▯ ¹▯ ▯ ±▯
        f[Base]=min(f[Base],i);
        Base=((ll)Base*A)%C;
    }
    for (int i=0;i<=m-1;i++) {
        int x,y,r=exgcd(D,C,x,y);
        x=((ll)x*B%C+C)%C; //)ŷ▯ ▯ A^j
        if (f.count(x)) return i*m+f[x]; //▯ ▯ ½▯ ▯
        D=((ll)D*Base)%C;
    }
    return -1;
```

```
}
```

4.29 Simpson 积分

```

inline double f(double x) {
    return 0.0;
}
double simpson(double l, double r) {
    double mid = (l + r) / 2;
    return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
}
double asr(double l, double r, double ans, double eps) {
    double mid = (l + r) / 2;
    double fl = simpson(l, mid), fr = simpson(mid, r);
    if (abs(fl + fr - ans) <= 15 * eps)
        return fl + fr + (fl + fr - ans) / 15;
    return asr(l, mid, fl, eps * 0.5) + asr(mid, r, fr, eps * 0.5);
}
```

4.30 2 维计算几何

```
const double pi = acos(-1.0);
// `¼□ □ 瓻°□ ǵ°□ `
const double eps = 1e-8;
const int maxp = 1010;
//`Compares a double to zero`
int sgn(double x) {
    if(fabs(x) < eps)return 0;
    return (x > 0? 1: -1);
}
//square of a double
inline double sqr(double x) { return x * x; }
/*
 * Point
 * Point()                  - Empty constructor
 * Point(double _x,double _y)  - constructor
 * input()                - double input
 * output()               - %.2f output
 * operator ==            - compares x and y
 * operator <             - compares first by x, then by y
 * operator -             - return new Point after subtracting curresponging x and y
 * operator ^             - cross product of 2d points
 * operator *             - dot product
 * len()                  - gives length from origin
 * len2()                 - gives square of length from origin
 * distance(Point p)      - gives distance from p
 * operator + Point b     - returns new Point after adding curresponging x and y
 * operator * double k    - returns new Point after multiplieing x and y by k
 * operator / double k    - returns new Point after divideing x and y by k
 * rad(Point a,Point b)- returns the angle of Point a and Point b from this Point
 * trunc(double r)        - return Point that if truncated the distance from center to r
 * rotleft()              - returns 90 degree ccw rotated point
 * rotright()             - returns 90 degree cw rotated point
 * rotate(Point p,double angle) - returns Point after rotateing the Point centering at
   p by angle radian ccw
 */
```

```
struct Point {
    double x,y;
    Point() {}
    Point(double _x,double _y) { x = _x; y = _y; }
    void input() { cin >> x >> y; }
    bool operator == (Point b)const {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }
    bool operator < (Point b)const {
        return sgn(x - b.x) == 0? sgn(y - b.y) < 0: x < b.x;
    }
    Point operator -(const Point &b)const {
        return Point(x - b.x, y - b.y);
    }
    double operator ^(const Point &b)const {//²□ □
        return x * b.y - y * b.x;
    }
    double operator *(const Point &b)const {//µ□ □
        return x * b.x + y * b.y;
    }
    double len() {//·µ»□ ¤¶□
        return hypot(x, y);//¿⺯ □ □
    }
    double len2() { //·µ»□ ¤¶□ □ □ ·½
        return x * x + y * y;
    }
    double distance(Point p) {  //·µ»□ }µ□ l'□ □ □
        return hypot(x - p.x, y - p.y);
    }
    Point operator +(const Point &b)const {
        return Point(x + b.x, y + b.y);
    }
    Point operator *(const double &k)const {
        return Point(x * k, y * k);
    }
    Point operator /(const double &k)const {
        return Point(x / k, y / k);
    }
    //`¼□ □ □ pa  °□   pb µļн□ `
    double rad(Point a,Point b) {
        Point p = *this;
        return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p) ));
    }
    //`»¯□  ³¤¶□ □ rµ□ □ □ -`
    Point trunc(double r) {
        double l = len();
        if(!sgn(l)) return *this;
        r /= l;
        return Point(x*r,y*r);
    }
    //`□ □ ʱ□ □ □ □ ¶90□ □ `
    Point rotleft() { return Point(-y,x); }
    //`。ʱ□ □ □ □ ¶90□ □ `
    Point rotright() { return Point(y,-x); }
    //`□ □ □ □ pµ□ □ □ ʱ□ □ □ □ angle□ `
    Point rotate(Point p,double angle) {
        Point v = (*this) - p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x * c - v.y * s,
                     p.y + v.x * s + v.y * c);
    }
};
/*
 * Stores two points
 * Line()                         - Empty constructor
 * Line(Point _s,Point _e)        - Line through _s and _e
 * operator ==                    - checks if two points are same
 * Line(Point p,double angle)     - one end p , another end at angle degree
 * Line(double a,double b,double c) - Line of equation ax + by + c = 0
 * input()                        - inputs s and e
 * adjust()                       - orders in such a way that s < e
 * length()                       - distance of se
 * angle()                        - return 0 <= angle < pi
 * relation(Point p)              - 3 if point is on line
 *                                  1 if point on the left of line
 *                                  2 if point on the right of line
 * pointonseg(double p)           - return true if point on segment
 * parallel(Line v)               - return true if they are parallel
 * segcrossseg(Line v)            - returns 0 if does not intersect
 *                                  returns 1 if non-standard intersection
 *                                  returns 2 if intersects
 * linecrossseg(Line v)           - line and seg
 * linecrossline(Line v)          - 0 if parallel
 *                                  1 if coincides
 *                                  2 if intersects
 * crosspoint(Line v)             - returns intersection point
 * dispointtoline(Point p)        - distance from point p to the line
 * dispointtoseg(Point p)         - distance from p to the segment
 * dissegtoseg(Line v)            - distance of two segment
 * lineprog(Point p)              - returns projected point p on se line
 * symmetrypoint(Point p)         - returns reflection point of p over se
 *
 */
struct Line {
    Point s, e;
    Line() {}
    Line(Point _s,Point _e) { s = _s; e = _e; }
    bool operator ==(Line v) {
        return (s == v.s) && (e == v.e);
    }
    //`¸□ ¾□ h¸□ □ □ □ □ 6½□ angleȷ¶□ □ □ □ ,0 <= angle < pi`
    Line(Point p, double angle) {
        s = p;
        if(sgn(angle-pi/2) == 0) e = (s + Point(0,1));
        else e = (s + Point(1, tan(angle)));
    }
    //ax + by + c = 0
    Line(double a, double b, double c) {
        if(sgn(a) == 0) {
            s = Point(0,-c/b);
            e = Point(1,-c/b);
        } else if(sgn(b) == 0) {
            s = Point(-c/a,0);
            e = Point(-c/a,1);
        } else {
            s = Point(0,-c/b);
            e = Point(1,(-c-a)/b);
        }
    }
    void input() { s.input(); e.input(); }
    void adjust() { if(e < s) swap(s, e); }
    //□ □ □ □ □ ¤¶□
    double length() { return s.distance(e); }
    //`·µ»□ □ □ □ □ □ 6½□  0 <= angle < pi`
    double angle() {
        double k = atan2(e.y-s.y,e.x-s.x);
        if(sgn(k) < 0) k += pi;
        if(sgn(k-pi) == 0) k -= pi;
        return k;
    }
    //`µ□ □ □ □ □ □ □ `1  □ □ □ □ □ `2  □ □ □ Ҳ□ `3  □ □ □ □ □ □ □ `
    int relation(Point p) {
        int c = sgn((p-s)^(e-s));
        if(c < 0) return 1;
        else if(c > 0) return 2;
        else return 3;
    }
    // µ□ □ □ □ □ □ □ □ □ □ ж□
    bool pointonseg(Point p) {
        return sgn((p - s) ^ (e - s)) == 0 &&
               sgn((p - s) * (p - e)) <= 0;
    }
    //`}□ □ -□ □ □ (¶□ □ □ □ □ □ □ л□ □ □ □ )`
    bool parallel(Line v) {
        return sgn((e - s) ^ (v.e - v.s)) == 0;
    }
    //`}□ □ □ □ □ □ ж□ `2 ¹涞□ □ `1 ·ṅ涞□ □ `0 ²»□ □ `
    int segcrossseg(Line v) {
        int d1 = sgn((e - s) ^ (v.s - s));
        int d2 = sgn((e - s) ^ (v.e - s));
        int d3 = sgn((v.e - v.s) ^ (s - v.s));
        int d4 = sgn((v.e - v.s) ^ (e - v.s));
        if( (d1 ^ d2) == -2 && (d3 ^ d4) == -2 )return 2;
        return (d1 == 0 && sgn((v.s - s) * (v.s - e)) <= 0) ||
               (d2 == 0 && sgn((v.e - s) * (v.e - e)) <= 0) ||
               (d3 == 0 && sgn((s - v.s) * (s - v.e)) <= 0) ||
               (d4 == 0 && sgn((e - v.s) * (e - v.e)) <= 0);
    }
    //`□ □ □ □ □ □ □ □ □ □ ж□ `2 ¹涞□ □ `1 ·ṅ涞□ □ `0 ²»□ □ `
    int linecrossseg(Line v) {
        int d1 = sgn((e - s) ^ (v.s - s));
        int d2 = sgn((e - s) ^ (v.e - s));
        if((d1 ^ d2) == -2) return 2;
        return (d1 == 0 || d2 ==0 );
    }
    //`}□ □ □ □ □ `0 □ □ □ `1 □ □ □ `2 □ □ `
    int linecrossline(Line v) {
        if((*this).parallel(v))
            return v.relation(s) == 3;
        return 2;
    }
    //`□ □ }□ □ Ľ»µ□ □ `Ç±□ □ }□ □ □ »□ □ л□ □ □ □ `
    Point crosspoint(Line v) {
        double a1 = (v.e - v.s) ^ (s - v.s);
        double a2 = (v.e - v.s) ^ (e - v.s);
        return Point((s.x * a2 - e.x * a1) / (a2 - a1),
                        (s.y * a2 - e.y * a1) / (a2 - a1));
    }
    //μ㶠▯ ▯ ľ▯ ▯ ▯ ▯
    double dispointtoline(Point p) {
        return fabs((p - s) ^ (e - s)) / length();
    }
    //μ㶠▯ ▯ ▯ ľ▯ ▯ ▯
    double dispointtoseg(Point p) {
        if(sgn((p - s) * (e - s)) < 0 || sgn((p - e) * (s - e)) < 0)
            return min(p.distance(s), p.distance(e));
        return dispointtoline(p);
    }
    //`·μ»▯ ▯ ▯ ▯ ½▯ ▯ ▯ ľ▯ ▯ ▯ `ǰ▯ ▯ ▯ ▯ }▯ ▯ ▯ »▯ ▯ £¬▯ ▯ ¾▯ ▯ ▯ ▯ ▯ ▯ 0▯ ▯ `
    double dissegtoseg(Line v) {
        return min(min(dispointtoseg(v.s), dispointtoseg(v.e)),
                   min(v.dispointtoseg(s), v.dispointtoseg(e)));
    }
    //`·μ»▯ ▯ p▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ ÿ`
    Point lineprog(Point p) {
        return s + (((e - s) * ((e - s) * (p - s))) / ((e - s).len2()));
    }
    //`·μ»▯ ▯ p¹▯ ▯ ▯ ▯ ▯ Ķ▯ ƶ▯ ▯ `
    Point symmetrypoint(Point p) {
        Point q = lineprog(p);
        return Point(2 * q.x - p.x, 2 * q.y - p.y);
    }
};
//▯
struct circle {
    Point p;
    double r;
    circle() {}
    circle(Point _p,double _r) { p = _p; r = _r; }
    circle(double x,double y,double _r) {
        p = Point(x,y);
        r = _r;
    }
    //`▯ ▯ ½▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ `▯ ▯ ▯ ▯ }▯ ▯ ▯ ▯ ▯ д¹▯ õ½▯ ▯ ▯ ▯ `
    circle(Point a,Point b,Point c) {
        Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
        Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
        p = u.crosspoint(v);
        r = p.distance(a);
    }
    //`▯ ▯ ½▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ `²▯ ▯ ▯ bool tû▯ ▯ ▯ ▯ ▯ ã▯ ▯ ▯ ▯ ▯ ▯ ˥▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ ▯ °-
    ▯ ▯ ▯ ▯ ±▯ `
    circle(Point a,Point b,Point c,bool t) {
        Line u, v;
        double m = atan2(b.y - a.y, b.x - a.x),
               n = atan2(c.y - a.y, c.x - a.x);
        u.s = a;
        u.e = u.s + Point(cos((n + m) / 2), sin((n + m) / 2));
        v.s = b;
        m = atan2(a.y - b.y, a.x - b.x),
        n = atan2(c.y - b.y, c.x - b.x);
        v.e = v.s + Point(cos((n + m) / 2), sin((n + m) / 2));
        p = u.crosspoint(v);
        r = Line(a, b).dispointtoseg(p);
    }
    }
    bool operator == (circle v) {
        return (p == v.p) && sgn(r - v.r) == 0;
    }
    bool operator < (circle v)const {
        return ((p < v.p) || ((p == v.p) && sgn(r - v.r) < 0));
    }
    double area() { return pi * r * r; }
    double circumference() {//□ □ ¤
        return 2*pi*r;
    }
    //`µ□ □ □ µĹ□ □ `0 □ □ □ `1 □ □ □ `2 □ □ □ `
    int relation(Point b) {
        double dst = b.distance(p);
        if(sgn(dst - r) < 0) return 2;
        else if(sgn(dst - r) == 0) return 1;
        return 0;
    }
    //`□ □ □ □ □ µĹ□ □ `±Ł□ □ □ □ □ □ ĵ½□ □ □ ľ□ □ □ □ □µĹ□ □ `
    int relationseg(Line v) {
        double dst = v.dispointtoseg(p);
        if(sgn(dst - r) < 0) return 2;
        else if(sgn(dst - r) == 0) return 1;
        return 0;
    }
    //`□ □ □ □ □ µĹ□ □ `±Ł□ □ □ □ □ □ ĵ□ □ □ ľ□ □ □ □ □ □µĹ□ □ `
    int relationline(Line v) {
        double dst = v.dispointtoline(p);
        if(sgn(dst - r) < 0) return 2;
        else if(sgn(dst - r) == 0) return 1;
        return 0;
    }
    //`}□ µĹ□ □ `5 □ □ □ □ `4 □ □ □ □ `3 □ □ `2 □ □ □ □ `1 □ ¬□ `
    int relationcircle(circle v) {
        double d = p.distance(v.p);
        if(sgn(d - r - v.r) > 0) return 5;
        if(sgn(d - r - v.r) == 0) return 4;
        double l = fabs(r - v.r);
        if(sgn(d - r - v.r) < 0 && sgn(d - l) > 0) return 3;
        if(sgn(d - l) == 0) return 2;
        if(sgn(d - l) < 0) return 1;
    }
    //`□ □ }¸□ □ µĽ»µ徨·µ»□ 0±□ ʾû□ н»µ徨·µ»□ 1□ □ һ¸□ 徨2□ □ }¸□ □ `
    int pointcrosscircle(circle v, Point &p1, Point &p2) {
        int rel = relationcircle(v);
        if(rel == 1 || rel == 5)return 0;
        double d = p.distance(v.p);
        double l = (d*d+r*r-v.r*v.r)/(2*d);
        double h = sqrt(r*r-l*l);
        Point tmp = p + (v.p-p).trunc(l);
        p1 = tmp + ((v.p-p).rotleft().trunc(h));
        p2 = tmp + ((v.p-p).rotright().trunc(h));
        if(rel == 2 || rel == 4) return 1;
        return 2;
    }
    //`□ □ □ □ □ □ □ µĽ»µ徨·µ»□ »µ□ □ □ □ `
    int pointcrossline(Line v,Point &p1,Point &p2) {
        if(!(*this).relationline(v)) return 0;
        Point a = v.lineprog(p);
        double d = v.dispointtoline(p);
        d = sqrt(r * r - d * d);
        if(sgn(d) == 0) { p1 = p2 = a; return 1; }
        p1 = a + (v.e-v.s).trunc(d);
        p2 = a - (v.e-v.s).trunc(d);
        return 2;
    }
    //`µõ½¹▯ a,b}µ�George°□▯ r1µ▯ }¸▯ ▯ `
    int gercircle(Point a,Point b,double r1,circle &c1,circle &c2) {
        circle x(a,r1),y(b,r1);
        int t = x.pointcrosscircle(y,c1.p,c2.p);
        if(!t) return 0;
        c1.r = c2.r = r1;
        return t;
    }
    //`µõ½▯ ▯ ▯ ▯ ▯ u▯ ▯ ▯ У¬¹▯ µ▯ q,°□▯ r1µ▯ ▯ `
    int getcircle(Line u,Point q,double r1,circle &c1,circle &c2) {
        double dis = u.dispointtoline(q);
        if(sgn(dis - r1 * 2) > 0) return 0;
        if(sgn(dis) == 0) {
            c1.p = q + ((u.e - u.s).rotleft().trunc(r1));
            c2.p = q + ((u.e - u.s).rotright().trunc(r1));
            c1.r = c2.r = r1;
            return 2;
        }
        Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),
                       (u.e + (u.e-u.s).rotleft().trunc(r1)));
        Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),
                       (u.e + (u.e-u.s).rotright().trunc(r1)));
        circle cc = circle(q,r1);
        Point p1,p2;
        if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
        c1 = circle(p1,r1);
        if(p1 == p2) { c2 = c1; return 1; }
        c2 = circle(p2,r1);
        return 2;
    }
    //`ʳɦ▯ ▯ ▯ ▯ ▯ u,v▯ ▯ ▯ У¬°□▯ r1µ▯ ▯ `
    int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4)
    {
        if(u.parallel(v))return 0;//}▯ ▯ ▯ ▯ ▯ ▯
        Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),
                       u.e + (u.e-u.s).rotleft().trunc(r1));
        Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),
                       u.e + (u.e-u.s).rotright().trunc(r1));
        Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),
                       v.e + (v.e-v.s).rotleft().trunc(r1));
        Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),
                       v.e + (v.e-v.s).rotright().trunc(r1));
        c1.r = c2.r = c3.r = c4.r = r1;
        c1.p = u1.crosspoint(v1);
        c2.p = u1.crosspoint(v2);
        c3.p = u2.crosspoint(v1);
        c4.p = u2.crosspoint(v2);
        return 4;
    }
    //`ʳɦ▯ □▯ ▯ ▯ cx,cy▯ ▯ ▯ У¬°□▯ r1µ▯ ▯ `
    int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2) {
```

```
    circle x(cx.p, r1 + cx.r), y(cy.p, r1 + cy.r);
    int t = x.pointcrosscircle(y, c1.p, c2.p);
    if(!t) return 0;
    c1.r = c2.r = r1;
    return t;
}

//`¹□ һµ□ □ □ □ µ□ □ □ □ □ (□ □ □ ж□ □ □ □ µĹ□ □ )`
int tangentline(Point q,Line &u,Line &v) {
    int x = relation(q);
    if(x == 2) return 0;
    if(x == 1) {
        u = Line(q, q + (q - p).rotleft());
        v = u;
        return 1;
    }
    double d = p.distance(q);
    double l = r * r / d;
    double h = sqrt(r * r - l * l);
    u = Line(q, p + ((q - p).trunc(l) + (q - p).rotleft().trunc(h)));
    v = Line(q, p + ((q - p).trunc(l) + (q - p).rotright().trunc(h)));
    return 2;
}
//`□ □ }□ □ □ µ□ □ □ □ `
double areacircle(circle v) {
    int rel = relationcircle(v);
    if(rel >= 4) return 0.0;
    if(rel <= 2) return min(area(),v.area());
    double d = p.distance(v.p);
    double hf = (r + v.r + d) / 2.0;
    double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) * (hf - d));
    double a1 = acos((r * r + d * d - v.r * v.r) / (2.0 * r * d));
    a1 = a1 * r * r;
    double a2 = acos((v.r * v.r + d * d - r * r) / (2.0 * v.r * d));
    a2 = a2 * v.r * v.r;
    return a1 + a2 - ss;
}
//`□ □ }□ □ □ µ□ □ □ □ (¾«¶ȸ□ ¸□ )(□ □ Çlong double)`
double areacircle2(circle v) {
    double a = hypot(p.x-v.p.x,p.y-v.p.y),b=r,c=v.r;
    double s1 = pi * r* r, s2 = pi * v.r * v.r;
    if(sgn(a - b - c) >= 0) return 0;
    if(sgn(a + min(b,c) - max(b,c)) <= 0) return min(s1, s2);
    else {
        double cta1 = 2 * acos((a * a + b * b - c * c) / (2 * a * b));
        double cta2 = 2 * acos((a * a + c * c - b * b) / (2 * a * c));
        return cta1 / (2 * pi) * s1 - 0.5 * sin(cta1) * b * b +
               cta2 / (2 * pi) * s2 - 0.5 * sin(cta2) * c * c;
    }
}
//`□ □ □ °□ □ □ ½□ □ □ pabµ□ □ □ □ □ □ `
double areatriangle(Point a,Point b) {
    if(sgn((p - a) ^ (p - b)) == 0) return 0.0;
    Point q[5];
    int len = 0;
    q[len ++] = a;
    Line l(a,b);
    Point p1,p2;
    if(pointcrossline(l, q[1], q[2]) == 2) {
            if(sgn((a - q[1]) * (b - q[1])) < 0) q[len ++] = q[1];
            if(sgn((a - q[2]) * (b - q[2])) < 0) q[len ++] = q[2];
        }
        q[len ++] = b;
        if(len == 4 && sgn((q[0] - q[1]) * (q[2] - q[1])) > 0) swap(q[1], q[2]);
        double res = 0;
        for(int i = 0; i < len - 1; ++ i) {
            if(relation(q[i]) == 0 || relation(q[i + 1]) == 0) {
                double arg = p.rad(q[i], q[i + 1]);
                res += r * r * arg / 2.0;
            } else res += fabs((q[i] - p) ^ (q[i + 1] - p)) / 2.0;
        }
        return res;
    }
};
/*
 * n,p  Line l for each side
 * input(int _n)                        - inputs _n size polygon
 * add(Point q)                         - adds a point at end of the list
 * getline()                            - populates line array
 * cmp                                  - comparision in convex_hull order
 * norm()                               - sorting in convex_hull order
 * getconvex(polygon &convex)           - returns convex hull in convex
 * Graham(polygon &convex)              - returns convex hull in convex
 * isconvex()                           - checks if convex
 * relationpoint(Point q)               - returns 3 if q is a vertex
 *                                                2 if on a side
 *                                                1 if inside
 *                                                0 if outside
 * convexcut(Line u,polygon &po)        - left side of u in po
 * gercircumference()                   - returns side length
 * getarea()                            - returns area
 * getdir()                             - returns 0 for cw, 1 for ccw
 * getbarycentre()                      - returns barycenter
 *
 */
struct polygon {
    int n;
    Point p[maxp];
    Line l[maxp];
    void input(int _n) {
        n = _n;
        for(int i = 0; i < n; i++)
            p[i].input();
    }
    void add(Point q) {
        p[n ++] = q;
    }
    void getline() {
        for(int i = 0; i < n; i++) {
            l[i] = Line(p[i],p[(i+1)%n]);
        }
    }
    struct cmp {
        Point p;
        cmp(const Point &p0) { p = p0; }
        bool operator()(const Point &aa,const Point &bb) {
            Point a = aa, b = bb;
            int d = sgn((a-p)^(b-p));
            if(d == 0) {
                return sgn(a.distance(p)-b.distance(p)) < 0;
            }
            return d > 0;
        }
    };
    //`½□ □ м«½□ □ □ □ □ `
    //`□ □ □ □ □ □ Ç□ □ ½□ □ □ □ □ ½ģĵ□ `
    //`□ □ Ç□ □ □ ź□ □ Pointµ□  < ²□ □ □ □ (min°¯□ □ Ç□ □ ) `
    void norm() {
        Point mi = p[0];
        for(int i = 1; i < n; i++) mi = min(mi, p[i]);
        sort(p, p + n, cmp(mi));
    }
    //`µõ½□ °□ `
    //`µõ½µ□ □ °□ □ □ □ □ ĵ□ □ □ □ □ 0$\sim$n-1µ□ `
    //`}□ □ □ °□ µķ½·¨`
    //`□ □ □ □ □ □ □ □ Ÿ□ □Ç□ □ □ □ □ □ □ □ □ e狰µ�israel»□ □ □ ²□ □ □ □ □ □ □ □ □ □ `
    void getconvex(polygon &convex) {
        sort(p,p+n);
        convex.n = n;
        for(int i = 0; i < min(n,2); i++) {
            convex.p[i] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//□ □ □ □
        if(n <= 2)return;
        int &top = convex.n;
        top = 1;
        for(int i = 2; i < n; i++) {
            while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
                top--;
            convex.p[++top] = p[i];
        }
        int temp = top;
        convex.p[++top] = p[n-2];
        for(int i = n-3; i >= 0; i--) {
            while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
                top--;
            convex.p[++top] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//□ □ □ □
        convex.norm();//`□ 4µõ½µ□ □ □ 。ĥ□ □ ĵ�israel□ □ □ □ □ □ ĥ□ □ `
    }
    //`µõ½□ °□ µ□ □ □ □ □ ћ□ □ ½·¨`
    void Graham(polygon &convex) {
        norm();
        int &top = convex.n;
        top = 0;
        if(n == 1) {
            top = 1;
            convex.p[0] = p[0];
            return;
        }
        if(n == 2) {
            top = 2;
            convex.p[0] = p[0];
            convex.p[1] = p[1];
            if(convex.p[0] == convex.p[1])top--;
            return;
        }
        convex.p[0] = p[0];
        convex.p[1] = p[1];
        top = 2;
        for(int i = 2; i < n; i++) {
            while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-convex.p[top-2])) <= 0 )
                top--;
            convex.p[top++] = p[i];
        }
        if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//□ □ □ □
    }
    //`□ ж□ □ □z»□ □ □ µ□ `
    bool isconvex() {
        bool s[3];
        memset(s,false,sizeof(s));
        for(int i = 0; i < n; i++) {
            int j = (i+1)%n;
            int k = (j+1)%n;
            s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
            if(s[0] && s[2])return false;
        }
        return true;
    }
    //`□ ж□ □ □ □ □ □ □ □ □ □ □ Ĺ□ □ `3 µ□ □ □ `2 ±□ □ □ `1 □ ¿□ `0 □ □ `
    int relationpoint(Point q) {
        for(int i = 0; i < n; i++) {
            if(p[i] == q)return 3;
        }
        getline();
        for(int i = 0; i < n; i++) {
            if(l[i].pointonseg(q))return 2;
        }
        int cnt = 0;
        for(int i = 0; i < n; i++) {
            int j = (i+1)%n;
            int k = sgn((q-p[j])^(p[i]-p[j]));
            int u = sgn(p[i].y-q.y);
            int v = sgn(p[j].y-q.y);
            if(k > 0 && u < 0 && v >= 0)cnt++;
            if(k < 0 && v < 0 && u >= 0)cnt--;
        }
        return cnt != 0;
    }
    //`□ □ □ u□ и□ □ ¶□ □ □ □ □ □ □ `
    //`□ □ □ □ □ □ ½□ □ `
    void convexcut(Line u,polygon &po) {
        int &top = po.n;//□ □ □ □ □ □ □
        top = 0;
        for(int i = 0; i < n; i++) {
            int d1 = sgn((u.e-u.s)^(p[i]-u.s));
            int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
            if(d1 >= 0)po.p[top++] = p[i];
            if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i+1)%n]));
        }
    }
    //`µõ½□ □ ¤`
    double getcircumference() {
        double sum = 0;
        for(int i = 0; i < n; i++) {
            sum += p[i].distance(p[(i+1)%n]);
        }
        return sum;
    }
    //`µõ½□ □ □ `
    double getarea() {
        double sum = 0;
        for(int i = 0; i < n; i++) {
            sum += (p[i]^p[(i+1)%n]);
        }
        return fabs(sum)/2;
    }
    //`µõ½·½□ □ `1 ±□ ¸□ □ ʱ□ □0±□ ¸。ʱ□ □ `
    bool getdir() {
        double sum = 0;
        for(int i = 0; i < n; i++)
            sum += (p[i]^p[(i+1)%n]);
        if(sgn(sum) > 0)return 1;
        return 0;
    }
    //`µõ½□ □ □ □ `
    Point getbarycentre() {
        Point ret(0,0);
        double area = 0;
        for(int i = 1; i < n-1; i++) {
            double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
            if(sgn(tmp) == 0)continue;
            area += tmp;
            ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
            ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
        }
        if(sgn(area)) ret = ret/area;
        return ret;
    }
    //`¶□ □ □ □ □ □ ½»µ□ □ □ □ `
    double areacircle(circle c) {
        double ans = 0;
        for(int i = 0; i < n; i++) {
            int j = (i+1)%n;
            if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
                ans += c.areatriangle(p[i],p[j]);
            else ans -= c.areatriangle(p[i],p[j]);
        }
        return fabs(ans);
    }
    //`¶□ □ □ □ □ □ ¹□ □ `
    //` 2 □ □ □ ő□ □ □ □ □ □ □ □ `1 □ □ □ □ □ □ □ □ □ □ 棬□ □ □ ″□ □ □ □ □ □ `0 □ □ □ □ `
    int relationcircle(circle c) {
        getline();
        int x = 2;
        if(relationpoint(c.p) != 1)return 0;//□ □ Ĳ»□ □ □ ¿□
        for(int i = 0; i < n; i++) {
            if(c.relationseg(l[i])==2)return 0;
            if(c.relationseg(l[i])==1)x = 1;
        }
        return x;
    }
};
```

```
//`AB X AC`
double cross(Point A,Point B,Point C) {
    return (B-A)^(C-A);
}
//`AB*AC`
double dot(Point A,Point B,Point C) {
    return (B-A)*(C-A);
}
//`□ □ C¾□ □ □ □ □ □ ¸²¸□ `
//` A ±□ □ □ □ □ □ °□ (¶□ □ □ □ □ □ □ ɦ□ □ ｡□ □ )`
double minRectangleCover(polygon A) {
    //`Ç□ □ □ □ A.n < 3µ□ □ □ □ `
    if(A.n < 3)return 0.0;
    A.p[A.n] = A.p[0];
    double ans = -1;
    int r = 1, p = 1, q;
    for(int i = 0; i < A.n; i++) {
        //`¿¨³□ □ □ □ A.p[i] - A.p[i+1]□ □ □ µĵ□ `
        while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) )
    >= 0 )
            r = (r+1)%A.n;
        //`¿¨³□ A.p[i] - A.p[i+1]·½□ □ □ □ □ □ □ □ n□ □ □ µĵ□ `
        while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
            p = (p+1)%A.n;
        if(i == 0)q = p;
        //`¿¨³□ A.p[i] - A.p[i+1]·½□ □ □ □ °□ □ □ □ □ µĵ□ `
        while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
            q = (q+1)%A.n;
        double d = (A.p[i] - A.p[i+1]).len2();
        double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
                     (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
        if(ans < 0 || ans > tmp)ans = tmp;
    }
    return ans;
}
//`□ □ □ □ □ □ ¶□ □ □ □ `
//`¶□ □ □ □ □ □ □ □ ɦ□ □ ǵ¬□ □ q1q2µ□ □ □ □ `
vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2) {
    vector<Point>qs;
    int n = ps.size();
    for(int i = 0; i < n; i++) {
        Point p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = sgn((q2-q1)^(p1-q1)), d2 = sgn((q2-q1)^(p2-q1));
        if(d1 >= 0)
            qs.push_back(p1);
        if(d1 * d2 < 0)
            qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
    }
    return qs;
}
//`°□ □ □ 漆`
struct halfplane:public Line {
    double angle;
    halfplane() {}
    //`±□ ʾ□ □ -s->e□ □ ɦ□ □ (□ □ □ )µİ□ □ □ □ `
    halfplane(Point _s,Point _e) {
        s = _s;
        e = _e;
    }
    halfplane(Line v) {
        s = v.s;
        e = v.e;
    }
    void calcangle() {
        angle = atan2(e.y-s.y,e.x-s.x);
    }
    bool operator <(const halfplane &b)const {
        return angle < b.angle;
    }
};
struct halfplanes {
    int n;
    halfplane hp[maxp];
    Point p[maxp];
    int que[maxp];
    int st,ed;
    void push(halfplane tmp) {
        hp[n++] = tmp;
    }
    //ȥ□ □
    void unique() {
        int m = 1;
        for(int i = 1; i < n; i++) {
            if(sgn(hp[i].angle-hp[i-1].angle) != 0)
                hp[m++] = hp[i];
            else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) ) > 0)
                hp[m-1] = hp[i];
        }
        n = m;
    }
    bool halfplaneinsert() {
        for(int i = 0; i < n; i++)hp[i].calcangle();
        sort(hp,hp+n);
        unique();
        que[st=0] = 0;
        que[ed=1] = 1;
        p[1] = hp[0].crosspoint(hp[1]);
        for(int i = 2; i < n; i++) {
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;
            while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;
            que[++ed] = i;
            if(hp[i].parallel(hp[que[ed-1]]))return false;
            p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
        }
        while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed--;
        while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)st
++;
        if(st+1>=ed)return false;
        return true;
    }
    //`µõ½□ □ □ □ □ □ 漆µõ½µ□ □ ¶□ □ □ □ `
    //`□ □ Ç□ □ □ □ □ halfplaneinsert() □ ҷµ»□ true`
    void getconvex(polygon &con) {
        p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
        con.n = ed-st+1;
        for(int j = st,i = 0; j <= ed; i++,j++)
            con.p[i] = p[j];
    }
        double minRectangleCover(polygon A) {
        //`Ç□ □ □ □ A.n < 3µ□ □ □ □ `
        if(A.n < 3)return 0.0;
        A.p[A.n] = A.p[0];
        double ans = -1;
        int r = 1, p = 1, q;
        for(int i = 0; i < A.n; i++) {
            //`¿¨³□ □ □ □ A.p[i] - A.p[i+1]□ □ □ µĵ□ `
            while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) ) >= 0 )
                r = (r+1)%A.n;
            //`¿¨³□ A.p[i] - A.p[i+1]·½□ □ □ □ □ □ □ □ n□ □ □ µĵ□ `
            while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
                p = (p+1)%A.n;
            if(i == 0)q = p;
            //`¿¨³□ A.p[i] - A.p[i+1]·½□ □ □ □ °□ □ □ □ □ µĵ□ `
            while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
                q = (q+1)%A.n;
            double d = (A.p[i] - A.p[i+1]).len2();
            double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
                         (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
            if(ans < 0 || ans > tmp)ans = tmp;
        }
        return ans;
    }
    circle minCircleCover(int n, Point p[], Point P = Point(0, 0)) {
        random_shuffle(p, p + n); double r2 = 0;
        for(int i = 0; i < n; ++ i) {
            if((p[i] - P).len2() > r2) {
                P = p[i], r2 = 0;
                for(int j = 0; j < i; ++ j) {
                    if((p[j]-P).len2() > r2) {
                        P = (p[i]+p[j])/2, r2 = (p[j]-P).len2();
                        for(int k = 0; k < j; ++ k) {
                            if((p[k]-P).len2() > r2) {
                                P = circle(p[i], p[j], p[k]).p, r2 = (p[k] - P).len2();
                            }
                        }
                    }
                }
            }
        }
        return circle(P, sqrt(r2));
    }
};
```

4.31 3 维计算几何

```
struct Point3 {
    double x, y, z;
    Point3(double xx = 0, double yy = 0,double zz = 0) { x = xx, y = yy, z = zz; }
    void input() { cin >> x >> y >> z; }
    void output(void) { cout << fixed << setprecision(3) << x << ' ' << y << ' ' << z << '\n'; }
    double len(void) { return sqrt(x * x + y * y + z * z); }
    double len2(void) { return x * x + y * y + z * z; }
    double dis(const Point3 &b) const { return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-
b.z)*(z-b.z)); }
    bool operator ==(const Point3 &b) const { return sgn(x-b.x)==0&&sgn(y-b.y)==0&&sgn(
z-b.z)==0; }
    bool operator <(const Point3 &b) const {
        if(sgn(x-b.x)!=0) return x < b.x;
        if(sgn(y-b.y)!=0) return y < b.y;
        return sgn(z-b.z) < 0;
    }
    Point3 operator + (const Point3 &b) const { return Point3(x+b.x,y+b.y,z+b.z); }
    Point3 operator - (const Point3 &b) const { return Point3(x-b.x,y-b.y,z-b.z); }
    Point3 operator * (const double &k) const { return Point3(x*k,y*k,z*k); }
    Point3 operator / (const double &k) const { return Point3(x/k,y/k,z/k); }
    Point3 operator ^ (const Point3 &b) const { return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x
*b.y-y*b.x); }
    double operator * (const Point3 &b) const { return x*b.x+y*b.y+z*b.z; }
};
struct CH3D {
    struct face {
        int a, b, c;//±□ ’□ °□ h¸□ □ □ □ □ □ □ □ ¸□ □ ı□ □
        bool ok;//±□ ’¸□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ °□ □ □ □ □ □
    };
    int n;//³□ ’¶¥µ□ □ □
    Point3 P[maxn];
    int num;//□ °□ ±□ □ □ □ □ □ ½□ □ □ □ □
    face F[maxn<<3];//□ °□ ±□ □ □ □ □ □ ½□ □ □
    int g[maxn][maxn];
    Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c) { return (b-a)^(c-a);
 }
    //□ □ ½□ □ □ □ □ □ *2
    double area_triangle(Point3 a,Point3 b,Point3 c) { return ((b-a)^(c-a)).len(); }
    //□ □ □ □ □ □ □ □ □ □ □ □ □ *6
    double volume_four(Point3 a,Point3 b,Point3 c,Point3 d) { return ((b-a)^(c-a))*(d-a
); }
    //□ □ £°µ□ □ □ □ □ ʳ□ □
    double dblcmp(Point3 &p,face &f) {
        Point3 p1=P[f.b]-P[f.a];
        Point3 p2=P[f.c]-P[f.a];
        Point3 p3=p-P[f.a];
        return (p1^p2)*p3;
    }
    void deal(int p,int a,int b) {
        int f=g[a][b];
        face add;
        if(F[f].ok) {
            if(sgn(dblcmp(P[p],F[f]))>0)
                dfs(p,f);
            else {
                add.a=b; add.b=a; add.c=p;
                add.ok=true;
                g[p][b]=g[a][p]=g[b][a]=num;
                F[num++]=add;
            }
        }
    }
    //µ□ □ □ □ □ □ □ □ □ □ Ö¸ô□ □ °□ □ □ ɼ³□ µ□ □ □
    void dfs(int p,int now) {
        F[now].ok = false;
        deal(p,F[now].b,F[now].a);
        deal(p,F[now].c,F[now].b);
        deal(p,F[now].a,F[now].c);
    }
    bool same(int s,int t) {
        Point3 &a=P[F[s].a];
        Point3 &b=P[F[s].b];
        Point3 &c=P[F[s].c];
        int d1=sgn(volume_four(a,b,c,P[F[t].a]));
        int d2=sgn(volume_four(a,b,c,P[F[t].b]));
        int d3=sgn(volume_four(a,b,c,P[F[t].c]));
        return (d1==0)&&(d2==0)&&(d3==0);
    }
    //¹¹½¨□ □ □ □ °□
    void create() {
        num=0;
        face add;
        //±□ □ ǰ□ κ□ 伙¹²□ □
        //**********************************
        bool flag=true;
        for(int i=1;i<n;i++) {
            if(!(P[0]==P[i])) {
                swap(P[1],P[i]);
                flag=false;
                break;
            }
        }
        if(flag) return ;
        flag=true;
        for(int i=2;i<n;i++) {
            if(sgn(((P[1]-P[0])^(P[i]-P[0])).len())>0) {
                swap(P[2],P[i]);
                flag=false;
                break;
            }
        }
        if(flag) return ;
        flag=true;
        for(int i=3;i<n;i++) {
            if(sgn(((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]))!=0) {
                swap(P[3],P[i]);
                flag=false;
                break;
            }
        }
        if(flag) return ;
        //**********************************
        for(int i=0;i<4;i++) {
            add.a=(i+1)%4;
            add.b=(i+2)%4;
            add.c=(i+3)%4;
            add.ok=true;
            if(sgn(dblcmp(P[i],add))>0) swap(add.b,add.c);
            g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;
            F[num++]=add;
        }
        for(int i=4;i<n;i++) {
            for(int j=0;j<num;j++) {
                if(F[j].ok&&sgn(dblcmp(P[i],F[j]))>0) {
                    dfs(i,j);
                    break;
                }
            }
        }
        int tmp=num;
        num=0;
        for(int i=0;i<tmp;i++)
            if(F[i].ok)
                F[num++]=F[i];
    }
    //±□ □ □ □
    double area(void) {
        double res=0;
        if(n==3) {
            Point3 p=cross(P[0],P[1],P[2]);
            return p.len()/2;
        }
        for(int i=0;i<num;i++)
            res+=area_triangle(P[F[i].a],P[F[i].b],P[F[i].c]);
        return res/2.0;
    }
    //□ □ □
    double volume(void) {
        double res=0;
        Point3 tmp=Point3(0,0,0);
        for(int i=0;i<num;i++)
            res+=volume_four(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
        return abs(res/6.0);
    }
    //±□ □ □ □ □ ½□ □ □ □ □ □
    int sum_of_triangle(void) {
        return num;
    }
    //±□ □ □ □ □ □ □ □ □ □
    int sum_of_polygon(void) {
        int res=0;
        for(int i=0;i<num;i++) {
            bool flag=true;
            for(int j=0;j<i;j++) {
                if(same(i,j)) {
                    flag=false;
                    break;
                }
            }
            res+=flag;
        }
        return res;
    }
    //□ □ □ □
    Point3 bary_center(void) {
        Point3 ans=Point3(0,0,0);
        Point3 o=Point3(0,0,0);
        double all=0;
        for(int i=0;i<num;i++) {
            double vol=volume_four(o,P[F[i].a],P[F[i].b],P[F[i].c]);
            ans=ans+(((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*vol);
            all+=vol;
        }
        ans=ans/all;
        return ans;
    }
    //µ漚□ □ l’□ □ □
    double dis_point_to_face(Point3 p,int i) {
        double tmp1=abs(volume_four(P[F[i].a],P[F[i].b],P[F[i].c],p));
        double tmp2=((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a])).len();
        return tmp1/tmp2;
    }
};
```

4.32 三分套三分

```
int n; double maxx = -double(inf);
double xmax = -10000000, xmin = 10000000, ymax = -10000000, ymin = 10000000;

double check(Point P) {
    double minn = double(inf);
    for(int i = 1; i <= n; ++ i) {
        minn = min(minn, P.rad(p[i], p[i + 1]));
    }
    return minn;
}

double Find(double x) {
    double yl = ymin, yr = ymax, midl, midr;
    while(yr - yl > eps) {
        midl = (yl + yl + yr) / 3.0;
        midr = (yl + yr + yr) / 3.0;
        if(check(Point(x, midl)) < check(Point(x, midr))) yl = midl;
        else yr = midr;
    }
    maxx = max(maxx, check(Point(x, yl)));
    return check(Point(x, yl));
}

inline void solve() {
    cin >> n; for(int i = 1; i <= n; ++ i) {
        p[i].input();;
        xmax = max(xmax, p[i].x);
        xmin = min(xmin, p[i].x);
        ymax = max(ymax, p[i].y);
        ymin = min(ymin, p[i].y);
    } p[n + 1] = p[1];

    double xl = xmin, xr = xmax, midl, midr;
    while(xr - xl > eps) {
        midl = (xl + xl + xr) / 3.0;
        midr = (xl + xr + xr) / 3.0;
        if(Find(midl) < Find(midr)) xl = midl;
        else xr = midr;
    }
    cout << fixed << setprecision(10) << maxx * 180.0 / pi << endl;
}
```

4.33 三维几何

```
#include <math.h>
#define eps 1e-8
#define zero(x) (((x)>0?(x):-(x))<eps)
struct point3{double x,y,z;};
struct line3{point3 a,b;};
struct plane3{point3 a,b,c;};
//计算 cross product U x V
point3 Cross(point3 u,point3 v){
    point3 ret;
    ret.x=u.y*v.z-v.y*u.z;
    ret.y=u.z*v.x-u.x*v.z;
    ret.z=u.x*v.y-u.y*v.x;
    return ret;
}
//计算 dot product U . V
double Dot(point3 u,point3 v){
    return u.x*v.x+u.y*v.y+u.z*v.z;
}
//矢量差 U - V
point3 subt(point3 u,point3 v){
    point3 ret;
    ret.x=u.x-v.x;
    ret.y=u.y-v.y;
    ret.z=u.z-v.z;
    return ret;
}
//取平面法向量
point3 pvec(plane3 s){
    return Cross(subt(s.a,s.b),subt(s.b,s.c));
}
point3 pvec(point3 s1,point3 s2,point3 s3){
    return Cross(subt(s1,s2),subt(s2,s3));
}
//两点距离,单参数取向量大小
double distance(point3 p1,point3 p2){
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1.z-p2.z)*(p1.z-p2.z));
}
//向量大小
double vlen(point3 p){
    return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
}
//判三点共线
int dots_inline(point3 p1,point3 p2,point3 p3){
    return vlen(Cross(subt(p1,p2),subt(p2,p3)))<eps;
}
//判四点共面
int dots_onplane(point3 a,point3 b,point3 c,point3 d){
    return zero(Dot(pvec(a,b,c),subt(d,a)));
}
//判点是否在线段上,包括端点和共线
int dot_online_in(point3 p,line3 l){
    return zero(vlen(Cross(subt(p,l.a),subt(p,l.b))))&&(l.a.x-p.x)*(l.b.x-p.x)<eps&&
           (l.a.y-p.y)*(l.b.y-p.y)<eps&&(l.a.z-p.z)*(l.b.z-p.z)<eps;
}
int dot_online_in(point3 p,point3 l1,point3 l2){
    return zero(vlen(Cross(subt(p,l1),subt(p,l2))))&&(l1.x-p.x)*(l2.x-p.x)<eps&&
           (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
}
```

```
//判点是否在线段上,不包括端点
int dot_online_ex(point3 p,line3 l){
    return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
            (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
}
int dot_online_ex(point3 p,point3 l1,point3 l2){
    return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))&&
           (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
}
//判点是否在空间三角形上,包括边界,三点共线无意义
int dot_inplane_in(point3 p,plane3 s){
    return zero(vlen(Cross(subt(s.a,s.b),subt(s.a,s.c)))-vlen(Cross(subt(p,s.a),subt(p,s.b)))-
                vlen(Cross(subt(p,s.b),subt(p,s.c)))-vlen(Cross(subt(p,s.c),subt(p,s.a))));
}
int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
    return zero(vlen(Cross(subt(s1,s2),subt(s1,s3)))-vlen(Cross(subt(p,s1),subt(p,s2)))-
                vlen(Cross(subt(p,s2),subt(p,s3)))-vlen(Cross(subt(p,s3),subt(p,s1))));
}
//判点是否在空间三角形上,不包括边界,三点共线无意义
int dot_inplane_ex(point3 p,plane3 s){
    return dot_inplane_in(p,s)&&vlen(Cross(subt(p,s.a),subt(p,s.b)))>eps&&
           vlen(Cross(subt(p,s.b),subt(p,s.c)))>eps&&vlen(Cross(subt(p,s.c),subt(p,s.a)))>eps;
}
int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
    return dot_inplane_in(p,s1,s2,s3)&&vlen(Cross(subt(p,s1),subt(p,s2)))>eps&&
           vlen(Cross(subt(p,s2),subt(p,s3)))>eps&&vlen(Cross(subt(p,s3),subt(p,s1)))>eps;
}
//判两点在线段同侧,点在线段上返回 0,不共面无意义
int same_side(point3 p1,point3 p2,line3 l){
    return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))>eps;
}
int same_side(point3 p1,point3 p2,point3 l1,point3 l2){
    return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))>eps;
}
//判两点在线段异侧,点在线段上返回 0,不共面无意义
int opposite_side(point3 p1,point3 p2,line3 l){
    return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))<-eps;
}
int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2){
    return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))<-eps;
}
//判两点在平面同侧,点在平面上返回 0
int same_side(point3 p1,point3 p2,plane3 s){
    return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))>eps;
}
int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
    return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))>eps;
}
//判两点在平面异侧,点在平面上返回 0
int opposite_side(point3 p1,point3 p2,plane3 s){
    return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))<-eps;
```

```
}
int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
    return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))<-eps;
}
//判两直线平行
int parallel(line3 u,line3 v){
    return vlen(Cross(subt(u.a,u.b),subt(v.a,v.b)))<eps;
}
int parallel(point3 u1,point3 u2,point3 v1,point3 v2){
    return vlen(Cross(subt(u1,u2),subt(v1,v2)))<eps;
}
//判两平面平行
int parallel(plane3 u,plane3 v){
    return vlen(Cross(pvec(u),pvec(v)))<eps;
}
int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
    return vlen(Cross(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;
}
//判直线与平面平行
int parallel(line3 l,plane3 s){
    return zero(Dot(subt(l.a,l.b),pvec(s)));
}
int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    return zero(Dot(subt(l1,l2),pvec(s1,s2,s3)));
}
//判两直线垂直
int perpendicular(line3 u,line3 v){
    return zero(Dot(subt(u.a,u.b),subt(v.a,v.b)));
}
int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2){
    return zero(Dot(subt(u1,u2),subt(v1,v2)));
}
//判两平面垂直
int perpendicular(plane3 u,plane3 v){
    return zero(Dot(pvec(u),pvec(v)));
}
int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
    return zero(Dot(pvec(u1,u2,u3),pvec(v1,v2,v3)));
}
//判直线与平面平行
int perpendicular(line3 l,plane3 s){
    return vlen(Cross(subt(l.a,l.b),pvec(s)))<eps;
}
int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    return vlen(Cross(subt(l1,l2),pvec(s1,s2,s3)))<eps;
}
//判两线段相交,包括端点和部分重合
int intersect_in(line3 u,line3 v){
    if (!dots_onplane(u.a,u.b,v.a,v.b))
        return 0;
    if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
        return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
    return dot_online_in(u.a,v)||dot_online_in(u.b,v)||dot_online_in(v.a,u)||dot_online_in(v.b,u);
}
int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2){
    if (!dots_onplane(u1,u2,v1,v2))
        return 0;
    if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
```

```
        return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
    return
            dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||dot_online_in(v2,u1,u2);
}
//判两线段相交,不包括端点和部分重合
int intersect_ex(line3 u,line3 v){
    return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u);
}
int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2){
    return  dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,u1,u2);
  }
//判线段与空间三角形相交,包括交于边界和(部分)包含
int intersect_in(line3 l,plane3 s){
    return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
           !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
}
int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
           !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
}
//判线段与空间三角形相交,不包括交于边界和(部分)包含
int intersect_ex(line3 l,plane3 s){
    return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
           opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b);
}
int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
           opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
}
//计算两直线交点,注意事先判断直线是否共面和平行!
//线段交点请另外判线段相交(同时还是要判断是否平行!)
point3 intersection(line3 u,line3 v){
    point3 ret=u.a;
    double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
             /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
    ret.x+=(u.b.x-u.a.x)*t;
    ret.y+=(u.b.y-u.a.y)*t;
    ret.z+=(u.b.z-u.a.z)*t;
    return ret;
}
point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2){
    point3 ret=u1;
    double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
             /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x+=(u2.x-u1.x)*t;
    ret.y+=(u2.y-u1.y)*t;
    ret.z+=(u2.z-u1.z)*t;
    return ret;
}
//计算直线与平面交点,注意事先判断是否平行,并保证三点不共线!
//线段和空间三角形交点请另外判断
point3 intersection(line3 l,plane3 s){
    point3 ret=pvec(s);
    double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
             (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
    ret.x=l.a.x+(l.b.x-l.a.x)*t;
    ret.y=l.a.y+(l.b.y-l.a.y)*t;
    ret.z=l.a.z+(l.b.z-l.a.z)*t;
    return ret;
}
point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    point3 ret=pvec(s1,s2,s3);
    double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
             (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
    ret.x=l1.x+(l2.x-l1.x)*t;
    ret.y=l1.y+(l2.y-l1.y)*t;
    ret.z=l1.z+(l2.z-l1.z)*t;
    return ret;
}
//计算两平面交线,注意事先判断是否平行,并保证三点不共线!
line3 intersection(plane3 u,plane3 v){
    line3 ret;
    ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(v.a,v.b,u.a,u.b,u.
            c);
    ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(v.c,v.a,u.a,u.b,u.
            c);
    return ret;
}
line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
    line3 ret;
    ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,u2,u3);
    ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,u2,u3);
    return ret;
}
//点到直线距离
double ptoline(point3 p,line3 l){
    return vlen(Cross(subt(p,l.a),subt(l.b,l.a)))/distance(l.a,l.b);
}
double ptoline(point3 p,point3 l1,point3 l2){
    return vlen(Cross(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
}
//点到平面距离
double ptoplane(point3 p,plane3 s){
    return fabs(Dot(pvec(s),subt(p,s.a)))/vlen(pvec(s));
}
double ptoplane(point3 p,point3 s1,point3 s2,point3 s3){
    return fabs(Dot(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
}
//直线到直线距离
double linetoline(line3 u,line3 v){
    point3 n=Cross(subt(u.a,u.b),subt(v.a,v.b));
    return fabs(Dot(subt(u.a,v.a),n))/vlen(n);
}
double linetoline(point3 u1,point3 u2,point3 v1,point3 v2){
    point3 n=Cross(subt(u1,u2),subt(v1,v2));
    return fabs(Dot(subt(u1,v1),n))/vlen(n);
}
//两直线夹角 cos 值
double angle_cos(line3 u,line3 v){
    return Dot(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
}
```

```
double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2){
    return Dot(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
}
//两平面夹角 cos 值
double angle_cos(plane3 u,plane3 v){
    return Dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
}
double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
    return Dot(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3));
}
//直线平面夹角 sin 值
double angle_sin(line3 l,plane3 s){
    return Dot(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
}
double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
    return Dot(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
}

// 球体相交
double vol_ints(double x1, double y1, double z1, double r1, double x2, double y2, double z2, double r2) {
    double sum = 4.00 / 3.00 * PI * r1 * r1 * r1 + 4.00 / 3.00 * PI * r2 * r2 * r2;
    double ans = 0;
    double dis = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 - z2)); //球心距离
    if (dis >= r1 + r2) //没有交到的地方
    {
        ans = 0;
    } else if (dis + r1 <= r2)//重合
    {
        ans = (4.00 / 3.00) * PI * r1 * r1 * r1;
    } else if (dis + r2 <= r1) {
        ans = (4.00 / 3.00) * PI * r2 * r2 * r2;
    } else  //相交
    {
        double cal = (r1 * r1 + dis * dis - r2 * r2) / (2.00 * dis * r1);
        double h = r1 * (1 - cal);
        ans += (1.00 / 3.00) * PI * (3.00 * r1 - h) * h * h;
        cal = (r2 * r2 + dis * dis - r1 * r1) / (2.00 * dis * r2);
        h = r2 * (1.00 - cal);
        ans += (1.00 / 3.00) * PI * (3.00 * r2 - h) * h * h;
    }
    return ans;
}
```

4.34 多项式 ln$_exp_pow$ 1

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;

struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x * rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

ll mod, inv2;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;

int getLen(int n) ;

void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) ;

void Get_Der(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i - 1] = f[i] * i % mod; g[len - 1] = 0; }

void Get_Int(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i] = f[i - 1] * quick_pow(i, mod - 2) % mod; g[0] = 0; }

void Get_Ln(ll *f, ll *g, int n) ;

void Get_Exp(ll *f, ll *g, int n) ;

void Get_Pow(ll *f, ll *g, int n, ll k) ;

void Get_Sqrt(ll *f, ll *g, int n) {
    static ll a[N];
    Get_Ln(f, a, n);
    for(int i = 0;i < n; i++) a[i] = a[i] * inv2 % mod;
    Get_Exp(a, g, n);
    int len = getLen(n);
    for(int i = n;i < len; i++) g[i] = 0;
    for(int i = 0;i < len; i++) a[i] = 0;
}
```

4.35 二次剩余处理边界不为 1

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 1e5 + 10;


struct Complex {
    double x, y;
    Complex(double a = 0, double b = 0): x(a), y(b) {}
    Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
    Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
    Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x * rhs.y + y * rhs.x); }
    Complex conj() { return Complex(x, -y); }
} w[N];

ll mod, inv2;
int tr[N];
ll F[N], G[N];

ll quick_pow(ll a, ll b) ;

typedef struct{
    ll x, y; // 把求出来的w作为虚部，则为a + bw
}num;

num num_mul(num a, num b, ll w, ll p) {// 复数乘法
    num ans = {0, 0};
    ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
    ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
    return ans;
}

ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
    num ans = {1, 0};
    while(b) {
        if(b & 1)
            ans = num_mul(ans, a, w, p);
        a = num_mul(a, a, w, p);
        b >>= 1;
    }
    return ans.x % p;
}

ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
    return quick_pow(a, (p - 1) >> 1);
}

ll Cipolla(ll n, ll p) {// 输入a和p，是否存在x使得x^2 = a (mod p)，存在二次剩余返回x，存在二次非剩余返回-1     注意：p是奇质数
    n %= p;
    if(n == 0)
        return 0;
    if(p == 2)
        return 1;
    ll a, w;

    while(true) {// 找出a，求出w，随机成功的概率是50%，所以数学期望是2
        a = rand() % p;
        w = ((a * a - n) % p + p) % p;
        if(legendre(w, p) + 1 == p) // 找到w，非二次剩余条件
            break;
    }
    num x = {a, 1};
    return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x,一个解是x，另一个解是p-x，这里的w其实要开方，但是由拉格朗日定理可知虚部为0，所以最终答案就是对x的实部用快速幂求解
}

int getLen(int n) ;

```

```cpp
void FFT(Complex *A, int len) ;

inline void MTT(ll *x, ll *y, ll *z, int len) ;

void Get_Inv(ll *f, ll *g, int n) ;

void Get_Sqrt(ll *f, ll *g, int n) {
    if(n == 1) { ll t = Cipolla(f[0], mod); g[0] = min(mod - t, t); return ; }
    Get_Sqrt(f, g, (n + 1) >> 1);

    int len = getLen(n);
    static ll c[N], invg[N];
    for(int i = 0;i < len; i++) c[i] = i < n ? f[i] : 0;
    Get_Inv(g, invg, n);
    MTT(c, invg, c, len);
    for(int i = 0;i < n; i++) g[i] = inv2 * (c[i] + g[i]) % mod;
    for(int i = n;i < len; i++) g[i] = 0;
    for(int i = 0;i < len; i++) c[i] = invg[i] = 0;
}

int main() {
    inv2 = quick_pow(2, mod - 2);
    int n;
    cin >> n;
    for(int i = 0;i < n; i++) cin >> F[i];
    Get_Sqrt(F, G, n);
    for(int i = 0;i < n; i++) cout << G[i] << " ";
}
```

4.36 x 不连续、暴力插值

```cpp

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const double PI = acos(-1);
const int N = 3e5 + 10;

ll mod;
ll X[N], Y[N];

ll quick_pow(ll a, ll b) ;

ll Lagrange(ll *x, ll *y, int n, int k) {
    ll ans = 0;
    for(int i = 0;i < n; i++) {
        ll s1 = 1, s2 = 1;
        for(int j = 0;j < n; j++) {
            if(i == j) continue;
            s1 = s1 * (k - x[j] + mod) % mod;
            s2 = s2 * (x[i] - x[j] + mod) % mod;
        }
        ans = (ans + 1ll * y[i] * s1 % mod * quick_pow(s2, mod - 2) % mod) % mod;
    }
    return ans;
}

int main() {
    int n, k;
    cin >> n >> k;
    for(int i = 0;i < n; i++) cin >> X[i] >> Y[i];
    cout << Lagrange(X, Y, n, k) << endl;
}
```

4.37 x 连续、前缀优化

```

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;

ll mod;
ll F[N];
ll pre[N], suf[N];
ll fac[N], invf[N];


ll quick_pow(ll a, ll b) ;

void init() {
    fac[0] = 1;
    for(int i = 1;i < N; i++) fac[i] = fac[i - 1] * i % mod;
    invf[N - 1] = quick_pow(fac[N - 1], mod - 2);
    for(int i = N - 1;i >= 1; i--) invf[i - 1] = invf[i] * i % mod;
}

ll Lagrange(ll *f, int k, int n) {
    if(k <= n) return f[k];
    pre[0] = suf[n] = 1;
    for(int i = 1;i <= n; i++) pre[i] = pre[i - 1] * (k - i + 1) % mod;
    for(int i = n;i >= 1; i--) suf[i - 1] = suf[i] * (k - i) % mod;
    ll ans = 0;
    for(int i = 0;i <= n; i++) {
        int opt = (n - i) & 1 ? -1 : 1;
        ans = (ans + 1ll * opt * pre[i] % mod * suf[i] % mod * invf[i] % mod * invf[n - i] % mod * f[i] % mod + mod) % mod;
    }
    return f[k] = ans;
}

int main() {
    init();
    int n, k;
    cin >> n >> k;
    for(int i = 0;i <= n; i++) cin >> F[i];
    cout << Lagrange(F, k, n) << endl;

}
```

4.38 FFT 加速带有通配符字符串匹配

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

```

// p[x] = \sum_{i=0}^{m-1} A[i]^3 * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 - 2 * \sum_{i=0}^{m-1} A[i]^2 * B[x-m+i+1]^2

const int N = 1e6 + 1e5;

ll qpow(ll a, ll b, ll mod) {
    ll ans = 1;
    while(b) {
        if(b & 1) ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans % mod;
}

const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = qpow(len, mod - 2, mod);
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    NTT(a, len, 1), NTT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
    NTT(a, len, -1);
}

ll a1[N], a2[N], a3[N], b1[N], b2[N], b3[N];

void solve() {
    int m, n; cin >> m >> n;
    string s, t; cin >> t >> s;
    for(int i = 0;i < m; i++) {
        if(t[i] == '*') continue ;
        int temp = t[i] - 'a' + 1;
        a1[i] = temp;
        a2[i] = temp * temp;
        a3[i] = temp * temp * temp;
    }
    for(int i = 0;i < n; i++) {
        if(s[i] == '*') continue;
        int temp = s[i] - 'a' + 1;
        b1[i] = temp;
        b2[i] = temp * temp;
        b3[i] = temp * temp * temp;
    }
    reverse(a1, a1 + m);
    reverse(a2, a2 + m);
    reverse(a3, a3 + m);
    mul(a1, b3, n + m);
    mul(a2, b2, n + m);
    mul(a3, b1, n + m);
    vector<int> ans;
    for(int x = m - 1;x < n; x++) {
        ll res = a1[x] + a3[x] - a2[x] * 2;
        if(!res) ans.push_back(x - m + 2);
    }
    cout << ans.size() << endl;
    for(int i = 0;i < ans.size(); i++) cout << ans[i] << (i == ans.size() - 1 ? endl : " ");
}
```

4.39 FFT 加速带有通配符字符串匹配 2

```
void solve() {
    int o; cin >> o; while(o --) {
        int n, m; cin >> n >> m;
        vector<int> ans(n + m), res(n + m);
        string s, t; cin >> s >> t;
        reverse(t.begin(), t.end());
        Polynomial A(s.size() + 10), B(t.size() + 10), C;

        for(int cch = 0; cch <= 9; ++ cch) {
            char now = cch <= 9? char(cch + '0'): '*';
            for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
            for (int i = 0; i < s.size(); ++i) A[i] = s[i] == now;
            for (int i = 0; i < t.size(); ++i) B[i] = t[i] == now;
            C = A * B;
            for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
        }

//        for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
        for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
        for(int i = 0; i < t.size(); ++ i) B[i] = 1;
        C = A * B;
        for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];


//        for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
        for(int i = 0; i < s.size(); ++ i) A[i] = 1;
        for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
        C = A * B;
        for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];

//        for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
        for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
        for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
        C = A * B;
        for(int i = m - 1; i < n; ++ i) ans[i - m + 1] -= C[i];

        for(int i = 0; i <= n - m; ++ i) res[m - ans[i]] ++;
        for(int i = 1; i <= m; ++ i) res[i] += res[i - 1];
        for(int i = 0; i <= m; ++ i) cout << res[i] << '\n';

    }
}
```

4.40 FFT 加速朴素字符串匹配

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 4e5 + 10;

// P[x] = \sum_{i=0}^{m-1} A[i] + \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m-1}A[i] * B[x - m + i + 1]

// reverse(a)

// 当串中的字符集较少时，可以针对每个字符进行FFT，计算每个字符对整个串的贡献

ll qpow(ll a, ll b, ll mod) ;

const ll mod = 998244353;
const ll G = 3;
const ll invG = qpow(G, mod - 2, mod);
int tr[N];

void NTT(ll *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {
        int mid = i / 2;
        ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
        for (int k = 0; k < len; k += i) {
            ll w = 1;
            for (int l = k; l < k + mid; l++) {
                ll t = w * A[l + mid] % mod;
                A[l + mid] = (A[l] - t + mod) % mod;
                A[l] = (A[l] + t) % mod;
                w = w * Wn % mod;
            }
        }
    }
    if (type == -1) {
        ll invn = qpow(len, mod - 2, mod);
        for (int i = 0; i < len; i++)
            A[i] = A[i] * invn % mod;
    }
}

void mul(ll *a, ll *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    NTT(a, len, 1), NTT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
    NTT(a, len, -1);
}

ll a[N], b[N];

void solve() {
    string s, t; cin >> s >> t;
    int n = s.length(), m = t.length();
    for(int i = 0;i < n; i++) a[i] = s[i] - 'a' + 1;
    for(int i = 0;i < m; i++) b[i] = t[i] - 'a' + 1;
    reverse(b, b + m);
    mul(a, b, n + m - 2);
    double P = 0;
    for(int i = 0;i < m; i++) {
        P += (t[i] - 'a' + 1) * (t[i] - 'a' + 1);
    }
    vector<int> f(n + 1);
    for(int i = 1;i < n; i++) {
        f[i] = f[i - 1] + (s[i] - 'a' + 1) * (s[i] - 'a' + 1);
    }
    for(int x = m - 1;x < n; x++) {
        double res;
        if(x == m - 1) res = P + f[x] - a[x] * 2;
        else res = P + f[x] - f[x - m] - a[x] * 2;
        if(!res) cout << x - m + 2 << endl;
    }
}
```

4.41 CDQ$_{F}FT$

```
#include <bits/stdc++.h>
using namespace std;
#define gcd(a,b) __gcd(a,b)
#define lcm(a,b) (1ll * a * b / gcd(a, b))
#define Polynomial vector<int>
#define Inv(x) quick_pow(x, mod - 2)
#define DEBUG(x, y) cout << x << ": " << y << '\n';
using ld = long double;
using ll = long long;
using ull = unsigned long long;
const ll mod = 100003;
const ld pi = acos(-1.0);
struct Complex {
    ld r, i;
    Complex(ld _r = 0, ld _i = 0) : r(_r), i(_i) {}
};

Complex operator + (const Complex &a, const Complex &b) {
    return Complex(a.r + b.r, a.i + b.i);
}
Complex operator - (const Complex &a, const Complex &b) {
    return Complex(a.r - b.r, a.i - b.i);
}
Complex operator * (const Complex &a, const Complex &b) {
    return Complex(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
```

```
}
Complex operator / (const Complex &a, const Complex &b) {
    return Complex((a.r * b.r + a.i * b.i) / (b.r * b.r + b.i * b.i), (a.i * b.r - a.r * b.i) / (b.r * b.r + b.i * b.i));
}

int R[int(1e6 + 10)];
Complex x[int(1e6 + 10)], y[int(1e6 + 10)];

void get_R(int lim) {
    for (int i = 0; i < lim; i++) {
        R[i] = (i & 1) * (lim >> 1) + (R[i >> 1] >> 1);
    }
}

void FFT(Complex *f, int lim, int rev) {
    for (int i = 0; i < lim; i++) {
        if (i < R[i]) swap(f[i], f[R[i]]);
    }
    for (int mid = 1; mid < lim; mid <<= 1) {
        Complex wn = Complex(cos(pi / mid), rev * sin(pi / mid));
        for (int len = mid << 1, cur = 0; cur < lim; cur += len) {
            Complex w = Complex(1, 0);
            for (int k = 0; k < mid; k++, w = w * wn) {
                Complex x = f[cur + k], y = w * f[cur + mid + k];
                f[cur + k] = x + y, f[cur + mid + k] = x - y;
            }
        }
    }
    if (rev == -1) {
        for (int i = 0; i < lim; i++) {
            f[i].r /= lim;
        }
    }
}

int s[int(1e6 + 10)];
vector<int> ans[int(1e6 + 10)];

int quick_pow(int ans, int p, int res = 1) {
    for(; p; p >>= 1, ans = 1ll * ans * ans % mod)
        if(p & 1) res = 1ll * res * ans % mod;
    return res % mod;
}
int fac[int(1e6 + 10)] = {1}, ifac[int(1e6 + 10)] = {1};
void init(int n) {
    for(int i = 1; i <= n; ++ i)
        fac[i] = 1ll * fac[i - 1] * i % mod,
                ifac[i] = Inv(fac[i]);
}
ll C(int n, int m) {
    if(n < 0 || m < 0 || m > n) return 0;
    if(m == 0 || m == n)    return 1;
    return 1ll * fac[n] * ifac[m] % mod * ifac[n - m] % mod;
}
ll F(ll a, ll n, ll k) {
    return (1ll * ans[1][k] - C(n, k) + mod) % mod * Inv(a - 1) % mod;
}
void CDQ_FFT(int rt, int l, int r) {
    if (l == r) {
        ans[rt].push_back(1);
        ans[rt].push_back(s[l]);
        return ;
    }
    int mid = l + r >> 1;
    CDQ_FFT(rt << 1, l, mid);
    CDQ_FFT(rt << 1 | 1, mid + 1, r);
    int len1 = mid - l + 1, len2 = r - mid;
    for (int i = 0; i <= len1; i++) x[i] = Complex(ans[rt << 1][i], 0);
    for (int i = 0; i <= len2; i++) y[i] = Complex(ans[rt << 1 | 1][i], 0);
    int lim = 1; while (lim <= r - l + 1) lim <<= 1;
    get_R(lim);
    FFT(x, lim, 1);
    FFT(y, lim, 1);
    for (int i = 0; i < lim; i++) x[i] = x[i] * y[i];
    FFT(x, lim, -1);
    for (int i = 0; i <= r - l + 1; i++) ans[rt].push_back(ll(x[i].r + 0.5) % mod);
    for (int i = 0; i < lim; i++) x[i] = y[i] = Complex(0, 0);
}

inline void solve() {
    init(int(1e6));
    int n, a, q; cin >> n >> a >> q;
    for(int i = 1; i <= n; ++ i) {
        cin >> s[i]; s[i] = quick_pow(a, s[i]);
    }
    CDQ_FFT(1, 1, n);
    while(q --) {
        int k; cin >> k;
        cout << F(a, n, k) << '\n';
    }
}
```

4.42 FFT

```
#include <bits/stdc++.h>
using namespace std;
const double PI = acos(-1);
const int N = 4e5 + 10;

struct Complex {
    double a, b;
    Complex(double a = 0, double b = 0): a(a), b(b) {}
    Complex operator * (const Complex &rhs) { return Complex(a * rhs.a - b * rhs.b, a * rhs.b + b * rhs.a); }
    Complex operator + (const Complex &rhs) { return Complex(a + rhs.a, b + rhs.b); }
    Complex operator - (const Complex &rhs) { return Complex(a - rhs.a, b - rhs.b); }
};

int tr[N];

void FFT(Complex *A, int len, int type) {
    for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
    for (int i = 2; i <= len; i <<= 1) {//区间长度
        int mid = i / 2;
        Complex Wn(cos(2 * PI / i), type * sin(2 * PI / i));//单位根
        for (int k = 0; k < len; k += i) {//每个子问题的起始点
```

```
            Complex w(1, 0);//omega
            for (int l = k; l < k + mid; l++) {
                Complex t = w * A[l + mid];
                A[l + mid] = A[l] - t;
                A[l] = A[l] + t;
                w = w * Wn;
            }
        }
    }
}

void mul(Complex *a, Complex *b, int n) {
    int len = 1; while (len <= n) len <<= 1;
    for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
    FFT(a, len, 1), FFT(b, len, 1);
    for (int i = 0; i < len; i++) a[i] = a[i] * b[i];
    FFT(a, len, -1);
    for (int i = 0; i < len; i++) a[i].a /= len;
}

Complex a[N], b[N];

void solve() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; i++) scanf("%lf", &a[i].a);
    for (int i = 0; i <= m; i++) scanf("%lf", &b[i].a);

    mul(a, b, n + m);
    for (int i = 0; i <= n + m; i++)
        printf("%d ", (int)(a[i].a + 0.5));
}
```

4.43 FWT

```
inline void OR(ll *f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++j) {
                f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod)) % mod;
            }
        }
    }
}
inline void AND(ll *f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++j) {
                f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
            }
        }
    }
}
inline void XOR(ll *f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++j) {
                f[i + j] += f[i + j + k],
                f[i + j + k] = f[i + j] - f[i + j + k] - f[i + j + k],
                f[i + j] *= x, f[i + j + k] *= x;
                f[i + j] %= mod;
                f[i + j + k] %= mod;
                while (f[i + j] < 0) f[i + j] += mod;
                while (f[i + j + k] < 0) f[i + j + k] += mod;
            }
        }
    }
}
inline void OR(vector<ll> &f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++j) {
                f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod)) % mod;
            }
        }
    }
}
inline void AND(vector<ll> &f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++j) {
                f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
            }
        }
    }
}
inline void XOR(vector<ll> &f, int n, int x = 1) {
    for (int o = 2; o <= n; o <<= 1) {
        for (int i = 0, k = o >> 1; i < n; i += o) {
            for (int j = 0; j < k; ++ j) {
                ll X = f[i + j], Y = f[i + j + k];
                f[i + j] = (X + Y) % mod;
                f[i + j + k] = ((X - Y) % mod + mod) % mod;
                // mod is a prime
                if(x != 1) {
                    f[i + j] = f[i + j] * inv2 % mod;
                    f[i + j + k] = f[i + j + k] * inv2 % mod;
                }
            }
        }
    }
// mod is not a prime, let mod = mod * (1 << m), n = 1 << m;
//          if(x != 1) for(int i = 0; i < n; ++ i) {
//              f[i] /= n;
//          }
}
```

4.44 NTT

```
Polynomial R;
inline int Binary_Rounding(const int &n, int len = 1) { //¶Ô ½Ø       
    £¬Ô NTT±任Ö ±¸¡£
    while (len < n) len <<= 1;
    return len;
```

```
}
//□ ´¦□ □ R□ □ □ 餋□ ±¸±任£¬□ □ ÿ´□ NT□ □ ǰ□ □ □ □ ¼Çµ□ □ ô¯¯□ □ ¡£
inline int Prepare_Transformation(int n) {
    int L = 0, len;
    for (len = 1; len < n; len <<= 1) L++;
    R.clear();
    R.resize(len);
    for (int i = 0; i < len; ++i)
        R[i] = (R[i >> 1] >> 1) | ((i & 1) << (L - 1));
    return len;
}
inline void NTT(Polynomial &a, int f) {
    int n = a.size();
    for (int i = 0; i < n; ++i)
        if (i < R[i])swap(a[i], a[R[i]]);
    for (int i = 1; i < n; i <<= 1)
        for (int j = 0, gn = quick_pow(mod_g, (mod - 1) / (i << 1)); j < n; j += (i << 1))
            for (int k = 0, g = 1, x, y; k < i; ++k, g = 1ll * g * gn % mod)
                x = a[j + k], y = 1ll * g * a[i + j + k] % mod,
                a[j + k] = (x + y) % mod, a[i + j + k] = (x - y + mod) % mod;
    if (f == -1) {
        reverse(a.begin() + 1, a.end());
        int inv = Inv(n);
        for (int i = 0; i < n; ++i) a[i] = 1ll * a[i] * inv % mod;
    }
}
inline Polynomial operator+(const Polynomial &a, const int &b) {
    int sizea = a.size();
    Polynomial ret = a;
    ret.resize(sizea);
    for (int i = 0; i < sizea; ++i)ret[i] = (1ll * a[i] + b + mod) % mod;
    return ret;
}
inline Polynomial operator-(const Polynomial &a, const int &b) {
    int sizea = a.size();
    Polynomial ret = a;
    ret.resize(sizea);
    for (int i = 0; i < sizea; ++i)ret[i] = (1ll * a[i] - b + mod) % mod;
    return ret;
}
inline Polynomial operator*(const Polynomial &a, const int &b) {
    int sizea = a.size();
    Polynomial ret = a;
    ret.resize(sizea);
    for (int i = 0; i < sizea; ++i) ret[i] = (1ll * a[i] * b % mod + mod) % mod;
    return ret;
}
inline Polynomial operator+(const Polynomial &a, const Polynomial &b) {
    int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
    Polynomial ret = a;
    ret.resize(size);
    for (int i = 0; i < sizeb; ++i) ret[i] = (1ll * ret[i] + b[i]) % mod;
    return ret;
}
inline Polynomial operator-(const Polynomial &a, const Polynomial &b) {
    int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
    Polynomial ret = a;
    ret.resize(size);
    for (int i = 0; i < sizeb; ++i) ret[i] = (1ll * ret[i] - b[i] + mod) % mod;
    return ret;
}
inline Polynomial Inverse(const Polynomial &a) {
    Polynomial ret, inv_a;
    ret.resize(1);
    ret[0] = Inv(a[0]);
    int ed = a.size();
    for (int len = 2; len <= ed; len <<= 1) {
        int n = Prepare_Transformation(len << 1);
        inv_a = a;
        inv_a.resize(n);
        ret.resize(n);
        for (int i = len; i < n; ++i) inv_a[i] = 0;
        NTT(inv_a, 1);
        NTT(ret, 1);
        for (int i = 0; i < n; ++i)
            ret[i] = 1ll * (2ll - 1ll * inv_a[i] * ret[i] % mod + mod) % mod * ret[i] %
 mod;
        NTT(ret, -1);
        for (int i = len; i < n; ++i) ret[i] = 0;
    }
    ret.resize(ed);
    return ret;
}


inline Polynomial operator*(const Polynomial &a, const Polynomial &b) {
    Polynomial lsa = a, lsb = b, ret;
    int n = lsa.size(), m = lsb.size();
    n = Prepare_Transformation(n + m);
    lsa.resize(n);
    lsb.resize(n);
    ret.resize(n);
    NTT(lsa, 1);
    NTT(lsb, 1);
    for (int i = 0; i < n; ++i) ret[i] = 1ll * lsa[i] * lsb[i] % mod;
    NTT(ret, -1);
    return ret;
}
inline Polynomial operator/(const Polynomial &a, const Polynomial &b) {
    Polynomial ret = a, ls = b;
    reverse(ret.begin(), ret.end());
    reverse(ls.begin(), ls.end());
    ls.resize(Binary_Rounding(a.size() + b.size()));
    ls = Inverse(ls);
    ls.resize(a.size() + b.size());
    ret = ret * ls;
    ret.resize(a.size() - b.size() + 1);
    reverse(ret.begin(), ret.end());
    return ret;
}
inline Polynomial operator%(const Polynomial &a, const Polynomial &b) {
    Polynomial ret = a / b;
    ret = ret * b;
    ret.resize(a.size() + b.size());
    ret = a - ret;
    ret.resize(a.size() + b.size());
    return ret;
```

```
}
inline Polynomial Derivation(const Polynomial &a) {
    int size = a.size();
    Polynomial ret;
    ret.resize(size);
    for (int i = 1; i < size; ++i) ret[i - 1] = 1ll * i * a[i] % mod;
    ret[size - 1] = 0;
    return ret;
}
inline Polynomial Integral(const Polynomial &a) {
    int size = a.size();
    Polynomial ret;
    ret.resize(size);
    for (int i = 1; i < size; ++i) ret[i] = 1ll * Inv(i) * a[i - 1] % mod;
    ret[0] = 0;
    return ret;
}
inline Polynomial Composition_Inverse(const Polynomial &a) {
    int n = a.size();
    Polynomial ret, Cinv = a, Pow;
    Cinv.resize(n);
    ret.resize(n);
    Pow.resize(n);
    Pow[0] = 1;
    for (int i = 0; i < n - 1; ++i) Cinv[i] = Cinv[i + 1];
    Cinv[n - 1] = 0;
    Cinv = Inverse(Cinv);
    for (int i = 1; i < n; ++i) {
        Pow = Pow * Cinv;
        Pow.resize(n);
        ret[i] = 1ll * Pow[i - 1] * Inv(i) % mod;
    }
    return ret;
}
inline Polynomial Logarithmic(const Polynomial &a) {
    Polynomial ln_a = Derivation(a) * Inverse(a);
    ln_a.resize(a.size());
    return Integral(ln_a);
}
inline Polynomial Exponential(const Polynomial &a, int Constant = 1) {
    Polynomial ret, D;
    int ed = a.size();
    ret.resize(1);
    ret[0] = Constant;
    for (int len = 2; len <= ed; len <<= 1) {
        D = Logarithmic(ret);
        D.resize(len);
        D[0] = (1ll * a[0] + 1ll - D[0] + mod) % mod;
        for (int i = 1; i < len; ++i) D[i] = (1ll * a[i] - D[i] + mod) % mod;
        int n = Prepare_Transformation(len << 1);
        ret.resize(n);
        D.resize(n);
        NTT(ret, 1);
        NTT(D, 1);
        for (int i = 0; i < n; ++i) ret[i] = 1ll * ret[i] * D[i] % mod;
        NTT(ret, -1);
        for (int i = len; i < (len << 1); ++i) ret[i] = D[i] = 0;
    }
    ret.resize(ed);
    return ret;
}
```

4.45 高斯消元

```
#include <bits/stdc++.h>
using namespace std;
const int N=210;
int a[N][N];//增广矩阵
int x[N];//解集
int freeX[N];//自由变元
// equ:方程个数 var:变量个数
int Gauss(int equ, int var){//返回自由变元个数
    /*初始化*/
    for(int i = 0;i <= var; i++){
        x[i] = 0;
        freeX[i] = 0;
    }

    /*转换为阶梯阵*/
    int col = 0;//当前处理的列
    int num = 0;//自由变元的序号
    int k;//当前处理的行
    for(k = 0;k < equ && col < var; k++, col++) {//枚举当前处理的行
        int maxr = k;//当前列绝对值最大的行
        for(int i = k + 1;i < equ; i++){//寻找当前列绝对值最大的行
            if(a[i][col] > a[maxr][col]){
                maxr = i;
                swap(a[k], a[maxr]);//与第k行交换
                break;
            }
        }
        if(a[k][col] == 0){//col列第k行以下全是0，处理当前行的下一列
            freeX[num++] = col;//记录自由变元
            k--;
            continue;
        }

        for(int i = k + 1;i < equ; i++){
            if(a[i][col] != 0){
                for(int j = col;j < var + 1; j++){//对于下面出现该列中有1的行，需要把1消掉
                    a[i][j] ^= a[k][j];
                }
            }
        }
    }

    /*求解*/
    //无解：化简的增广阵中存在(0,0,...,a)这样的行，且a!=0
    for(int i = k;i < equ; i++)
        if(a[i][col] != 0)
            return -1;

    //无穷解：在var*(var+1)的增广阵中出现(0,0,...,0)这样的行
    if(k < var)//返回自由变元数
        return var - k;//自由变元有var-k个

    //唯一解：在var*(var+1)的增广阵中形成严格的上三角阵
    for(int i = var - 1;i >= 0; i--){//计算解集
        x[i] = a[i][var];
        for(int j = i + 1;j < var; j++)
            x[i] ^= (a[i][j] && x[j]);
    }
    return 0;
}
```

4.46 高斯消元 2

```
#include<bits/stdc++.h>
using namespace std;
#define MAX_SIZE 1048
int Matrix[MAX_SIZE][MAX_SIZE];
int Free_x[MAX_SIZE]; //□ □ □ ɱ□ □
int X_Ans[MAX_SIZE]; //½⼯
int Free_num=0;  //□ □ □ ɱ□ □ □ □
int gcd(int a,int b) { return b==0? a : gcd(b,a%b); }
int lcm(int a,int b) { return a/gcd(a,b)*b; }

int Guass(int Row,int Column) {//□ □ □ ¾□ □ □ □ □ У¬□ □
    int row=0,col=0,max_r;
    for(row=0;row<Row&&col<Column;row++,col++) {
        max_r=row;
        for(int i=row+1;i<Row;i++)    //□ ҳ□ ǰ□ e□ □ □ □ □
            if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
                max_r=i;
        if(Matrix[max_r][col]==0) {  //□ □ □ □ □ 0£¬µ¢□ □ □ □ □ □ □ □ £¬¾□ ¼
            row--;
            Free_x[++Free_num]=col+1;
            continue;
        }
        if(max_r!=row)   //½«□ □ □ □ »»µ½µ±ǰ□ □
            for(int i=col;i<Column+1;i++)
                swap(Matrix[row][i],Matrix[max_r][i]);
        for(int i=row+1;i<Row;i++) { //□ □ □
            if(Matrix[i][col]!=0) {
                int LCM=lcm(abs(Matrix[i][col]),abs(Matrix[row][col]));
                int ta=LCM/abs(Matrix[i][col]);
                int tb=LCM/abs(Matrix[row][col]);
                if(Matrix[i][col]*Matrix[row][col]<0)//□ □ □ □ ɼ□ □ □
                    tb=-tb;
                for(int j=col;j<Column+1;j++)
                    Matrix[i][j]=Matrix[i][j]*ta-Matrix[row][j]*tb;
            }
        }
    }
    //row□ □ ³□ ʱ□ ˀ¾□ □ □ □ □ □ □ □ □ □

    for(int i=row;i<Row;i++)  //□ □ □
        if(Matrix[i][Column]!=0)
            return -1;

    if(row<Column)   //□ □ □ □ □ □ ·µ»□ □ □ □ ɱ□ □ □ □
        return Column-row;


    for(int i=Column-1;i>=0;i--) { //Ψһ½□
```

```
        int temp=Matrix[i][Column];
        for(int j=i+1;j<Column;j++)
            if(Matrix[i][j]!=0)
                temp-=Matrix[i][j]*X_Ans[j];
        X_Ans[i]=temp/Matrix[i][i];
    }
    return 0;
}
```

4.47 高斯消元异或

```
#include<bits/stdc++.h>
using namespace std;
#define MAX_SIZE 350
#define ll long long
ll Matrix[MAX_SIZE][MAX_SIZE];
ll Free_x[MAX_SIZE];   //□ □ □ ŋ□ □
ll X_Ans[MAX_SIZE];  //½⊥
ll Free_num=0;  //□ □ □ ŋ□ □ □ □

ll Guass(ll Row,ll Column) { //□ □ □ ¾□ □ □ □ □ κ□ □ □
    ll row=0,col=0,max_r;
    for(row=0;row<Row&&col<Column;row++,col++) {
        max_r=row;
        for(ll i=row+1;i<Row;i++)   //□ χ□ ǰ□ □ □ □ □ □
            if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
                max_r=i;
        if(Matrix[max_r][col]==0) { //¼□ ¼□ □ □ ŋ□ □
            row--;
            Free_x[Free_num++]=col+1;
            continue;
        }
        if(max_r!=row)  //½»»»
            for(ll i=col;i<Column+1;i++)
                swap(Matrix[row][i],Matrix[max_r][i]);
        for(ll i=row+1;i<Row;i++) {  //□ □ □
            if(Matrix[i][col]!=0) {
                for(ll j=col;j<Column+1;j++)
                    Matrix[i][j]^=Matrix[row][j];
            }
        }
    }
    for(ll i=row;i<Row;i++)   //□ □ □
        if(Matrix[i][Column]!=0)
            return -1;

    if(row<Column)   //□ □ □ □ □ □
        return Column-row;

    //Ψh½□
    for(ll i=Column-1;i>=0;i--) {
        X_Ans[i]=Matrix[i][Column];
        for(ll j=i+1;j<Column;j++)
           X_Ans[i]^=(Matrix[i][j]&&X_Ans[j]);
    }
    return 0;
}
```

4.48 矩阵快速幂

```
/***¾□ □ □ □ □ □ □ □ ***/

struct Matrix {
    static const int M = 2;
    double mx[M][M];

    Matrix() { memset(mx, 0, sizeof mx); }

    void Out() {
        for (auto &i : mx) {
            for (auto j : i)
                cout << j << ' ';
            cout << endl;
        }
    }

    void Tranfer_E() {
        memset(mx, 0, sizeof mx);
        for (int i = 0; i < M; ++i) mx[i][i] = 1;
    }

    Matrix operator*(const struct Matrix a) const {
        Matrix x;
        memset(x.mx, 0, sizeof x.mx);
        for (int i = 0; i < M; ++i)
            for (int j = 0; j < M; ++j)
                for (int k = 0; k < M; ++k)
                    x.mx[i][j] = (x.mx[i][j] + mx[i][k] * a.mx[k][j]);
        return x;
    }
};
Matrix mat_pow(Matrix mx, int p) {
    Matrix res;
    res.Tranfer_E();
    for (; p; p >>= 1, mx = mx * mx)
        if (p & 1) res = res * mx;
    return res;
}
```

4.49 矩阵求逆

```
/***¾□ □ □ □ □ □ □ ***/
void Gauss_jordan() {
    /***** □ eĽ»»»&¼□ □ □ □ □  *****/
    for(int i = 0, r; i < n; ++ i) {    //□ □ □ ¦□ □ □ □ i□ □
        r = i;
        for(int j = i + 1; j < n; ++j)
            if(fabs(a[j][i]) > fabs(a[r][i])) r = j;
        if(fabs(a[r][i]) < eps) {
            puts("No Solution");
            return;
        }
        if(i!=r) swap(a[i],a[r]);

        for(int k = 0; k < n; ++ k) {
            //ÿh□ ж¼´¦□ □
            if(k==i) continue;
            double p=a[k][i]/a[i][i];
            for(re int j=i; j<=n; ++j) a[k][j]-=p*a[i][j];
        }
    }

    //□ □ □ □ □ □ □ □ □ ¢□ ¶□ Ó□ □ □ ,´□ Ç³□ □ □ □ □ □
    for(int i = 0; i < n; ++i) printf("%.2lf\n",a[i][n+1]/a[i][i]);
}
```

4.50 线性基

```
// 每个异或值都相同的个数都为2^n-r,所以不同的异或值有2^r个.
#include <bits/stdc++.h>
using namespace  std;

typedef long long ll;
const int maxl = 60;
ll quick_pow(ll a, ll b) ;
struct LinearBasis {
    ll a[maxl + 10];
    int n, size; // 每个相同异或值有2^{n-size}个
    vector<ll> v;

    LinearBasis() {
        memset(a, 0, sizeof(a));
        size = n = 0;
        v.clear();
    }

    void insert(ll t) {
        n++;
        for (int i = maxl; i >= 0; --i) {
            if (!(t >> i & 1)) continue;
            if (a[i]) t ^= a[i];
            else {
                ++size;
                for (int j = i - 1; j >= 0; j--) if (t >> j & 1) t ^= a[j];
                for (int j = i + 1; j <= maxl; ++j) if (a[j] >> i & 1) a[j] ^= t;
                a[i] = t;
                return;
            }
        }
    }

    void basis() {
        for (int i = 0; i <= maxl; ++i) if (a[i]) v.push_back(i);
    }

    // 查询能否xor出x这个数
    bool find(ll x) {
        for(int i = maxl;i >= 0; i--) {
            if(x >> i & 1) {
                if(!a[i]) return 0;
                x ^= a[i];
            }
        }
        return 1;
    }

    // 查询异或最大值
    ll askmax() {
        ll ans = 0;
        for(int i = maxl;i >= 0; i--) ans = max(ans, ans ^ a[i]);
        return ans;
    }

    // 查询异或最小值
    ll askmin() {
        for(int i = 0;i <= maxl; i++) if(a[i]) return a[i];
        return 0;
    }

    // 查询异或第k大
    ll askmaxk(ll x) {

    }

    // 查询异或第k小
    ll askmink(ll x) {
        if(v.size() != n) x--;
        if(!x) return 0;
        if(x >= (1ll << v.size())) return -1;
        ll ans = 0;
        for(int i = 0;i < v.size(); i++) {
            if(x >> i & 1) ans ^= a[v[i]];
        }
        return ans;
    }

    ll rank(ll x) {
        ll ret = 0;
        for (int i = 0; i < v.size(); ++i) if (x >> v[i] & 1) ret += 1LL << i;
        return ret;
    }
};


void solve() {
    int n, x, q;
    scanf("%d", &n);
    LinearBasis lb;
    for (int i = 0; i < n; ++i) scanf("%d", &x), lb.insert(x);
    lb.basis();
    scanf("%d", &q);
    ll num = quick_pow(2, n - lb.size);
    printf("%lld\n", (lb.rank(q) * num + 1));
}
```

4.51 线性基

```
struct LinerBase {
    static const int MAX_BIT = 61;
    bool isZero; ll num[N], tmp[N];
    //判断线性基中是否存在0
    LinerBase() {
        memset(num, 0, sizeof num);
        memset(tmp, 0, sizeof tmp);
    }
    void insert(ll x) {
        for (int i = MAX_BIT; i >= 0; i--) {
            if (!(x & (1ll << i))) continue;
            if (num[i]) {
                x ^= num[i];
            } else {
                num[i] = x;
                return;
            }
        }
        isZero = true;
    }
    bool find(ll x) {
        for (int i = MAX_BIT; i >= 0; i--) {
            if (x & (1ll << i)) {
                if (!num[i]) return false;
                x ^= num[i];
            }
        }
        return true;
    }
    int countNumber() {
        int cnt = 0;
        for (int i = 0; i <= MAX_BIT; i++) if (num[i] & (1ll << i)) cnt++;
        return cnt;
    }
    void rebuild() {
        int cnt = 0;
        for (int i = MAX_BIT; i >= 0; i--)
            for (int j = i-1; j >= 0; j--)
                if (num[i] & (1ll << j)) num[i] ^= num[j];

        for (int i = 0; i <= MAX_BIT; i++) {
            if (num[i]) tmp[cnt++] = num[i];
        }
    }
    ll qryKth(ll k) {
        if (isZero) k--;
        if (!k) return 0;
        int cnt = countNumber(); ll ans = 0;
        if (k >= (1ll << cnt)) return -1;
        for (int i = 0; i < cnt; i++) {
            if (k & (1ll << i)) ans ^= tmp[i];
        }
        return ans;
    }
}lb;
```

4.52 康托展开

```
#include  <iostream>
#include <vector>
#include <algorithm>

using namespace std;
```

```cpp

typedef long long ll;
const int mod = 1e9 + 7;
const int N = 1e5 + 10;

ll fac[N];
int a[N]; // 排列，康托展开求解
int n;
ll x; // 逆康托展开求解

void Get_F() {
    fac[0] = 1;
    for(int i = 1;i < N; i++)
        fac[i] = fac[i - 1] * i % mod;
}

ll CanTor() {
    ll ans = 0;
    for(int i = 1;i <= n; i++) {
        ll smaller = 0;
        for(int j = i + 1;j <= n; j++) {
            if(a[j] < a[i])
                smaller++;
        }
        ans = (ans + fac[n - i] * smaller % mod) % mod;
    }
    return ans + 1;
}

void DeCantor() {
    vector<int> v; // 存放当前可选数
    vector<int> a; // 所求的排列组合序
    for(int i = 1;i <= n; i++) {
        v.push_back(i);
    }
    for(int i = n;i >= 1; i--) {
        int r = x % fac[i - 1];
        int t = x / fac[i - 1];
        x = r;
        sort(v.begin(), v.end());
        a.push_back(v[t]);
        v.erase(v.begin() + t);
    }
    for(int i = 0;i < a.size(); i++)
        cout << a[i] << " ";
    cout << endl;
}

// 线段树优化

const int N = 1000010;

ll fac[N];
int a[N]; // 排列，康托展开求解
int n;

struct SegmentTree {
    int ls, rs;
    int sum;
}t[N << 2];

int cnt, root;

void push_up(int u) {
    t[u].sum = (t[lc].sum + t[rc].sum) % mod;
}

void build(int &u, int l, int r) {
    if(!u) u = ++cnt;
    if(l == r) {
        t[u].sum = 1;
        return ;
    }
    build(lc, l, m);
    build(rc, m + 1, r);
    push_up(u);
}

void update(int &u, int l, int r, int k) {
    if(!u) u = ++cnt;
    if(l == r) {
        t[u].sum = 0;
        return ;
    }
    if(k <= m) update(lc, l, m, k);
    else update(rc, m + 1, r, k);
    push_up(u);
}

ll query(int u, int l, int r, int ql, int qr) {
    if(ql > qr) return 0;
    if(ql == l && qr == r) {
        return t[u].sum;
    }
    if(qr <= m) return query(lc, l, m, ql, qr) % mod;
    else if(ql > m) return query(rc, m + 1, r, ql, qr) % mod;
    else return (query(lc, l, m, ql, m) + query(rc, m + 1, r, m + 1, qr)) % mod;
}

void Get_F() {
    fac[0] = 1;
    for(int i = 1;i < N; i++)
        fac[i] = fac[i - 1] * i % mod;
}

void solve()
{
    Get_F();
    cin >> n;
    build(root, 1, n);
    ll ans = 0;
    for(int i = 1;i <= n; i++) {
        cin >> a[i];
        update(root, 1, n, a[i]);
        ans = (ans + query(root, 1, n, 1, a[i] - 1) * fac[n - i]) % mod;
    }
    cout << (ans + 1) % mod << endl;
}
```

4.53 模数非质数的组合

```cpp
// 模数非质数情况下的组合问题
// one way, use CRT merge ans
// https://ac.nowcoder.com/discuss/655940?type=101&order=0&pos=2&page=1&channel=-1&source_id=discuss_tag_nctrack
// another way
// https://ac.nowcoder.com/acm/contest/view-submission?submissionId=47754622

#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
const int N = 1e6 + 10;

ll qpow(ll a, ll b, ll mod) {
    ll res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    ll res = exgcd(b, a % b, x, y);
    ll t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

ll inv(ll a, ll b) {
    ll x = 0, y = 0;
    exgcd(a, b, x, y);
    return x = (x % b + b) % b;
}

//r[]为余数，m为模数，其中模数互质
//M = pi(mi), Mi = M / mi, invMi = Mi % mi
//ni满足是除了mi之外的倍数，且模mi为ri
//利用逆元性质，即ri * Mi * invMi = ri (mod mi)
//res = (sigma(ri * Mi * invMi)) % M

ll china(ll r[], ll m[], int n) {
    ll M = 1, res = 0;
    for (int i = 1; i <= n; i++) M *= m[i];
    for (int i = 1; i <= n; i++) {
        ll Mi = M / m[i], invMi = inv(Mi, m[i]);
        res = (res + r[i] * Mi % M * invMi % M) % M;
        //res = (res + mul(mul(r[i], Mi, M), invMi, M)) % M;按位乘
    }
    return (res % M + M) % M;
}
```

```

int f[N], g[N], F[N], G[N], invF[N];

int calc(int n, int p, int k) {
    ll mod = qpow(p, k, LONG_LONG_MAX);
    F[0] = 1, G[0] = 0;
    for (int i = 1; i <= n; i++) {
        g[i] = 0, f[i] = i;
        while (f[i] % p == 0) f[i] /= p, g[i]++;
        F[i] = 1ll * F[i - 1] * f[i] % mod;
        G[i] = G[i - 1] + g[i];
    }
    invF[n] = inv(F[n], mod);
    for (int i = n; i >= 1; i--) invF[i - 1] = 1ll * invF[i] * f[i] % mod;
    int ans = 0;
    for (int i = 0; i <= n / 2; i++) {
        int t = 1ll * F[n] * invF[n - 2 * i] % mod * invF[i] % mod * invF[i] % mod *
                qpow(p, G[n] - G[n - 2 * i] - 2 * G[i], LONG_LONG_MAX) % mod;
        ans = (ans + 1ll * t) % mod;
    }
    return ans;
}

ll r[20], m[20];

int main() {
#ifdef ACM_LOCAL
    freopen("input.in", "r", stdin);
    freopen("output.out", "w", stdout);
#endif
    int n, p;
    scanf("%d%d", &n, &p);
    int num = 0;
    for (int i = 2; i * i <= p; i++)
        if (p % i == 0) {
            int k = 0;
            m[++num] = 1;
            while (p % i == 0) p /= i, k++, m[num] *= i;
            r[num] = calc(n, i, k);
        }
    if (p > 1) {
        m[++num] = p;
        r[num] = calc(n, p, 1);
    }
    printf("%lld\n", china(r, m, num));

    return 0;
}
```

4.54 普通型母函数

```
// 普通型母函数: (1+x^1+x^2+...) (1+x^2+x^4)(1+x^3+x^6..)(...)(...)... 类似整数拆分

// a_n=1,1,1,1... = \frac{1}{1-x}
// a_n=1,0,1,0... = \frac{1}{1-x^2}
// a_n=1,2,3,4... = \frac{1}{(1-x)^2}
// a_n=C(m,n)     = (1+x)^m
// a_n=C(m+n,n)   = \frac{1}{(1-x)^{m+1}}
```

```cpp

#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

// 求解硬币等普通问题

const int N = 1e5 + 10;

int a[N]; // 权重为i的组合数，a[P]为答案
int b[N]; // 辅助数组
int P; // 需要被分解的数
int k; // 物品个数
int v[N]; // 每个物品的权重
int n1[N]; // 对于每种物品起始的因子（所需要的每个物品最小个数），最小为0
int n2[N]; // 对于每种物品最终的因子（所需要的每个物品最大个数），最大为INF

// 模板一(标准)

void Calc1() {
    memset(a, 0, sizeof(a));
    a[0] = 1;

    for(int i = 1;i <= k; i++) { // 枚举每个物品因子
        memset(b, 0, sizeof(b));
        for(int j = n1[i];j <= n2[i] && j * v[i] <= P; j++) { // 每个物品从最小因子到最大因子循环,如果n2是无穷的，则j<=n2[i]可以删去
            for(int m = 0;m + j * v[i] <= P; m++) { // 循环a的每个项
                b[m + j * v[i]] += a[m]; // 把结果加到对应项里，有点dp的味道
            }
        }
        memcpy(a, b, sizeof(b));
    }
}

// 模板二（数据量大的时候可以用，快速）

void Calc2() {
    memset(a, 0, sizeof(a));
    a[0] = 1;
    int last = 0;
    for(int i = 1; i <= k; i++) {
        int last2 = min(last + n2[i] * v[i], P);//计算下一次的last
        memset(b, 0, sizeof(int) * (last2 + 1));//只清空b[0..last2]
        for(int j = n1[i]; j <= n2[i] && j * v[i] <= last2; j++) //last2
            for(int m = 0; m <= last && m + j * v[i] <= last2; m++) //一个是last，一个是last2
                b[m + j * v[i]] += a[m];
        memcpy(a, b, sizeof(int) * (last2 + 1));//b赋值给a，只赋值0..last2
        last = last2;//更新last
    }
}
```

4.55 生成函数

```cpp

// O(n*sqrt(n))
```

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 1e5 + 5;
const ll mod = 1e9 + 7;
int f1[270], f2[270];

ll ans[N];

void init() {
    for (int i = 1; ; i++) {
        f1[i] = (3 * i * i - i) >> 1;
        if (f1[i] > 100000) break;
        f2[i] = (3 * i * i + i) >> 1;
        if (f2[i] > 100000) break;
    }
    ans[0] = 1;
    for (int i = 1; i <= 100000; i++) {
        for (int j = 1; ; j++) {
            if (f1[j] <= i) ans[i] += j & 1 ? ans[i-f1[j]] : -ans[i-f1[j]];
            else break;
            if (f2[j] <= i) ans[i] += j & 1 ? ans[i-f2[j]] : -ans[i-f2[j]];
            else break;
        }
        ans[i] = (ans[i] % mod + mod) % mod;
    }
}
```

4.56 斯特林数

```
/***「▯ ▯ ▯ ▯ ▯ ▯ ***/
1
1 1
2 3 1
6 11 6 1
24 50 35 10 1
120 274 225 85 15 1
720 1764 1624 735 175 21 1
namespace STIRLING {
    typedef long long ll;
    const int N = 21;
    const int mod = 1e9 + 7;
    ll Stirling1[N][N], fac[N] = {1};
    void get_stirling1() {
        for(int i = 1; i < N; i++)
            fac[i] = 1ll * fac[i - 1] * i % mod;
        Stirling1[0][0] = 0;
        Stirling1[1][1] = 1;
        for(int i = 2; i < N; i++) {
            for(int j = 1; j <= i; j++) {
                Stirling1[i][j] = (Stirling1[i - 1][j - 1] + (i - 1) * Stirling1[i - 1][j] % mod) % mod;
            }
        }
    }
1
1 1
1 3 1
1 7 6 1
1 15 25 10 1
1 31 90 65 15 1
1 63 301 350 140 21 1
1 127 966 1701 1050 266 28 1
    ll Stirling2[N][N];
    void get_stirling2() {
        Stirling2[0][0] = 0;
        Stirling2[1][1] = 1;
        for(int i = 2; i < N; i++) {
            for(int j = 1; j <= i; j++) {
                Stirling2[i][j] = (Stirling2[i - 1][j - 1] + j * Stirling2[i - 1][j] % mod) % mod;
            }
        }
    }
}
```

4.57 五边形数

```
/***□ □ □ □ □ □ □ ***/
namespace WUBIANXING {
/*1,5,12,22,35..n^2+n(n-1)/2*/
    const int N = 2e5 + 10;
    const int mod = 1e5 + 10;
    int f[N], p[N];
    void init() {
        for (int i = 1; i < N; i++) {
            f[i + i - 1] = (1ll * i * (i + i + i - 1) >> 1) % mod;
            f[i + i] = (1ll * i * (i + i + i + 1) >> 1) % mod;
        }
        for (int i = p[0] = 1; i < N; i++) {
            p[i] = 0;
            for (int j = 1, k = -1; i >= f[j]; j++) {
                if (j & 1) k *= -1;
                p[i] = (p[i] + 1ll * k * p[i - f[j]] % mod) % mod;
            }
            p[i] = (p[i] + mod) % mod;
        }
    }
}
```

4.58 指数型母函数

```

// 需要借助e^x的泰勒展开，一般求解多重排列数，即有 种物品，已知每种物品的数量为 k1,k2,...,kn 个，求从中选出m件物品的排列数。

// 对n个元素全排列，方案数为n!/(n1!n2!...nk!)，对n个中的r个元素进行全排列，这里就用到了指数型母函数，即G(x)=(1+x/1!+x^2/2!+...+x^k1/k1!)(1+x/1!+x^2/2!+...+x^k2/k2!)...(1+x/1!+x^2/2!+...+x^kn/kn!)

// 化简得G(x)=a0 + a1*x+a2*x^2/2!+...+ap*x^p/p!   (p = k1+k2+k3+...) ai为选出i个物品的排列方案数
```

```cpp

//  若题目有规定条件，比如需要物品i出现非0的偶数次，即原式为(x^2/2!+x^4/4!+...+x^ki/ki!)

#include <bits/stdc++.h>
using namespace std;

typedef long double ld;

double num[15]; // 每种物品的数量，第i个物品有num[i]个

double a[15], b[15];

double f[120]; // 阶乘

void fac()
{
    f[0] = 1;
    for(int i = 1;i <= 105; i++)
        f[i] = f[i - 1] * i;
}

void Calc() {
    int n, m;
    cin >> n >> m;
    for(int i = 1;i <= n; i++)
        cin >> num[i];

    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));

    for(int i = 0;i <= num[1]; i++) {
        a[i] = 1.0 / f[i];
    }

    for(int i = 2;i <= n; i++) {
        for(int j = 0;j <= m; j++) {
            for(int k = 0;k <= num[i] && j + k <= m; k++) {
                b[j + k] += a[j] / f[k];
            }
        }

        for(int j = 0;j <= m; j++) {
            a[j] = b[j];
            b[j] = 0;
        }
    }

    cout << a[m] * f[m] << endl;
}
```

4.59 组合数学基础

```cpp
/***□ □ □ □ □ Ạ»□ ´¡***/
namespace COMB {
    typedef long long ll;
    const int mod = 1e9 + 7;
    const int N = 5e5 + 10;
    int fac[N];
    void get_fac(int n, int i = 1) {
        for (fac[0] = 1; i <= n; i++) fac[i] = 1ll * fac[i - 1] * i % mod;
    }
    ll quick_pow(ll ans, ll p, ll res = 1) {
        ans %= mod, p %= mod - 1;
        for (; p; p >>= 1, ans = 1ll * ans * ans % mod) {
            if (p & 1) res = 1ll * res * ans % mod;
        }
        return res % mod;
    }
    ll inv(ll ans) {
        return quick_pow(ans, mod - 2);
    }
    ll C(ll n, ll m) {
        if (m == 0 || n == m) return 1;
        if (n < m || n < 0 || m < 0) return 0;
        return 1ll * fac[n] % mod * inv(1ll * fac[m] * fac[n - m] % mod) % mod;
    }
    ll Lucas(ll n, ll m) {
        if (m == 0) return 1;
        return 1ll * (C(n % mod, m % mod) * Lucas(n / mod, m / mod)) % mod;
    }
}
```

4.60 组合数

```
ll f[N];
ll ksm(ll a, ll b) {
    ll res = 1, base = a;
    while (b) {
        if (b & 1) res = res * base % MOD;
        base = base * base % MOD;
        b >>= 1;
    }
    return res;
}
void init() {
    f[0] = 1;
    for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % MOD;
}
ll C(ll a, ll b) {
    if (a < 0 || b < 0 || b > a) return 0;
    return f[a] * ksm(f[a-b], MOD-2) % MOD * ksm(f[b], MOD-2) % MOD;
}
---------------------------------------------------------------//卢卡斯定理
ll f[N];
ll ksm(ll a, ll b, ll mm) {
    ll res = 1, base = a;
    while (b) {
        if (b & 1) res = res * base % mm;
        base = base * base % mm;
        b >>= 1;
    }
    return res;
}
void init(ll mm) {
    f[0] = 1;
    for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % mm;
}
ll C(ll a, ll b, ll mm) {
    if (a < 0 || b < 0 || b > a) return 0;
    return f[a] * ksm(f[a-b], mm-2, mm) % mm * ksm(f[b], mm-2, mm) % mm;
}

ll lucas(ll a,ll b,ll mm) {
    if(b == 0) return 1;
    return C(a % mm,b % mm,mm) * lucas(a / mm,b / mm,mm) % mm;
}
```

4.61 卡特兰数

```
namespace KART {
/*1,1,2,5,14,42,132,429,1430,4862,16796...*/
/*h(n) = h(n-1)(4n+2)/(n+1) = C(2n,n)/(n+1) = C(2n,n)-C(2n,n-1) (n=0,1,2,...)*/
    typedef long long ll;
    const int N = 1e5 + 10;
    const int mod = 1e9 + 7;
    int cart[N], inv[N];
    void get_inv() {
        inv[0] = inv[1] = 1;
        for(int i = 2; i < N; ++i)
            inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
    }
    void get_cart() {
        cart[0] = 1;
        for(int i = 1; i < N; ++ i)
            cart[i] = 1ll * cart[i - 1] * ((4 * i % mod - 2 + mod) % mod) % mod * inv[i + 1] % mod;
    }
}
```

5 随机化算法

```
int n, m, r;
circle c[20];
Point p[1010], ansp;
double ans, now, nxt;

double Rand() { return (double)rand() / RAND_MAX; }
double calc(Point P) {
    double minn = r;
    rep(i, 1, n) minn = min(minn, P.distance(c[i].p) - c[i].r);
    double res = 0;
    rep(i, 1, m) if(P.distance(p[i]) <= minn) res ++;
    return res;
}

void SA() {
    double x = ansp.x, y = ansp.y;
    double t = 3722;
    while(t > 9e-16) {
        double X = x + ((rand() * 2) - RAND_MAX) * t;
        double Y = y + ((rand() * 2) - RAND_MAX) * t;
        now = calc(Point(X, Y));
        double Delta = now - ans;
        if(Delta > 0) {
            ansp = Point(X, Y);
            x = X, y = Y;
            ans=now;
        } else if(exp(-Delta/t) * RAND_MAX < rand()) x = X, y = Y;
        t *= 0.997577;
    }
}
inline void solve() {
    srand(time(0));
    cin >> n >> m >> r;
    rep(i, 1, n) c[i].input();
    rep(i, 1, m) {
        p[i].input();
        ansp = ansp + p[i];
    }
    ansp = ansp / m;
    ans = calc(ansp);
    while ((double)clock()/CLOCKS_PER_SEC < 0.5) SA();
    cout << ans << '\n';
}
```

5.1 SA2

```
circle c[4];
Point p;

double dx[4]={0, 1, 0, -1},
       dy[4]={1, 0, -1, 0};

double calc(Point a) {
    double dif = 0, d[3];
    for(int i = 0; i < 3; ++ i)
        d[i] = a.distance(c[i].p) / c[i].r;
    for(int i = 0; i < 3; ++ i)
        for(int j = i + 1; j < 3; ++ j)
            dif += (d[i] - d[j]) * (d[i] - d[j]);
    return dif;
}
void solve() {
    double dis = 1.0, now, t;
    for(int i = 0; i < 3; ++ i) {
        c[i].input();
        p = p + c[i].p;
    }
    p = p / 3.0;
    while(dis > eps) {
        now = calc(p);
        int best = -1;
        for(int i = 0; i < 4; ++ i) {
            t = calc(p + Point(dis * dx[i], dis * dy[i]));
            if(now > t) {
                now = t;
                best = i;
            }
        }
        if(best == -1) dis = 0.7 * dis;
        else p = p + Point(dis * dx[best], dis * dy[best]);

    }
    if(calc(p) < eps) p.output();
}
```

6 图论

```
/*

        x=w/l -> w-l*x =0
        f(x) = w-l*x;
        将边权更改为w-l*x来求生成树

       因为f(x)是个单调递减函数,随着x的增大而减少
       对于任意一个生成树如果
        f(x)>0        则   l需要增大
        f(x)<0        否则 l需要减小

       若要满足f(x)==0恒成立
       1.若要x取最大值，则不能存在任意一个生成树f(x)>0,否则x还能继续增大,即任意生成树f(x)<=0
         若存在一个生成树f(x)>0，则那个生成树的比率一定大于当前x,w/l > x -> w-l*x > 0
       2.若要x取最小值，则不能存在任意一个生成树f(x)<0,否则x还能继续减小,即任意生成树f(x)>=0
         若存在一个生成树f(x)<0，则那个生成树的比率一定小于当前x,w/l < x -> w-l*x < 0

       若要满足f(x)>0恒成立，则最小生成树>0
       若要满足f(x)<0恒成立，则最大生成树<0

        此题目求解最小的x值，也就是检查是否所有的生成树f(x)>=0，即最小生成树>=0

        如果最小生成树大于0,所有的生成树都满足f(x)>0,尝试增加x得到f(x)=0
        否则，有生成树不满足这个条件，那么x一定要减少来使所有f(x)>=0

*/
```

6.1 DAG 图上建支配树

```
支配树概念:比如说我们要到a节点必须经过b节点，那么b节点就是a的支配点，在一棵支配树下，所有节点都是子树节点的支配点。
vector<int> G1[N], G2[N];
int in[N], n, m, q[N], idx;
int f[N][25], dep[N];
void tupo() {
    idx = 0; queue<int> que;
    for (int i = 1; i <= n; i++) {
        if (!in[i]) que.push(i);
        if (!G1[i].size()) G1[i].push_back(0), in[0]++;
    }
    while (que.size()) {
        int now = que.front();
        que.pop();
        q[++idx] = now;
        for (auto v : G1[now]) {
            in[v]--;
            if (!in[v]) que.push(v);
        }
    }
}
int lca(int x, int y) {
    if (dep[x] > dep[y]) swap(x, y);
    for (int i = 20; i >= 0; i--) if (dep[f[y][i]] >= dep[x]) y = f[y][i];

    if (x == y) return x;
    for (int i = 20; i >= 0; i--) {
        if (f[y][i] != f[x][i])
            y = f[y][i], x = f[x][i];
    }
    return f[x][0];
}

void solve() {
    int T; cin >> T; while (T--) {
        cin >> n >> m;
        for (int i = 0; i <= n; i++) G1[i].clear(), G2[i].clear(), dep[i] = 0;
        for (int i = 1; i <= m; i++) {
            int u, v; cin >> u >> v;
            G1[u].push_back(v);
            in[v]++;
        }
        tupo();
        for (int i = n; i >= 1; i--) {
            int now = q[i];
            if (!G1[now].size()) continue;
            int _lca = G1[now][0];
            for (int j = 1; j < G1[now].size(); j++) _lca = lca(_lca, G1[now][j]);
            f[now][0] = _lca;
            dep[now] = dep[_lca] + 1;
            for (int j = 1; j <= 20; j++) f[now][j] = f[f[now][j-1]][j-1];
        }
        int qry; cin >> qry; while (qry--) {
            int x, y; cin >> x >> y;
            cout << dep[x] + dep[y] - dep[lca(x, y)] << endl;
        }
    }
}
```

6.2 spfa 判负环（01 分数规划）

```

//bfs版
#include <bits/stdc++.h>
using namespace std;
double eps = 1e-4;
const int N = 1e3 + 5;
const int M = 5e3 + 5;
struct edge {
    int to, next;
    double vi;
}e[M * 2];

int vis[N], n, cnt, m, h[N], times[N];
double dis[N];
int ve[N];

void add(int u, int v, double w) {
    e[cnt].to = v;
    e[cnt].vi = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}
bool spfa(double mid) {
    queue<int> q;
    for (int i = 1; i <= n; i++) {
        q.push(i);
        vis[i] = 1, dis[i] = 0, times[i] = 0;
    }
    while (q.size()) {
        int u = q.front();
        q.pop();
        vis[u] = 0;
        for (int i = h[u]; ~i; i = e[i].next) {
            int v = e[i].to;
            double va = mid * e[i].vi - (double)ve[u];
            if (dis[v] > dis[u] + va) {
                dis[v] = dis[u] + va;
                times[v] = times[u] + 1;
                if (times[v] >= n) return true;
                if (!vis[v]) {
                    q.push(v);
                    vis[v] = 1;
                }
            }
        }
    }
    return false;
}

//dfs版
#include <bits/stdc++.h>
using namespace std;
double eps = 1e-9;
const int N = 3e3 + 5;
const int M = 1e4 + 5;
struct edge {
    int to, next;
    double vi;
}e[M << 1];

int vis[N], n, cnt, m, h[N], times[N], flag;
double dis[N];

void add(int u, int v, double w) {
    e[cnt].to = v;
    e[cnt].vi = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}
void spfa(int u, double mid) {
    vis[u] = 1;
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        double tmp = dis[u] + e[i].vi - mid;
        if (dis[v] > tmp) {
            dis[v] = tmp;
            if (vis[v]) {
                flag = 1;
                return;
            }
            spfa(v, mid);
        }
    }
    vis[u] = 0;
}

bool check(double mid) {
    memset(vis, 0, sizeof vis);
    memset(times, 0, sizeof times);
    for (int i = 1; i <= n; i++) dis[i] = 0;
    for (int i = 1; i <= n; i++) {
        flag = 0;
        spfa(i, mid);
        if (flag) return true;
    }
    return false;
}

int main() {
    scanf("%d %d", &n, &m);
    memset(h, -1, sizeof h);

    for (int i = 1; i <= m; i++) {
        int x, y;
        double z;
        scanf("%d %d %lf", &x, &y, &z);
        add(x, y, z);
    }

    double l = -1e5, r = 1e5, mid;
    while (r - l > eps) {
        mid = (l + r) * 0.5;
        if (check(mid)) r = mid;
        else l = mid;
    }
    printf("%.8lf", mid);
}
```

6.3 边双连通分量 +e-dcc 缩点

```
const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int in[N];
int ans;
struct E_DCC {
    int dfn[N], low[N], bridge[M], h1[N], h2[N], belong[N];
    int cnt1, cnt2, idx, scc, num;
    struct Edge {
        int to, next;
    } e1[N << 1], e2[N << 1];

    void init(int n, int m) {
        idx = cnt1 = cnt2 = 0;
        for (int i = 0; i <= n; i++) h1[i] = h2[i] = -1;
        for (int i = 0; i <= n; i++) belong[i] = low[i] = dfn[i] = 0;
        for (int i = 0; i <= 2*m; i++) bridge[i] = 0;
    }

    void add(int u, int v) {
        e1[cnt1].to = v;
        e1[cnt1].next = h1[u];
        h1[u] = cnt1++;
    }

    void add_cc(int u, int v) {
        e2[cnt2].to = v;
        e2[cnt2].next = h2[u];
        h2[u] = cnt2++;
    }

    void tarjan(int u, int in_edge) {
        dfn[u] = low[u] = ++idx;
        for (int i = h1[u]; ~i; i = e1[i].next) {
            int v = e1[i].to;
            if (!dfn[v]) {
                tarjan(v, i);
                low[u] = min(low[u], low[v]);
                if (low[v] > dfn[u]) num++, bridge[i] = bridge[i ^ 1] = 1;
            } else if (i != (in_edge ^ 1))
                low[u] = min(low[u], dfn[v]);
        }
    }

    void rebuild(int x) {
        belong[x] = scc;
        for (int i = h1[x]; ~i; i = e1[i].next) {
            int v = e1[i].to;
            if (belong[v] || bridge[i]) continue;
            rebuild(v);
        }
    }
}cc;
void solve() {
    cc.init(n, m);
    for (int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        cc.add(u, v), cc.add(v, u);
    }
    for (int i = 1; i <= n; i++) if (!cc.dfn[i]) cc.tarjan(i, 0);
    for (int i = 1; i <= n; i++) {
        if (!cc.belong[i]) {
            ++cc.scc; cc.rebuild(i);
        }
    }
    for (int i = 0; i < cc.cnt1; i+=2) {
        int u = cc.e1[i].to, v = cc.e1[i^1].to;
        if (cc.belong[u] == cc.belong[v]) continue;
        cc.add_cc(cc.belong[u], cc.belong[v]), cc.add_cc(cc.belong[v], cc.belong[u]);
        in[cc.belong[u]]++, in[cc.belong[v]]++;
    }
}
```

6.4 差分约束系统

```
/*
1.求s[n] - s[1]的最小值时，转换为1到n的最长路模型

2.求s[n] - s[1]的最大值时，转换为1到n的最短路模型

```

```
ep:
求最短路时
1. xi + T >= xj --> i 向 j 建一条长度为T的边
2. xi + T > xj  --> xi + T-1 >= xj  i 向 j 建一条长度为T-1的边
3. xi + T == xj  --> x + T >= xj && x + T <= xj (同(1))

求最长路时，将 >= 换成 <= 即可
*/
```

6.5 点双连通分量 +v-dcc 缩点

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
int dfn[N], low[N], idx, cut[N], cnt, stk[N], tp;
vector<int> g[N], dcc[N];
int n, m;
void tarjan(int u, int root) {
    dfn[u] = low[u] = ++idx;
    stk[++tp] = u;
    if (u == root && g[u].empty()) {
        dcc[++cnt].push_back(u);
        return;
    }
    int child = 0;
    for (auto v : g[u]) {
        if (!dfn[v]) {
            tarjan(v, root);
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                child++;
                if (child > 1 || u != root) cut[u] = 1;
                cnt++;
                int y;
                while (y = stk[tp--]) {
                    dcc[cnt].push_back(y);
                    if (y == v) break;
                }
                dcc[cnt].push_back(u);
            }
        }
        else
            low[u] = min(low[u], dfn[v]);
    }
}

int main() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i, i);
    for (int i = 1; i <= n; i++) {
        for (auto x : dcc[i]) cout << x << " ";
        cout << endl;
    }
}

/*
9 11
1 2
1 5
2 3
2 5
3 4
4 5
1 6
6 7
9 8
6 9
6 8
*/
```

6.6 二分图 HK 算法

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <queue>
#include <stack>
#include <string>
#include <set>
#include <map>
#include <bitset>
using namespace std;
#define ACM_LOCAL

const int N = 3e3 + 5;
const int INF = 0x3f3f3f3f;
typedef pair<int, int> PII;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

int h[N], cnt, n;  //存图
int dx[N], dy[N], dis;                   //左边部分距离，右边部分距离，记录右边部分没有被匹配过的点的最大距离
int machx[N], machy[N];              //左边部分点匹配右边点，左边部分的点个数，右边部分点匹配左边的点
bool vist[N];

struct edge {
    int to, next;
}e[N * N];

void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

```

```
bool searchpath() {//找有没有增广路
    queue<int>q;
    dis = INF;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));

    for (int i = 1; i <= n; i++)
        if (machx[i] == -1)
            q.push(i), dx[i] = 0;

    while (!q.empty()) {
        int u = q.front(); q.pop();
        if (dx[u] > dis)
            break;
        for (int i = h[u]; i != -1; i = e[i].next) {
            int v = e[i].to;
            if (dy[v] == -1) {
                dy[v] = dx[u] + 1;
                if (machy[v] == -1)
                    dis = dy[v];
                else {
                    dx[machy[v]] = dy[v] + 1;
                    q.push(machy[v]);
                }
            }
        }
    }
    return dis != INF;
}
bool findroad(int u) {
    for (int i = h[u]; i != -1; i = e[i].next) {
        int v = e[i].to;
        if (!vist[v] && dy[v] == dx[u] + 1) {
            vist[v] = 1;
            if (machy[v] != -1 && dy[v] == dis)
                continue;
            if (machy[v] == -1 || findroad(machy[v])) {
                machy[v] = u; machx[u] = v;  return true;
            }
        }
    }
    return false;
}
int MaxMatch() {
    int ans = 0;
    memset(machx, -1, sizeof(machx));
    memset(machy, -1, sizeof(machy));
    while (searchpath()) {
        memset(vist, 0, sizeof(vist));
        for (int i = 1; i <= n; i++)
            if (machx[i] == -1)
                ans += findroad(i);
    }
    return ans;
}
```

6.7 二分图 KM 算法

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <algorithm>
#include <queue>
#include <stack>
#include <string>
#include <set>
#include <map>
#include <bitset>
using namespace std;
#define ACM_LOCAL

const int INF = 0x3f3f3f3f;
typedef pair<int, int> PII;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
const int N = 310;
int n, m, match[N], pre[N];
bool vis[N];
ll g[N][N];
ll val1[N], val2[N], slack[N];

void bfs(int p){
    memset(pre, 0, sizeof pre);
    memset(slack, 0x3f, sizeof slack);
    match[0] = p;
    int x = 0, nex = 0;
    do{
        vis[x] = true;
        int y = match[x];
        ll d = INF;
        for (int i = 1; i <= m; i++)
            if (!vis[i]){
                if (slack[i] > val1[y] + val2[i] - g[y][i])
                    slack[i] = val1[y] + val2[i] - g[y][i], pre[i] = x;
                if (slack[i] < d)
                    d = slack[i], nex = i;
            }
        for (int i = 0; i <= m; i++){
            if (vis[i])
                val1[match[i]] -= d, val2[i] += d;
            else
                slack[i] -= d;
        }
        x = nex;
    } while (match[x]);
    while (x){
        match[x] = match[pre[x]];
        x = pre[x];
    }
}

ll KM(){
    memset(match, 0, sizeof match);
    memset(val1, 0, sizeof val1);
    memset(val2, 0, sizeof val2);
    for (int i = 1; i <= n; i++){
        memset(vis, false, sizeof vis);
        bfs(i);
    }
    ll res = 0;
    for (int i = 1; i <= m; i++) res += g[match[i]][i];
    return res;
}
```

6.8 二分图匈牙利算法

```
#include <bits/stdc++.h>
using namespace std;
const int N = 10010;
int match[N], vis[N];
int n, m, ans;
vector<int> g[N];

bool dfs(int x) {
    for (auto v : g[x]) {
        if (!vis[v]) {
            vis[v] = 1;
            if (!match[v] || dfs(match[v])) {
                match[v] = x;
                return true;
            }
        }
    }
    return false;
}
```

6.9 割边

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100010;
int dfn[N], low[N], idx;
int n, m, ans;
vector<int> g[N];
void tarjan(int u, int fa) {
    dfn[u] = low[u] = ++idx;
    for (auto v : g[u]) {
        if(v == fa) continue;
        if (!dfn[v]) {
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u]) ans++;
        }
        else
            low[u] = min(low[u], dfn[v]);
    }
}

int main() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }

    tarjan(1, 0);
    cout << m - ans;
}
```

6.10 割点

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
int dfn[N], low[N], idx, cut[N], siz[N], n, m;
vector<int> g[N];
int n, m;
void tarjan(int u, int root) {
    dfn[u] = low[u] = ++idx;
    int child = 0;
    for (auto v : g[u]) {
        if (!dfn[v]) {
            tarjan(v, root);
            siz[u] += siz[v];
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                child++;
                if (child > 1 || u != root) cut[u] = 1;
            }
        }
        else
            low[u] = min(low[u], dfn[v]);
    }
}
```

6.11 克鲁斯卡尔重构树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int M = 2e5 + 10;
struct Edge {
    int u, v, w;
    bool operator < (const Edge &rhs) const {
        return w > rhs.w;
        // > 最大边中的最小值
        // < 最小边中的最大值
    }
} e[M];
int n, m, son[N], dep[N], pre[N], siz[N], top[N], tot, q, rnk[N], dfn[N], fat[N];
int cnt, val[N], fa[N];
int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }

vector<int> g[N];

void dfs1(int u, int fa) {
    son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;
    for (auto v : g[u]) {
        if (v == fa) continue;
        dfs1(v, u);
        siz[u] += siz[v];
        if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
    }
}

void dfs2(int u, int t) {
    rnk[dfn[u] = ++tot] = u; top[u] = t;
    if (son[u] == -1) return;
    dfs2(son[u], t);
    for (auto v : g[u]) {
        if (v != son[u] && v != fat[u]) dfs2(v, v);
    }
}

int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (dep[top[u]] > dep[top[v]])
            u = pre[top[u]];
        else
            v = pre[top[v]];
    }
    return dep[u] > dep[v] ? v : u;
}

void exKruskal() {
    cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
    sort(e + 1, e + 1 + m);
    for (int i = 1; i <= m; i++) {
        int u = find(e[i].u), v = find(e[i].v);
        if (u == v) continue;
        val[++cnt] = e[i].w;
        fa[u] = fa[v] = cnt;
        g[u].push_back(cnt), g[cnt].push_back(u);
        g[v].push_back(cnt), g[cnt].push_back(v);
        if (cnt == (n << 1) - 1) break;//最多2N-1个点
    }
    for (int i = 1; i <= cnt; i++)
        if (!siz[i]) {//未访问过
            int rt = find(i);//下树剖lca
            dfs1(rt, 0); dfs2(rt, rt);
        }
}
```

6.12 克鲁斯卡尔重构树上建主席树

```
#include <bits/stdc++.h>
#define ACM_LOCAL
using namespace std;
typedef long long ll;
const int INF = 0x3f3f3f3f;
const int N = 2e5 + 10;
const int M = 5e5 + 10;
const int MOD = 1e9 + 7;

struct Hash {
```

```cpp
    int b[N], tot;
    void init() { tot = 0; }
    void insert(int x) { b[++tot] = x; }
    void build() {
        sort(b + 1, b + 1 + tot);
        tot = unique(b + 1, b + 1 + tot) - (b + 1);
    }
    int pos(int x) { return lower_bound(b + 1, b + 1 + tot, x) - b; }
} ha;
struct Edge {
    int u, v, w;
    bool operator < (const Edge &rhs) const {
        return w < rhs.w;
        // > 最大边中的最小值
        // < 最小边中的最大值
    }
} e[M];

int n, m, d[N], dfn[N], t, f[N][20], q, idx, rnk[N], in[N], out[N];
int cnt, val[N], fa[N], a[N];
int rt[N], NodeNum;
int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }

vector<int> g[N];

void dfs(int x, int fa) {
    rnk[dfn[x] = ++idx] = x, f[x][0] = fa, d[x] = d[fa] + 1;
    for (int i = 1; i <= 19; i++) f[x][i] = f[f[x][i-1]][i-1];
    in[x] = idx;
    for (auto &y : g[x])
        if (y != fa) dfs(y, x);
    out[x] = idx;
}

void exKruskal() {
    cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
    sort(e + 1, e + 1 + m);
    for (int i = 1; i <= m; i++) {
        int u = find(e[i].u), v = find(e[i].v);
        if (u == v) continue;
        val[++cnt] = e[i].w;
        fa[u] = fa[v] = cnt;
        g[u].push_back(cnt), g[cnt].push_back(u);
        g[v].push_back(cnt), g[cnt].push_back(v);
        if (cnt == (n << 1) - 1) break;//最多2N-1个点
    }
    for (int i = 1; i <= cnt; i++)
        if (!d[i]) {//未访问过
            int rt = find(i);//下树剖lca
            dfs(rt, 0);
        }
}
int FindPoint (int x, int p) {
    for (int i = 20; i >= 0; i--)
        if (d[x] > (1 << i) && val[f[x][i]] <= p) x = f[x][i];
    return x;
}

struct {
    int t[N << 5], lc[N << 5], rc[N << 5];
    int update(int pre, int l, int r, int x) {
        int num = ++NodeNum;
        lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
        if (l != r) {
            int mid = (l + r) >> 1;
            if (x <= mid) lc[num] = update(lc[pre], l, mid, x);
            else rc[num] = update(rc[pre], mid + 1, r, x);
        }
        return num;
    }
    int query(int pre, int now, int l, int r, int k) {
        if (t[now] - t[pre] < k) return -1;//如果之间的数少于k个，返回-1
        k = t[now] - t[pre] - k + 1;//这里主席树写的是第k小，转换一下变成第k大
        while (l < r) {
            int sum = t[lc[now]] - t[lc[pre]];
            if (k <= sum) {
                now = lc[now], pre = lc[pre];
                r = l + r >> 1;
            }
            else {
                now = rc[now], pre = rc[pre];
                l = (l + r >> 1) + 1;
                k -= sum;
            }
        }
        return ha.b[l];
    }
}hjt;

int ask(int x, int p, int k) {
    int pos = FindPoint(x, p);
    return hjt.query(rt[in[pos]-1], rt[out[pos]], 1, ha.tot, k);
}

void solve() {
    cin >> n >> m >> q;
    for (int i = 1; i <= n; i++) cin >> a[i], ha.insert(a[i]);
    ha.build();
    for (int i = 1; i <= m; i++)
        cin >> e[i].u >> e[i].v >> e[i].w;
    exKruskal();
    for (int i = 1; i <= idx; i++) {
        rt[i] = rt[i-1];
        if (rnk[i] <= n)
            rt[i] = hjt.update(rt[i-1], 1, ha.tot, ha.pos(a[rnk[i]]));
    }

    while (q--) {
        int v, x, k;
        cin >> v >> x >> k;
        printf("%d\n", ask(v, x, k));
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
```

```
#ifdef ACM_LOCAL
    freopen("input", "r", stdin);
    freopen("output", "w", stdout);
#endif
    solve();
}
```

6.13 欧拉图

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int M = 2e5 + 10;

struct edge {
    int to, next;
}e[M<<1];

int n, m, t, stk[N], tp, cnt, h[N], vis[M<<1], ans[N];

void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

void euler() {
    stk[++tp] = 1;
    while (tp > 0) {
        int x = stk[tp], i = h[x];
        while (i && vis[i]) i = e[i].next;
        if (i) {
            stk[++tp] = e[i].to;
            vis[i] = vis[i ^ 1] = 1;
            h[x] = e[i].next;
        }
        else {
            tp--;
            ans[++t] = x;
        }
    }
}

int main() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        add(x, y), add(y, x);
    }
    euler();
    for (int i = t; i >= 1; i--) cout << ans[i] << " ";
}

/*
7 9
1 2
1 3
```

```
1 4
1 5
2 3
4 5
5 6
6 7
5 7
*/
```

6.14 强联通分量

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 10010;
int h[N], cnt, n, m, scc;
int dfn[N], low[N], idx, tp, in_stk[N], vis[N], sd[N];
set<int> sc[N];
struct edge{
    int to, next;
}e[N*2];

void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

void tarjan(int x) {
    low[x] = dfn[x] = ++idx;
    vis[x] = 1;
    in_stk[++tp] = x;
    for (int i = h[x]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (!dfn[v]) {
            tarjan(v);
            low[x] = min(low[x], low[v]);
        }
        else if (vis[v]) {
            low[x] = min(low[x], dfn[v]);
        }
    }
    if (low[x] == dfn[x]) {
        int y;
        ++scc;
        while(y = in_stk[tp--]) {
            sd[y] = scc;
            vis[y] = 0;
            sc[scc].insert(y);
            if (x == y) break;
        }
    }
}

int main() {
    cin >> n >> m;
    memset(h, -1, sizeof h);
    for (int i = 0; i < m; i ++ ) {
        int x, y;
        cin >> x >> y;
        add(x, y);
    }
    for (int i = 1; i <= n; i ++) if (!dfn[i]) tarjan(i);

    for (int i = 1; i <= scc; i ++) {
        cout << "#" << i << ":";
        for (auto x : sc[i]) {
            cout << x << " ";
        }
        cout << endl;
    }
}
```

6.15 限制流量费用流 (Dijkstra)

```

struct Edge {
    int to;
    int cap, dis; //容量、费用
    int rev;      //(u,v)的反向弧中,v在u的位置
    Edge() {}
    Edge(int to, int cap, int dis, int rev): to(to), cap(cap), dis(dis), rev(rev) {}
};
vector<Edge> G[N];
struct Pre {  //记录前驱
    int node; //前驱结点
    int edge; //对应的边
} pre[N];
int h[N];   //势能函数
ll dis[N]; //费用
void addEdge(int x, int y, int cap, int cost) {
    G[x].push_back(Edge(y, cap, cost, (int)G[y].size()));      //正向边
    G[y].push_back(Edge(x, 0, -cost, (int)(G[x].size() - 1))); //反向边
}
bool Dijkstra(int S, int T) {
    memset(dis, 0x3f3f3f3f, sizeof dis);
    dis[S] = 0;

    priority_queue<PII, vector<PII>, greater<PII>> Q;
    Q.push(PII(dis[S], S));
    while (!Q.empty()) {
        PII now = Q.top();
        Q.pop();

        int u = now.second;
        if (dis[u] < now.first)
            continue;

        for (int i = 0; i < G[u].size(); i++) {
            int v = G[u][i].to;
            int cap = G[u][i].cap;
            int w = G[u][i].dis;
            if (cap && dis[v] > w + dis[u] + h[u] - h[v]) {
                dis[v] = w + dis[u] + h[u] - h[v]; //进行松弛
                pre[v].node = u;                   //记录前驱点
                pre[v].edge = i;                   //记录前驱边
                Q.push(PII(dis[v], v));
            }
        }
    }
    if (dis[T] == INF)
        return false;
    else {
        for (int i = 0; i <= T + 1; i++) //对于势能函数，每次加上当前轮的dis
            h[i] += dis[i];
        return true;
    }
}
void maxFlow(int S, int T, int &flow, ll &cost) {//flow 代表最大总流量
    memset(h, 0, sizeof(h));
    memset(pre, 0, sizeof(pre));

    int newFlow = 0;                 //增广流量
    while (flow && Dijkstra(S, T)) { //当无法增广时，即找到答案
        int minn = INF;
        for (int i = T; i != S; i = pre[i].node) {
            int node = pre[i].node;
            int edge = pre[i].edge;
            minn = min(minn, G[node][edge].cap);
        }

        flow -= minn;        //原流量
        newFlow += minn;     //增广流量
        cost += 1ll * h[T] * minn; //增广流花销

        for (int i = T; i != S; i = pre[i].node) {
            int node = pre[i].node;
            int edge = pre[i].edge;
            int rev = G[node][edge].rev;
            G[node][edge].cap -= minn;
            G[i][rev].cap += minn;
        }
    }
}
```

6.16 一般图最大匹配（带花树）

```
struct Edge {
    int to, next;
} e[M];
int cnt, n, m, nn, h[N], fa[N], match[N], pre[N], tic[N], tim, ty[N];
queue<int> que;
void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}

int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }

int lca(int x, int y) {
    for (tim++;; swap(x, y))
        if (x) {
            x = find(x);
            if (tic[x] == tim) return x;
            tic[x] = tim;
            x = pre[match[x]];
        }
}

void shrink(int x, int y, int p) {
    while (find(x) != p) {
        pre[x] = y;
        y = match[x];
        if (ty[y] == 2) ty[y] = 1, que.push(y);
        if (find(x) == x) fa[x] = p;
        if (find(y) == y) fa[y] = p;
        x = pre[y];
    }
}

bool aug(int s) {
    for (int i = 1; i <= n; i++) fa[i] = i, ty[i] = pre[i] = 0;
    while (!que.empty()) que.pop();
    que.push(s); ty[s] = 1;
    while (!que.empty()) {
        int x = que.front();
        que.pop();
        for (int i = h[x], y = e[i].to; ~i; i = e[i].next, y = e[i].to) {
            if (find(x) == find(y) || ty[y] == 2) continue;
            if (!ty[y]) {
                ty[y] = 2;
                pre[y] = x;
                if (!match[y]) {
                    for (int tmp; y; y = tmp, x = pre[y])
                        tmp = match[x], match[x] = y, match[y] = x;
                    return 1;
                } else ty[match[y]] = 1, que.push(match[y]);
            } else if (ty[y] == 1) {
                int p = lca(x, y);
                shrink(x, y, p);
                shrink(y, x, p);
            }
        }
    }
    return 0;
}
int calc() {
    int ans = 0;
    for (int i = 1; i <= n; i++) {
        if (!match[i] && aug(i)) ans++;
    }
    return ans;
}
void init() {
    for (int i = 1; i <= n; i++) {
        match[i] = tic[i] = 0;
        h[i] = -1;
    }
    cnt = tim = 0;
}
```

6.17 最大独立集 (一般图)

```
const int N = 200 + 5;
int G[N][N];
int ans[N];
int vis[N];
int res, n;
void dfs(int x, int cnt) {
    if (x > n) {
        if (cnt > res) {
            res = cnt;
            for (int i = 1; i <= n; i++)
                ans[i] = vis[i];
        }
        return;
    }
    if (cnt + n - x + 1 < res) return;
    int pd = 0;
    for (int i = 1; i < x; i++)
        if (vis[i] && G[i][x]) {
            pd = 1;
            break;
        }
    if (!pd) {
        vis[x] = 1;
        dfs(x+1, cnt+1);
        vis[x] = 0;
    }
    dfs(x+1, cnt);
}
```

6.18 最大流 (Dinic)

```
typedef long long ll;
typedef pair<int, int> PII;
const int N = 1e3 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int state[N][11], n, p, cap[N], st, ed;
struct Maxflow {
    int h[N], cnt, maxflow, deep[N], cur[N];
    struct Edge {
        int to, next;
        ll cap;
    } e[M<<1];

    void init() {
        memset(h, -1, sizeof h);
        cnt = maxflow = 0;
    }
    void add(int u, int v, int cap) {
        e[cnt].to = v;
        e[cnt].cap = cap;
        e[cnt].next = h[u];
        h[u] = cnt++;

        e[cnt].to = u;
        e[cnt].cap = 0;
        e[cnt].next = h[v];
        h[v] = cnt++;

    }

    bool bfs() {
        for (int i = 0; i <= ed; i++) deep[i] = -1, cur[i] = h[i];
        queue<int> q; q.push(st); deep[st] = 0;
        while (q.size()) {
            int u = q.front();
            q.pop();
            for (int i = h[u]; ~i; i = e[i].next) {
                int v = e[i].to;
                if (e[i].cap && deep[v] == -1) {
                    deep[v] = deep[u] + 1;
                    q.push(v);
                }
            }
        }
        if (deep[ed] >= 0) return true;
        else return false;
    }

    ll dfs(int u, ll mx) {
        ll a;
        if (u == ed) return mx;
        for (int i = cur[u]; ~i; i = e[i].next) {
            cur[u] = i;//优化
            int v = e[i].to;
            if (e[i].cap && deep[v] == deep[u] + 1 && (a = dfs(v, min(e[i].cap, mx)))) {
                e[i].cap -= a;
                e[i ^ 1].cap += a;
                return a;
            }
        }
        return 0;
    }

    void dinic() {
        ll res;
        while (bfs()) {
            while (1) {
                res = dfs(st, INF);
                if (!res) break;
                maxflow += res;
            }
        }
    }
}mf;
```

6.19 最大团 (一般图)

```
const int N = 200 + 5;
int n;
int G[N][N];
int cnt[N];//cnt[i]为>=i的最大团点数
int group[N];//最大团的点
int vis[N];//记录点的位置
int res;//最大团的数目
bool dfs(int pos, int num) {//num为已取的点数
    for (int i = pos + 1; i <= n; i++) {
        if (cnt[i] + num <= res)//剪枝，若取i但cnt[i]+已经取了的点数仍<ans
            return false;

        if (G[pos][i]) {//与当前团中元素比较，取Non-N(i)
            int j;
            for (j = 0; j < num; j++)
                if (!G[i][vis[j]])
                    break;
            if (j == num) {//若为空，则皆与i相邻，则此时将i加入到最大团中
                vis[num] = i;
                if (dfs(i, num + 1))
                    return true;
            }
        }
    }

    if (num > res) {//每添加一个点最多使最大团数+1,后面的搜索就没有意义了
        res = num;//最大团中点的数目
        for (int i = 1; i <= num; i++)//最大团的元素
            group[i] = vis[i - 1];
        return true;
    }
    return false;
}

void maxClique() {
    res = -1;
    for (int i = n; i >= 1; i--) {//枚举所有点
        vis[0] = i;
        dfs(i, 1);
        cnt[i] = res;
    }
}
```

6.20 最短路

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
const int M = 5e5 + 10;
typedef long long ll;
struct Node {
    struct Edge {
        int to, next, w;
    }e[M<<1];

    int h[N], cnt, vis[N], count[N];
    ll dis[N];
    void init() {
        memset(h, -1, sizeof h);
        cnt = 0;
    }
    void add(int u, int v, int w) {
        e[cnt].to = v;
        e[cnt].w = w;
        e[cnt].next = h[u];
        h[u] = cnt++;
    }

    struct node {
        int now; ll d;
        bool operator < (const node &rhs) const {
            return d > rhs.d;
        }
    };

    void dij(int st) {
        memset(dis, 0x3f, sizeof dis);
        memset(vis, 0, sizeof vis);
        dis[st] = 0; priority_queue<node> q;
        q.push({st, dis[st]});
        while (q.size()) {
            int u = q.top().now;
            q.pop();
            if (vis[u]) continue;
            vis[u] = 1;
            for (int i = h[u]; ~i; i = e[i].next) {
                int v = e[i].to;
                if (dis[v] > dis[u] + e[i].w) {
                    dis[v] = dis[u] + e[i].w;
                    if (!vis[v]) {
                        q.push({v, dis[v]});
                    }
                }
                else if (dis[v] == dis[u] + e[i].w) {
                    count[v]++;
                }
            }
        }
    }
}Dij;
```

6.21 最小费用最大流 (SPFA)

```
typedef long long ll;
typedef pair<int, int> PII;
const int N = 1e4 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
const int MOD = 1e9 + 7;
int st, ed;
struct node {
    int maxflow, mincost, cnt;
    int vis[N], dis[N], pre[N], last[N], h[N], flow[N];
    struct edge {
        int to, next, cap, cos;
    } e[M << 1];

    void add(int u, int v, int cap, int cos) {
        e[cnt].to = v;
        e[cnt].cap = cap;
        e[cnt].cos = cos;
        e[cnt].next = h[u];
        h[u] = cnt++;

        e[cnt].to = u;
        e[cnt].cap = 0;
        e[cnt].cos = -cos;
        e[cnt].next = h[v];
        h[v] = cnt++;
    }

    void init() {
        memset(h, -1, sizeof h);
        cnt = 0;
        mincost = maxflow = 0;
    }
    bool spfa() {
        queue<int> q;
        for (int i = 0; i <= ed; i++) dis[i] = INF, vis[i] = 0;
        vis[st] = 1, dis[st] = 0, flow[st] = INF;
        q.push(st);
        while (q.size()) {
            int u = q.front();
            q.pop(); vis[u] = 0;
            for (int i = h[u]; ~i; i = e[i].next) {
                int v = e[i].to;
                if (e[i].cap && dis[v] > dis[u] + e[i].cos) {
                    dis[v] = e[i].cos + dis[u];
                    flow[v] = min(flow[u], e[i].cap);
                    pre[v] = u;
                    last[v] = i;
                    if (!vis[v]) {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
        }
        if (dis[ed] != INF) return true;
        else return false;
    }

    void MCMF() {
        while (spfa()) {
            int now = ed;
            maxflow += flow[ed];
            mincost += flow[ed] * dis[ed];

            while (st != now) {
                e[last[now]].cap -= flow[ed];
                e[last[now] ^ 1].cap += flow[ed];
                now = pre[now];
            }
        }
    }
}mcmf;
```

6.22 最小路径生成树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
```

```cpp
typedef long long ll;
typedef pair<int, int> PII;

struct Edge {
    int to, next, w;
}e[N<<1];

int h[N], cnt;
void add(int u, int v, int w) {
    e[cnt].to = v;
    e[cnt].w = w;
    e[cnt].next = h[u];
    h[u] = cnt++;
}
int n, m, k;
struct Node {
    vector<PII> g[N];
    int vis[N];
    ll dis[N];
    void init() {
        for (int i = 1; i <= n; i++) g[i].clear();
    }

    struct node {
        int now; ll d;
        bool operator < (const node &rhs) const {
            return d > rhs.d;
        }
    };

    void dij(int st) {
        memset(dis, 0x3f, sizeof dis);
        memset(vis, 0, sizeof vis);
        dis[st] = 0; priority_queue<node> q;
        q.push({st, dis[st]});
        while (q.size()) {
            int u = q.top().now;
            q.pop();
            if (vis[u]) continue;
            vis[u] = 1;
            for (auto item : g[u]) {
                int v = item.first;
                int w = item.second;
                if (dis[v] > dis[u] + w) {
                    dis[v] = dis[u] + w;
                    if (!vis[v]) {
                        q.push({v, dis[v]});
                    }
                }
            }
        }
    }
}Dij;
int tap[N];
void build_tree(int x) {
    tap[x] = 1;
    for (auto item : Dij.g[x]) {
        int v = item.first;
        int w = item.second;
        if (tap[v]) continue;
        if (Dij.dis[v] == Dij.dis[x] + w) {
            add(x, v, w), add(v, x, w);
            build_tree(v);
        }
    }
}

int main() {
    scanf("%d %d %d", &n, &m, &k);
    Dij.init();
    for (int i = 1; i <= m; i++) {
        int u, v, w; scanf("%d %d %d", &u, &v, &w);
        Dij.g[u].push_back(PII{v, w});
        Dij.g[v].push_back(PII{u, w});
    }
    Dij.dij(1);
    for (int i = 1; i <= n; i++)
        sort(Dij.g[i].begin(), Dij.g[i].end());
    memset(h, -1, sizeof h);
    build_tree(1);
}
```

6.23 最小生成树 boruvka

```
ll boruvka() {
    for (int i = 1; i <= n; i++) fa[i] = i;
    ll ans = 0, num = 0;
    while (num < n-1) {
        int tmp = 0;
        for (int i = 1; i <= n; i++) E[i] = PII{INF, INF};
        for (int i = 1; i <= m; i++) {
            int fx = find(e[i].u);
            int fy = find(e[i].v);
            if (fx == fy) continue;
            tmp++;
            E[fx] = min(E[fx], PII{e[i].w, i});
            E[fy] = min(E[fy], PII{e[i].w, i});
        }
        if (tmp == 0) break;
        for (int i = 1; i <= m; i++) {
            int fx = find(e[i].u);
            int fy = find(e[i].v);
            if (fx == fy) continue;
            if (E[fx] == PII{e[i].w, i} || E[fy] == PII{e[i].w, i}) {
                ans += e[i].w;
                num++;
                fa[fx] = fy;
            }
        }
    }
    if (num < n-1) return -1;
    else return ans;
}
```

6.24 最小生成树 prim

```
int prim() {
    memset(vis, 0, sizeof vis);
    memset(dis, 0x3f, sizeof dis);
    int ans = 0;
    for (int i = h[1]; ~i; i = e[i].next) {
        dis[e[i].to] = min(dis[e[i].to], e[i].vi);
    }
    dis[1] = 0, vis[1] = 1;
    for (int i = 2; i <= n; i++) {
        int minn = INF;
        int k;
        for (int j = 1; j <= n; j++) {
            if (!vis[j] && dis[j] < minn) {
                minn = dis[j];
                k = j;
            }
        }
        if (minn == INF) return -1;
        ans += minn;
        vis[k] = 1;
        for (int j = h[k]; ~j; j = e[j].next) {
            dis[e[j].to] = min(dis[e[j].to], e[j].vi);
        }
    }
    return ans;
}
```

7 杂项

```
vector<int> TreeToPrufer(int n) {// 1~n-2个数
    vector<int> pru(n+1);
    vector<int> in(n+1);
    vector<int> fa(n+1);
    for (int i = 1; i < n; i++) read(fa[i]), ++in[fa[i]];
    for (int i = 1, j = 1; i <= n - 2; i++, j++) {
        while (in[j]) ++j; pru[i] = fa[j];
        while (i <= n - 2 && !--in[pru[i]] && pru[i] < j) pru[i+1] = fa[pru[i]], ++i;
    }
    return pru;
}

vector<int> PruferToTree(int n) {// 1~n-1个数
    vector<int> fa(n+1);
    vector<int> in(n+1);
    vector<int> pru(n+1);
    for (int i = 1; i <= n - 2; i++) read(pru[i]), ++in[pru[i]]; pru[n-1] = n;
    for (int i = 1, j = 1; i < n; i++, j++) {
        while (in[j]) ++j; fa[j] = pru[i];
        while (i < n && !--in[pru[i]] && pru[i] < j) fa[pru[i]] = pru[i+1], ++i;
    }
    return fa;
}
```

7.1 DEBUG

```
#define debug(a...) cout << "(" << (#a) << ")" << " = ("; DEBUG(a)
template<typename T> void DEBUG(T value) {
    cout << value << ")" << endl;
}
template<typename T1, typename... T2>
void DEBUG(T1 now, T2... other) {
    cout << now << ", ", DEBUG(other...);
}
```

7.2 MODINT

```
namespace MODINT {
    template<unsigned M_> struct ModInt {
        static constexpr unsigned M = M_;
        unsigned x;
        constexpr ModInt() : x(0U) {}
        constexpr ModInt(unsigned x_) : x(x_ % M) {}
        constexpr ModInt(unsigned long long x_) : x(x_ % M) {}
        constexpr ModInt(int x_) : x(((x_ %= static_cast<int>(M)) < 0) ? (x_ +
    static_cast<int>(M)) : x_) {}
        constexpr ModInt(long long x_) : x(
              ((x_ %= static_cast<long long>(M)) < 0) ? (x_ + static_cast<long long>(M)
    ) : x_) {}
        ModInt &operator+=(const ModInt &a) {
            x = ((x += a.x) >= M) ? (x - M) : x;
            return *this;
        }
        ModInt &operator-=(const ModInt &a) {
            x = ((x -= a.x) >= M) ? (x + M) : x;
            return *this;
        }
        ModInt &operator*=(const ModInt &a) {
            x = (static_cast<unsigned long long>(x) * a.x) % M;
            return *this;
        }
        ModInt &operator/=(const ModInt &a) { return (*this *= a.inv()); }
        ModInt quick_pow(long long e) const {
            if (e < 0) return inv().quick_pow(-e);
            ModInt a = *this, b = 1U;
            for (; e; e >>= 1) { if (e & 1) b *= a; a *= a; }
            return b;
        }
        ModInt inv() const {
            unsigned a = M, b = x;
            int y = 0, z = 1;
            for (; b;) {
                const unsigned q = a / b;
                const unsigned c = a - q * b;
                a = b; b = c;
                const int w = y - static_cast<int>(q) * z;
                y = z; z = w;
            }
            assert(a == 1U);
            return ModInt(y);
        }
        ModInt operator+() const { return *this; }
        ModInt operator-() const { ModInt a; a.x = x ? (M - x) : 0U; return a; }
        ModInt operator+(const ModInt &a) const { return (ModInt(*this) += a); }
        ModInt operator-(const ModInt &a) const { return (ModInt(*this) -= a); }
        ModInt operator*(const ModInt &a) const { return (ModInt(*this) *= a); }
        ModInt operator/(const ModInt &a) const { return (ModInt(*this) /= a); }
        template<class T> friend ModInt operator+(T a, const ModInt &b) { return (ModInt(a) += b); }
        template<class T> friend ModInt operator-(T a, const ModInt &b) { return (ModInt(a) -= b); }
        template<class T> friend ModInt operator*(T a, const ModInt &b) { return (ModInt(a) *= b); }
        template<class T> friend ModInt operator/(T a, const ModInt &b) { return (ModInt(a) /= b); }
        explicit operator bool() const { return x; }
        bool operator==(const ModInt &a) const { return (x == a.x); }
        bool operator!=(const ModInt &a) const { return (x != a.x); }
        friend std::ostream &operator<<(std::ostream &os, const ModInt &a) { return os << a.x; }
    };
    constexpr unsigned MO = 1000000007;
//    constexpr unsigned MO = 998244353;
    using Mint = ModInt<MO>;
}
```

7.3 大数模板

```
constexpr int base = 1000000000;
constexpr int base_digits = 9;

struct bigint {
    // value == 0 is represented by empty z
    vector<int> z; // digits

    // sign == 1 <==> value >= 0
    // sign == -1 <==> value < 0
    int sign;

    bigint() : sign(1) {}

    bigint(ll v) { *this = v; }

    bigint &operator=(ll v) {
        sign = v < 0 ? -1 : 1;
        v *= sign;
        z.clear();
        for (; v > 0; v = v / base) z.push_back((int) (v % base));
        return *this;
    }

    bigint(const string &s) { read(s); }

    bigint &operator+=(const bigint &other) {
        if (sign == other.sign) {
            for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
                if (i == z.size())
                    z.push_back(0);
                z[i] += carry + (i < other.z.size() ? other.z[i] : 0);
                carry = z[i] >= base;
                if (carry)
                    z[i] -= base;
            }
        } else if (other != 0 /* prevent infinite loop */) {
            *this -= -other;
        }
        return *this;
    }

    friend bigint operator+(bigint a, const bigint &b) { return a += b; }

    bigint &operator-=(const bigint &other) {
        if (sign == other.sign) {
            if (sign == 1 && *this >= other || sign == -1 && *this <= other) {
                for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
                    z[i] -= carry + (i < other.z.size() ? other.z[i] : 0);
                    carry = z[i] < 0;
                    if (carry)
                        z[i] += base;
                }
                trim();
            } else {
                *this = other - *this;
                this->sign = -this->sign;
            }
        } else {
            *this += -other;
        }
        return *this;
    }

    friend bigint operator-(bigint a, const bigint &b) {
        return a -= b;
    }

    bigint &operator*=(int v) {
        if (v < 0) sign = -sign, v = -v;
        for (int i = 0, carry = 0; i < z.size() || carry; ++i) {
            if (i == z.size()) z.push_back(0);
            ll cur = (ll) z[i] * v + carry;
            carry = (int) (cur / base);
            z[i] = (int) (cur % base);
        }
        trim();
        return *this;
    }

    bigint operator*(int v) const { return bigint(*this) *= v; }

    friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {
        int norm = base / (b1.z.back() + 1);
        bigint a = a1.abs() * norm;
        bigint b = b1.abs() * norm;
        bigint q, r;
        q.z.resize(a.z.size());

        for (int i = (int) a.z.size() - 1; i >= 0; i--) {
            r *= base;
            r += a.z[i];
            int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;
            int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;
            int d = (int) (((ll) s1 * base + s2) / b.z.back());
            r -= b * d;
            while (r < 0) r += b, --d;
            q.z[i] = d;
        }

        q.sign = a1.sign * b1.sign;
        r.sign = a1.sign;
        q.trim();
        r.trim();
        return {q, r / norm};
    }

    friend bigint sqrt(const bigint &a1) {
        bigint a = a1;
        while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);

        int n = a.z.size();

        int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
        int norm = base / (firstDigit + 1);
        a *= norm;
        a *= norm;
        while (a.z.empty() || a.z.size() % 2 == 1) a.z.push_back(0);

        bigint r = (ll) a.z[n - 1] * base + a.z[n - 2];
        firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
        int q = firstDigit;
        bigint res;

        for (int j = n / 2 - 1; j >= 0; j--) {
            for (;; --q) {
                bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
                            (j > 0 ? (ll) a.z[2 * j - 1] * base + a.z[2 * j - 2] : 0);
                if (r1 >= 0) {
                    r = r1;
                    break;
                }
            }
            (res *= base) += q;

            if (j > 0) {
                int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;
                int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;
                int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;
                q = (int) (((ll) d1 * base * base + (ll) d2 * base + d3) / (firstDigit
    * 2));
            }
        }

        res.trim();
        return res / norm;
    }

    bigint operator/(const bigint &v) const {
        return divmod(*this, v).first;
    }

    bigint operator%(const bigint &v) const {
        return divmod(*this, v).second;
    }

    bigint &operator/=(int v) {
        if (v < 0) sign = -sign, v = -v;
        for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {
            ll cur = z[i] + rem * (ll) base;
            z[i] = (int) (cur / v);
            rem = (int) (cur % v);
        }
        trim();
        return *this;
    }

    bigint operator/(int v) const {
        return bigint(*this) /= v;
    }

    int operator%(int v) const {
        if (v < 0) v = -v;
        int m = 0;
        for (int i = (int) z.size() - 1; i >= 0; --i)
            m = (int) ((z[i] + m * (ll) base) % v);
        return m * sign;
    }

    bigint &operator*=(const bigint &v) {
        return *this = *this * v;;
    }

    bigint &operator/=(const bigint &v) {
        return *this = *this / v;
    }

    bool operator<(const bigint &v) const {
        if (sign != v.sign)
            return sign < v.sign;
        if (z.size() != v.z.size())
            return z.size() * sign < v.z.size() * v.sign;
        for (int i = (int) z.size() - 1; i >= 0; i--)
            if (z[i] != v.z[i])
                return z[i] * sign < v.z[i] * sign;
        return false;
    }

    bool operator>(const bigint &v) const { return v < *this; }

    bool operator<=(const bigint &v) const { return !(v < *this); }

    bool operator>=(const bigint &v) const { return !(*this < v); }

    bool operator==(const bigint &v) const { return !(*this < v) && !(v < *this); }

    bool operator!=(const bigint &v) const { return *this < v || v < *this; }

    void trim() {
        while (!z.empty() && z.back() == 0) z.pop_back();
        if (z.empty()) sign = 1;
    }

    bool isZero() const {
        return z.empty();
    }

    friend bigint operator-(bigint v) {
        if (!v.z.empty()) v.sign = -v.sign;
        return v;
    }

    bigint abs() const {
        return sign == 1 ? *this : -*this;
    }

    ll longValue() const {
        ll res = 0;
        for (int i = (int) z.size() - 1; i >= 0; i--)
            res = res * base + z[i];
        return res * sign;
    }

    friend bigint gcd(const bigint &a, const bigint &b) {
        return b.isZero() ? a : gcd(b, a % b);
    }

    friend bigint lcm(const bigint &a, const bigint &b) {
        return a / gcd(a, b) * b;
    }

    void read(const string &s) {
        sign = 1;
        z.clear();
        int pos = 0;
        while (pos < s.size() && (s[pos] == '-' || s[pos] == '+')) {
            if (s[pos] == '-') sign = -sign;
            ++pos;
        }
        for (int i = (int) s.size() - 1; i >= pos; i -= base_digits) {
            int x = 0;
            for (int j = max(pos, i - base_digits + 1); j <= i; j++)
                x = x * 10 + s[j] - '0';
            z.push_back(x);
        }
        trim();
    }

    friend istream &operator>>(istream &stream, bigint &v) {
        string s;
        stream >> s;
        v.read(s);
        return stream;
    }

    friend ostream &operator<<(ostream &stream, const bigint &v) {
        if (v.sign == -1)
            stream << '-';
        stream << (v.z.empty() ? 0 : v.z.back());
        for (int i = (int) v.z.size() - 2; i >= 0; --i)
            stream << setw(base_digits) << setfill('0') << v.z[i];
        return stream;
    }

    static vector<int> convert_base(const vector<int> &a, int old_digits, int new_digits) {
        vector<ll> p(max(old_digits, new_digits) + 1);
        p[0] = 1;
        for (int i = 1; i < p.size(); i++)
            p[i] = p[i - 1] * 10;
        vector<int> res;
        ll cur = 0;
        int cur_digits = 0;
        for (int v : a) {
            cur += v * p[cur_digits];
            cur_digits += old_digits;
            while (cur_digits >= new_digits) {
                res.push_back(int(cur % p[new_digits]));
                cur /= p[new_digits];
                cur_digits -= new_digits;
            }
        }
        res.push_back((int) cur);
        while (!res.empty() && res.back() == 0) res.pop_back();
        return res;
    }

    typedef vector<ll> vll;

    static vll karatsubaMultiply(const vll &a, const vll &b) {
        int n = a.size();
        vll res(n + n);
        if (n <= 32) {
            for (int i = 0; i < n; i++)
                for (int j = 0; j < n; j++)
                    res[i + j] += a[i] * b[j];
            return res;
        }

        int k = n >> 1;
        vll a1(a.begin(), a.begin() + k);
        vll a2(a.begin() + k, a.end());
        vll b1(b.begin(), b.begin() + k);
        vll b2(b.begin() + k, b.end());

        vll a1b1 = karatsubaMultiply(a1, b1);
        vll a2b2 = karatsubaMultiply(a2, b2);

        for (int i = 0; i < k; i++) a2[i] += a1[i];
        for (int i = 0; i < k; i++) b2[i] += b1[i];

        vll r = karatsubaMultiply(a2, b2);
        for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
        for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];

        for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
        for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];
        for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];
        return res;
    }

    bigint operator*(const bigint &v) const {
        vector<int> a6 = convert_base(this->z, base_digits, 6);
        vector<int> b6 = convert_base(v.z, base_digits, 6);
        vll a(a6.begin(), a6.end());
        vll b(b6.begin(), b6.end());
        while (a.size() < b.size()) a.push_back(0);
        while (b.size() < a.size()) b.push_back(0);
        while (a.size() & (a.size() - 1)) a.push_back(0), b.push_back(0);
        vll c = karatsubaMultiply(a, b);
        bigint res;
        res.sign = sign * v.sign;
        for (int i = 0, carry = 0; i < c.size(); i++) {
            ll cur = c[i] + carry;
            res.z.push_back((int) (cur % 1000000));
            carry = (int) (cur / 1000000);
        }
        res.z = convert_base(res.z, 6, base_digits);
        res.trim();
        return res;
    }
};
```

7.4 待修莫队

```
#include <bits/stdc++.h>
using namespace std;
#define ACM_LOCAL
const int N = 2e6 + 10;
```

```
int n, m, k, a[N], cnt[N], qnum, cnum, pos[N];
int Ans, ans[N], x, y;
char op;
inline int read(){
   int s=0,w=1;
   char ch=getchar();
   while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
   while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
   return s*w;
}

struct node {
    int l, r, pre, id;
}q[N];

struct node2 {
    int val, pos;
}c[N];

bool cmp(node x, node y) {
    if (x.l != y.l) return pos[x.l] < pos[y.l];
    if (x.r != y.r) return pos[x.r] < pos[y.r];
    return x.pre < y.pre;
}

inline void add(int x) {if (++cnt[a[x]] == 1) ++Ans;}

inline void sub(int x) {if (--cnt[a[x]] == 0) --Ans;}


inline void work(int now, int i) {
    if (c[now].pos >= q[i].l && c[now].pos <= q[i].r) {
        if (--cnt[a[c[now].pos]] == 0) --Ans;
        if (++cnt[c[now].val] == 1) ++Ans;
    }
    swap(c[now].val, a[c[now].pos]);// 浜ゆ崲鍦瑰彍鐨勫€煎拰鍘熸潵鐨勫€硷紝鏁堟灉鏄竴€€鐨勬椂鍊欏□ 鍘?
}

void solve() {
    n = read(), m = read();
    int siz = pow(n, 2.0 / 3.0);
    for (int i = 1; i <= n; ++i) a[i] = read(), pos[i] = (i - 1) / siz + 1;

    while (m--) {
        op = getchar();
        if (op == 'Q') {
            q[++qnum].l = read();
            q[qnum].r = read();
            q[qnum].id = qnum;
            q[qnum].pre = cnum;
        }
        else {
            c[++cnum].pos = read();
            c[cnum].val = read();
        }
    }

    sort(q+1, q+qnum+1, cmp);
```

```cpp
    int l = 1, r = 0, now = 0;
    for (int i = 1; i <= qnum; ++i) {
        while (q[i].l < l) add(--l);
        while (q[i].r > r) add(++r);
        while (q[i].l > l) sub(l++);
        while (q[i].r < r) sub(r--);
        while (now < q[i].pre) work(++now, i);
        while (now > q[i].pre) work(now--, i);
        ans[q[i].id] = Ans;
    }

    for (int i = 1; i <= qnum; ++i)  printf("%d\n", ans[i]);
}

signed main() {
    //ios_base::sync_with_stdio(false);
    //cin.tie(0);
    //cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
//#else
    solve();
#endif
    return 0;
}
```

7.5 对拍

```
/***¶□ □ □ ***/
//□ □ □ □ □ ļ□ £°test.cpp biaoda.cpp data.cpp input.txt duipai.bat:
duipai.bat:
:again
data  >  input.txt
biaoda  <  input.txt  >  biaoda_out.txt
test  <  input.txt  >  test_out.txt
fc biaoda_out.txt  test_out.txt
if not errorlevel 1 goto again
pause
```

7.6 二进制压缩枚举子集

```cpp
for (int sub = S; sub; sub = (sub - 1) & S) {
    // sub 为 S 的子集
}

a ^ b = c   -->  c ^ b = a
```

7.7 二维差分

```cpp
void Insert(int x1, int y1, int x2, int y2, int v) {
    c[x1][y1] += v;
    c[x1][y2+1] -= v;
    c[x2+1][y1] -= v;
    c[x2+1][y2+1] += v;
}
```

7.8 分块

```
int block = sqrt(n);
int num = n / block + (n % block ? 1 : 0);
for (int i = 1; i <= num; i++) {
    l[i] = (i - 1) * block + 1;
    r[i] = i * block;
}
r[num] = n;
for (int i = 1; i <= n; i++) belong[i] = (i - 1) / block + 1;
```

7.9 高精度

```
#include<cstdio>
#include <cassert>
#include<cstring>
#include<algorithm>
using namespace std;
const int MOD=10000;
const int B=10000;
const int SIZEN=505;
const int L=505;
struct Mat{
    int num[40][40];
    void init(int n){
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                num[i][j]=i*n+j;
    }
    void change(int n){
        int t_num[40][40];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) t_num[j][n-i-1]=num[i][j];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
    }
    void change1(int n){
        int t_num[40][40];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) t_num[i][n-j-1]=num[i][j];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
    }
    void change2(int n){
        int t_num[40][40];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) t_num[n-i-1][j]=num[i][j];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
    }
    void change3(int n){
        int t_num[40][40];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) t_num[j][i]=num[i][j];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
    }
    void change4(int n){
        int t_num[40][40];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) t_num[n-1-j][n-1-i]=num[i][j];
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++) num[i][j]=t_num[i][j];
    }
    void output(int n){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++) printf("%d ",num[i][j]);
            printf("\n");
        }
    }
};
struct BigInteger {
    BigInteger(int number = 0) : length(!!number) {
        assert(0 <= number && number < B);
        memset(digit, 0, sizeof(digit));
        digit[0] = number;
    }

    BigInteger normalize() {
        while (length && !digit[length - 1]) {
            length --;
        }
        return *this;
    }

    int operator[](int index) const {
        return digit[index];
    }

    int& operator[](int index) {
        return digit[index];
    }

    void output(){
        printf("%d",digit[length-1]);
        for(int i=length-2;i>=0;i--) printf("%04d",digit[i]);
        printf("\n");
    }

    int length, digit[L];
};

bool operator < (const BigInteger &a, const BigInteger &b)
{
    if (a.length != b.length) {
        return a.length < b.length;
    }
    for (int i = 0; i < a.length; ++ i) {
        if (a[i] != b[i]) {
            return a[i] < b[i];
        }
    }
    return false;
}

BigInteger operator + (const BigInteger &a, const BigInteger &b)
{
    BigInteger c;
    c.length = std::max(a.length, b.length) + 1;
    for (int i = 0, delta = 0; i < c.length; ++ i) {
        delta += a[i] + b[i];
        c[i] = delta % B;
        delta /= B;
    }
    return c.normalize();
}

BigInteger operator - (const BigInteger &a, int b)
{
    assert(0 <= b && b < B);
    BigInteger c;
    c.length = a.length;
    for (int i = 0, delta = -b; i < a.length; ++ i) {
        delta += a[i];
        c[i] = delta;
        delta = 0;
        if (c[i] < 0) {
            c[i] += B;
            delta = -1;
        }
    }
    return c.normalize();
}

BigInteger operator * (const BigInteger &a, const BigInteger &b)
{
    BigInteger c;
    c.length = a.length + b.length;
    for (int i = 0; i < a.length; ++ i) {
        for (int j = 0, delta = 0; j <= b.length; ++ j) {
            delta += a[i] * b[j] + c[i + j];
            c[i + j] = delta % B;
            delta /= B;
        }
    }
    return c.normalize();
}

BigInteger operator / (const BigInteger &a, int b)
{
    assert(0 <= b && b < B);
    BigInteger c;
    c.length = a.length;
    for (int i = c.length - 1, delta = 0; i >= 0; -- i) {
        delta = delta * B + a[i];
        c[i] = delta / b;
        delta %= b;
    }
    return c.normalize();
}
BigInteger operator ^(const BigInteger &a,int b){
    BigInteger ret,ta;
    ret=1;ta=a;
    while(b){
        if(b&1) ret=ret*ta;
        ta=ta*ta;
        b>>=1;
    }
    return ret;
}
Mat mat;
BigInteger ret,tmp;
void solve(int n,int c){
    ret=0;
    ret.normalize();
    tmp=c;
    tmp.normalize();
    if(n%2==0){
        ret=ret+(tmp^(n*n));
        ret=ret+(tmp^(n*n/4));
        ret=ret+(tmp^(n*n/2));
        ret=ret+(tmp^(n*n/4));
        ret=ret+(tmp^(n*n/2))*2;
        ret=ret+(tmp^((n*n-n)/2+n))*2;
    }
    else{
        ret=ret+(tmp^(n*n));
        ret=ret+(tmp^(n*n-1)/4+1);
        ret=ret+(tmp^(n*n-1)/2+1);
        ret=ret+(tmp^(n*n-1)/4+1);
        ret=ret+(tmp^((n*n-n)/2+n))*2;
        ret=ret+(tmp^((n*n-n)/2+n))*2;
    }
    ret=ret/8;
    ret.output();
}
int main()
{
    int n,c;
    while(scanf("%d%d",&n,&c)!=EOF)
        solve(n,c);
}
```

7.10 环形均分

```
//
// Created by SANZONG on 2021/7/8.
//
#include "bits/stdc++.h"

#define int long long
using namespace std;
const int maxn = 3000000;
int arr[maxn];
int sum[maxn];
signed main() {
//    freopen("in.txt","r",stdin);
    //前缀和为0说明前一段内部可以自己解决，其内部的前缀和绝对值相加即为前一段的子段转移花费
    int n;
    cin >> n;
    int ave = 0;
    for (int i = 1; i <= n; ++i) {
        cin >> arr[i];
        ave += arr[i];
    }
    ave /= n;
    for (int i = 1; i <= n; ++i) {
        arr[i] -= ave;
        sum[i] = sum[i-1] + arr[i];
    }
    sort(sum+1,sum+1+n);
    int mid = (n+1)/2;
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        ans += abs(sum[i] - sum[mid]);   //做前缀和相加模拟出了传递时的花费
    }
    cout << ans << endl;
}
```

7.11 回滚莫队

```
#include <bits/stdc++.h>
using namespace std;
//#define ACM_LOCAL
typedef long long ll;
const int N = 2e5 + 10;
int n, m, k, a[N], pos[N], xl[N], xr[N], b[N], _cnt[N], st[N], ed[N], clear[N];
int Max, temp, ans[N], num;

inline int read(){
    int s=0,w=1;
    char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
    return s*w;
}

struct node {
    int l, r, id;
    bool operator < (node xx) const {
        if(pos[l] == pos[xx.l]) return r < xx.r;
        else return pos[l] < pos[xx.l];
    }
}q[N];

void solve() {
    n = read();
    for (int i = 1; i <= n; i++) a[i] = read(), b[i] = a[i];

    sort(b+1, b+n+1);
    int tot = unique(b+1, b+n+1) - b-1;
    for (int i = 1; i <= n; i++) a[i] = lower_bound(b+1, b+tot+1, a[i]) - b;

    m = read();
    for (int i = 1; i <= m; i++) q[i].l = read(), q[i].r = read(), q[i].id = i;

    int siz = sqrt(n);
    for (int i = 1; i <= n; i++){
        pos[i] = i / siz;
        xl[pos[i]] = (xl[pos[i]] == 0 || xl[pos[i]] > i) ? i : xl[pos[i]];
        xr[pos[i]] = (xr[pos[i]] < i) ? i : xr[pos[i]];
    }

    sort(q+1, q+m+1);

    int l = 1, r = 0, lastblock = -1;

    for (int i = 1; i <= m; i++) {
        if (pos[q[i].l] == pos[q[i].r]) {
            int temp = 0;
            for (int j = q[i].l; j <= q[i].r; j++) {
                if (!_cnt[a[j]]) _cnt[a[j]] = j;
                else temp = max(temp, j - _cnt[a[j]]);
            }
            for (int j = q[i].l; j <= q[i].r; j++) _cnt[a[j]] = 0;
            ans[q[i].id] = temp;
        }
        else {
            if (lastblock != pos[q[i].l]) {
                l = xr[pos[q[i].l]] + 1;
                r = l - 1;
                for (int j = 1; j <= num; j++) st[clear[j]] = ed[clear[j]] = 0;
                num = 0;
                Max = 0, lastblock = pos[q[i].l];
            }
            while (r < q[i].r) {
                r++;
                ed[a[r]] = r;
                clear[++num] = a[r];
                if (!st[a[r]]) st[a[r]] = r;
                Max = max(Max, r - st[a[r]]);
            }
            temp = Max;
            while (l > q[i].l) {
                l--;
                if (ed[a[l]]) temp = max(temp, ed[a[l]] - l);
                else ed[a[l]] = l;
            }
            while (l < xr[pos[q[i].l]] + 1) {
                if (ed[a[l]] == l) ed[a[l]] = st[a[l]] = 0;
                l++;
            }
            ans[q[i].id] = temp;
        }
    }

    for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
}

signed main() {
    //ios_base::sync_with_stdio(false);
    //cin.tie(0);
    //cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    solve();
    return 0;
}
```

7.12 离散化

```
struct Hash {
    int b[N], tot;
    void init() {tot = 0;}
    void insert(int x) {b[++tot] = x;}
    void build() {
        sort(b+1, b+1+tot);
        tot = unique(b+1, b+tot+1) - (b+1);
    }
    int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
}ha;
```

7.13 莫队 + 树状数组

```
#include <bits/stdc++.h>
using namespace std;
#define ACM_LOCAL
const int N = 1e5 + 10;
int n, m, k, a[N], pos[N], b[N], upa[N], prea[N], sum[N], Ans, ans[N];

inline int read(){
   int s=0,w=1;
   char ch=getchar();
   while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
   while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();
   return s*w;
}

struct Q{
    int l, r, id;
}p[N];

bool cmp(Q x, Q y) {
    if (pos[x.l] == pos[y.l]) return x.r < y.r;
    else return pos[x.l] < pos[y.l];
}

inline int lowbit(int x) {return x & -x;}

void add(int p, int v) {for (int i = p; i <= 3*n+10; i += lowbit(i)) sum[i] += v;}

int query(int p) {
    int res = 0;
    for (int i = p; i > 0; i -= lowbit(i)) res += sum[i];
    return res;
}
int query_(int l, int r) {return query(r) - query(l-1);}

void solve() {
    n = read(), m = read(), k = read();
    int tot = 0;
    int siz = sqrt(n);
    for (int i = 1; i <= n; i++) a[i] = read();

    for (int i = 1; i <= n; i++) {
        upa[i] = a[i] + k, prea[i] = a[i] - k;
        b[++tot] = a[i], b[++tot] = upa[i], b[++tot] = prea[i];
    }
    sort(b + 1, b + tot + 1);
    int cnt = unique(b + 1, b + tot + 1) - b - 1;
    for (int i = 1; i <= n; i++) {
        a[i] = lower_bound(b+1, b+cnt+1, a[i]) - b;
        upa[i] = lower_bound(b+1, b+cnt+1, upa[i]) - b;
        prea[i] = lower_bound(b+1, b+cnt+1, prea[i]) - b;
    }

    for (int i = 1; i <= m; i++) {
        p[i].l = read(), p[i].r = read();
        p[i].id = i;
    }
    sort(p + 1, p + m + 1, cmp);
    int l = 1, r = 0;
    for (int i = 1; i <= m; i++) {
        while (p[i].l < l) {
            l--;
            Ans += query_(prea[l], upa[l]);
            add(a[l], 1);
        }
        while (p[i].l > l) {
            add(a[l], -1);
            Ans -= query_(prea[l], upa[l]);
            l++;
        }
        while (p[i].r < r) {
            add(a[r], -1);
            Ans -= query_(prea[r], upa[r]);
            r--;
        }
        while (p[i].r > r) {
            r++;
            Ans += query_(prea[r], upa[r]);
            add(a[r], 1);
        }
        ans[p[i].id] = Ans;
    }
    for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
//#else
    solve();
#endif
    return 0;
}
```

7.14 普通莫队

```
#include <bits/stdc++.h>
//#define ACM_LOCAL
```

```cpp
using namespace std;
const int N = 5e4 + 10;

inline int read() {
    int s = 0, w = 1;
    char ch = getchar();
    while (ch < '0' || ch>'9') { if (ch == '-')w = -1; ch = getchar(); }
    while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
    return s * w;
}

int pos[N], a[N], vis[2 * N];
int n, q, num, k;
long long res, ans[N];

struct Q {
    int l, r, id;
}p[N];

bool cmp(Q x, Q y) {
    if (pos[x.l] == pos[y.l]) return x.r < y.r;
    else return pos[x.l] < pos[y.l];
}

void add(int x) {

}

void sub(int x) {

}

void solve() {
    n = read(), q = read(), k = read();
    for (int i = 1; i <= n; i++) a[i] = read();

    int siz = sqrt(n);
    for (int i = 1; i <= n; i++) pos[i] = i / siz;

    for (int i = 1; i <= q; i++) {
        p[i].l = read(), p[i].r = read();
        p[i].id = i;
    }
    sort(p + 1, p + q + 1, cmp);

    int l = 1, r = 0;
    for (int i = 1; i <= q; i++) {
        while (p[i].l < l) add(--l);
        while (p[i].r > r) add(++r);
        while (p[i].l > l) sub(l++);
        while (p[i].r < r) sub(r--);
        ans[p[i].id] = res;
    }
    for (int i = 1; i <= q; i++) {
        printf("%lld\n", ans[i]);
    }
}
```

7.15 三维偏序问题（CDQ 分治）

```
#include <bits/stdc++.h>
using namespace std;
#define fi first
#define se second
#define re register
typedef long long ll;
typedef pair<int, int> PII;
typedef unsigned long long ull;
const int N = 5e5 + 10, M = 1e6 + 5, INF = 1e9;
const int MOD = 1e9 + 7;
int n, k, cnt[N], ans[N], f[N];
struct node {
    int a, b, c, cnt, id;
    bool operator < (const node &rhs) {
        if (a == rhs.a) {
            if (b == rhs.b) return c < rhs.c;
            return b < rhs.b;
        }
        return a < rhs.a;
    }
};
node p[N], q[N], tmp[N];
int lowbit(int x) {return x&-x;}
void add(int x, int c) {
    for (int i = x; i <= k; i += lowbit(i)) cnt[i] += c;
}
int query(int x) {
    int res = 0;
    for (int i = x; i > 0; i -= lowbit(i)) res += cnt[i];
    return res;
}
void clear(int x) {
    for (int i = x; i <= k; i += lowbit(i)) {
        if (cnt[i]) cnt[i] = 0;
        else break;
    }
}
void CDQ(int l, int r) {
    if (l == r) return;
    int mid = (l + r) >> 1;
    CDQ(l, mid);
    CDQ(mid+1, r);
    int t1 = l, t2 = mid+1;
    for (int i = l; i <= r; i++) {
        if ((t1 <= mid && q[t1].b <= q[t2].b) || t2 > r) {
            add(q[t1].c, q[t1].cnt);
            tmp[i] = q[t1++];
        } else {
            ans[q[t2].id] += query(q[t2].c);
            tmp[i] = q[t2++];
        }
    }
    for (int i = l; i <= r; i++) q[i] = tmp[i], clear(q[i].c);
}
void solve() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> p[i].a >> p[i].b >> p[i].c;
    sort(p+1, p+n+1);
    int tot = 0;
    q[++tot] = {p[1].a, p[1].b, p[1].c, 1, 1};
    for (int i = 2; i <= n; i++) {
        if (p[i].a == p[i-1].a && p[i].b == p[i-1].b && p[i].c == p[i-1].c) {
            q[tot].cnt++;
        } else {
            q[++tot] = {p[i].a, p[i].b, p[i].c, 1, tot};
        }
    }
    CDQ(1, tot);
    for (int i = 1; i <= tot; i++) {
        f[ans[q[i].id] + q[i].cnt - 1] += q[i].cnt;
    }
    for (int i = 0; i < n; i++) printf("%d\n", f[i]);
}

signed main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
#ifdef ACM_LOCAL
    freopen("input", "r", stdin);
    freopen("output", "w", stdout);
#endif
    solve();
    return 0;
}
```

7.16 树上莫队

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
inline int read() {
    int s = 0, w = 1;
    char ch = getchar();
    while (ch < '0' || ch>'9') { if (ch == '-')w = -1; ch = getchar(); }
    while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
    return s * w;
}
int n, m;
struct edge {
    int to, next;
}e[N<<1];

struct Q {
    int l, r, id, lca;
}p[N];

int h[N], cnt, tot, in[N], out[N], a[N], b[N], pos[N], son[N], rnk[N], siz[N], top[N],
    d[N], fa[N], res, used[N], ans[N], vis[N];

void add(int u, int v) {
    e[cnt].to = v;
    e[cnt].next = h[u];
    h[u] = cnt++;
}
```

```

bool cmp(Q x, Q y) {
    if (pos[x.l] == pos[y.l]) return x.r < y.r;
    else return pos[x.l] < pos[y.l];
}

void discrete() {
    sort(b + 1, b + n + 1);
    int num = unique(b + 1, b + n + 1) - b - 1;
    for (int i = 1; i <= n; i++) a[i] = lower_bound(b + 1, b + num + 1, a[i]) - b;
}

void dfs1(int u) {
    son[u] = -1;
    siz[u] = 1;
    in[u] = ++tot, rnk[tot] = u;
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (!d[v]) {
            d[v] = d[u] + 1;
            fa[v] = u;
            dfs1(v);
            siz[u] += siz[v];
            if (son[u] == -1 || siz[v] > siz[son[u]]) son[u] = v;
        }
    }
    out[u] = ++tot, rnk[tot] = u;
}

void dfs2(int u, int t) {
    top[u] = t;
    if (son[u] == -1) return;
    dfs2(son[u], t);
    for (int i = h[u]; ~i; i = e[i].next) {
        int v = e[i].to;
        if (v != son[u] && v != fa[u]) dfs2(v, v);
    }
}

int lca(int u, int v) {
    while (top[u] != top[v]) {
        if (d[top[u]] > d[top[v]])
            u = fa[top[u]];
        else
            v = fa[top[v]];
    }
    return d[u] > d[v] ? v : u;
}

void Add(int x) {
    if (++vis[a[x]] == 1) ++res;
}

void Sub(int x) {
    if (--vis[a[x]] == 0) --res;
}

void ADD(int x) {
    used[x] ? Sub(x) : Add(x);
    used[x] ^= 1;
}

int main() {
    n = read(), m = read();
    memset(h, -1, sizeof h);
    int siz = sqrt(n);
    for (int i = 1; i <= n; i++) a[i] = read(), b[i] = a[i];
    for (int i = 1; i <= 2 * n; i++) pos[i] = i / siz;
    discrete();
    for (int i = 1; i <= n - 1; i++) {
        int x, y;
        x = read(), y = read();
        add(x, y), add(y, x);
    }
    d[1] = 1;
    dfs1(1);
    dfs2(1, 1);
    for (int i = 1; i <= m; i++) {
        int x, y;
        x = read(), y = read();
        if (in[x] > in[y]) swap(x, y);
        int lca_ = lca(x, y);
        p[i].id = i;
        if (lca_ == x) p[i].l = in[x], p[i].r = in[y];
        else p[i].l = out[x], p[i].r = in[y], p[i].lca = lca_;
    }
    sort(p + 1, p + m + 1, cmp);

    int l = 1, r = 0;
    for (int i = 1; i <= m; i++) {
        while (p[i].l < l) ADD(rnk[--l]);
        while (p[i].r > r) ADD(rnk[++r]);
        while (p[i].l > l) ADD(rnk[l++]);
        while (p[i].r < r) ADD(rnk[r--]);
        if (p[i].lca) ADD(p[i].lca);
        ans[p[i].id] = res;
        if (p[i].lca) ADD(p[i].lca);
    }
    for (int i = 1; i <= m; i++) printf("%d\n", ans[i]);
}
```

8 字符串

```
//
// Created by acer on 2021/2/16.
//
//判断子串，不同子串个数，所有子串字典序第i大，最长公共子串

#include <cstring>
#include <iostream>
#include "string"


#define mem(x, i) memset(x,i,sizeof(x))
using namespace std;
const int MAXN = 1e6 + 10;
int id[MAXN<<1];
int visit[MAXN<<1];
int endpos[MAXN << 1];
int siz[MAXN << 1];
int len[MAXN << 1];
int ch[MAXN << 1][27];
int fa[MAXN << 1];
int last = 1;
int tot = 1;
int p;
int len1;
void add(int c) {
    p = last;
    last = ++tot;
    int np = last;
    siz[np] = 1;
    len[np] = len[p] + 1;
    for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
    if (!p) fa[np] = 1;
    else {
        int q = ch[p][c];
        if (len[q] == len[p] + 1) fa[np] = q;
        else {
            int nq = ++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            fa[nq] = fa[q];
            len[nq] = len[p] + 1;
            fa[np] = fa[q] = nq;
            for (; p && ch[p][c] == q; p = fa[p]) {
                ch[p][c] = nq;
            }
        }

    }

}
void getTuopu()
{
    for (int k = 1; k <= tot; ++k) {        //给每个点赋权值
        endpos[len[k]] ++;
    }
    for (int m = 1; m <= tot; ++m) {      //获得长度大于等于m的点的总个数，所以必然是不会相同的，
    故可以用来标记。
        endpos[m] += endpos[m-1];
    }
    for (int n = 1; n <= tot; ++n) {  //根据点出现顺序获得拓扑序
        id[endpos[len[n]]--] = n;
    }
}
char s[MAXN], s1[MAXN];

void doit()
{
    for (int i = tot; i >= 1; --i) {
       int  p = id[i];
        siz[fa[p]] += siz[p];
    }
}

int solve(int w) {
    int p = 1;
    int tmp = 0;
    int sum = 0;
    for (int i = 1; i <= len1+len1; ++i) {
        int v = s1[i]-'a';
        if (ch[p][v])
            p = ch[p][v],tmp++;
        else{
            while (p && !ch[p][v]) p = fa[p];
            if (!p) p = 1,tmp =0 ;
            else
                tmp = len[p]+1,p = ch[p][v];
        }
///siz[p]是[i-l+1,i]的出现次数,我们需要求的是[i-|t|+1,i]的
///直到parent树上的祖先是孩子的后缀,我们一直往上fa
///直到父亲的longest小于|t|

///fa即删前缀字符, ch加后缀字符, 因为在同一个endpos, 所以跳完之后即使长度不等, endpos所包含的个数也是一样的
        if(tmp >= len1)
        {
            while( len[fa[p]]>=len1 ) p = fa[p];      //fa的longest严格小于len1, 保证可以加
            if( visit[p]!=w )
                sum+=siz[p],visit[p] = w;

            tmp = len1;
        }
    }
    return sum;
}

int main() {
    scanf("%s", s + 1);
    int l = strlen(s + 1);
    for (int i = 1; i <= l; ++i) {
        add(s[i] - 'a');
    }
    getTuopu();
    doit();
    int n;
    cin >> n;
    for (int j = 1; j <= n; ++j)
    {

        scanf("%s",s1+1);
        len1 = strlen(s1+1);
        for (int i = 1; i <= len1; ++i) {
            s1[i+len1] = s1[i];
        }

        cout << solve(j) << endl;
    }
}
```

8.1 (SAM)k 长子串最大出现次数

```
//
// Created by acer on 2021/2/16.
//
//判断子串，不同子串个数，所有子串字典序第i大，最长公共子串

#include <cstring>
#include <iostream>
#include "string"


#define mem(x, i) memset(x,i,sizeof(x))
using namespace std;
const int MAXN = 3e5 + 10;

int len[MAXN << 1];int ch[MAXN << 1][27];int fa[MAXN << 1];int weig[MAXN << 1];
int last = 1;int tot = 1;int p;int size[MAXN<<1];
void add(int c) {
    p = last;
    last = ++tot;
    int np = last;
    size[np] = 1;          //np节点表示的后缀出现过几次
    len[np] = len[p] + 1;
    for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
    if (!p) fa[np] = 1;
    else {
        int q = ch[p][c];
        if (len[q] == len[p] + 1) fa[np] = q;
        else {
            int nq = ++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            fa[nq] = fa[q];
            len[nq] = len[p] + 1;
            fa[np] = fa[q] = nq;
            for (; p && ch[p][c] == q; p = fa[p]) {
                ch[p][c] = nq;
            }
        }

    }
}
char s[MAXN<<1];int id[MAXN<<1];int ans[MAXN<<1];
void getTuopu()
{
    for (int k = 1; k <= tot; ++k) {        //给每个点赋权值
        weig[len[k]] ++;
    }
    for (int m = 1; m <= tot; ++m) {     //获得长度小于等于m的点的总个数，所以必然是不会相同的，故可以用来标记。
        weig[m] += weig[m-1];
    }
    for (int n = 1; n <= tot; ++n) {  //根据点出现顺序获得拓扑序
        id[weig[len[n]]--] = n;
    }
}
int main() {
    scanf("%s", s + 1);
    int l = strlen(s+1);
    for (int i = 1; i <= l; ++i) {
        add(s[i]-'a');
    }
    getTuopu();
    for (int i = tot; i >= 1 ; --i) {
        size[fa[id[i]]] += size[id[i]];
        ans[len[id[i]]] = max(ans[len[id[i]]], size[id[i]]);
    }
    for(int i = tot; i >= 1; --i) ans[i] = max(ans[i], ans[i + 1]);
//    for(int i = 1; i <= l; ++i) printf("%d\n", ans[i]);
}
```

8.2 (SAM) 第 k 小子串

```
//
// Created by acer on 2021/2/16.
//
//判断子串，不同子串个数，所有子串字典序第i大，最长公共子串

#include <cstring>
#include <iostream>
#include "string"


#define mem(x, i) memset(x,i,sizeof(x))
using namespace std;
const int MAXN = 3e5 + 10;

int len[MAXN << 1];
int ch[MAXN << 1][27];
int fa[MAXN << 1];
int last = 1;
int tot = 1;
int p;

void add(int c) {
    p = last;
    last = ++tot;
    int np = last;
    len[np] = len[p] + 1;
    for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
    if (!p) fa[np] = 1;
    else {
        int q = ch[p][c];
        if (len[q] == len[p] + 1) fa[np] = q;
        else {
            int nq = ++tot;
            memcpy(ch[nq], ch[q], sizeof(ch[nq]));
            fa[nq] = fa[q];
            len[nq] = len[p] + 1;
            fa[np] = fa[q] = nq;
            for (; p && ch[p][c] == q; p = fa[p]) {
                ch[p][c] = nq;
            }
        }

    }
}

char s[MAXN];
int count[MAXN<<1];
void dfs(int k)
{
    if (count[k]) return;
    count[k] = 1;
    for (int i = 0; i <= 26; ++i) {          //此处是实现算法的关键，通过从a遍历到z来计算字典序
        if (!ch[k][i]) continue;
        dfs(ch[k][i]);
        count[k]+=count[ch[k][i]];       //很明显，count的意义就是字典序为k的子串的首字母
    }
}
int main() {
    scanf("%s", s + 1);
    int l = strlen(s + 1);
    for (int i = 1; i <= l; ++i) {
        add(s[i] - 'a');
    }
    int T;
    dfs(1);
    scanf("%d",&T);
    while (T--){
        int k;
        int now = 1;
        scanf("%d",&k);
        while (k)
        {
            if (!k) break;
            for (int i = 0; i <= 26; ++i) {
                if (ch[now][i]) {
                    if (count[ch[now][i]] >= k) {     //因为不能往前找，所以找到的第一个大于k的就肯定是属于答案的子串中
                        putchar(i+'a');
                        --k;
                        now = ch[now][i];
                        break;
                    } else
                        k-=count[ch[now][i]];       //这个字母包含的不够多
                }
            }

        }puts("");
    }
}
```

8.3 (SAM) 多串 lcs

```
//
// Created by acer on 2021/2/16.
//
//判断子串，不同子串个数，所有子串字典序第i大，最长公共子串

#include <cstring>
#include <iostream>
#include "string"


#define mem(x, i) memset(x,i,sizeof(x))
using namespace std;
const int MAXN = 1e5 + 10;
int mxlen[MAXN<<1];
int anslen[MAXN<<1];
int len[MAXN << 1];
int ch[MAXN << 1][27];
int fa[MAXN << 1];
int id[MAXN << 1];
int last = 1;
int tot = 1;

int p;
int visit[MAXN<<1];
void add(int c) {
    p = last;
    last = ++tot;
    int np = last;
    len[np] = len[p] + 1;
    for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
    if (!p) fa[np] = 1;
    else {
        int q = ch[p][c];
        if (len[q] == len[p] + 1) fa[np] = q;
        else {
            int nq = ++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            fa[nq] = fa[q];
            len[nq] = len[p] + 1;
            fa[np] = fa[q] = nq;
            for (; p && ch[p][c] == q; p = fa[p]) {
                ch[p][c] = nq;
            }
        }

    }

}
char s[MAXN], s1[MAXN];
int mx= 0;
void solve() {
    p = 1;
    int len1 = strlen(s1 + 1);
    int tmp = 0;
    for (int i = 1; i <= len1; ++i) {
        int c = s1[i] - 'a';
        if (ch[p][c]) {p = ch[p][c], tmp++;}
        else {
            while (p && !(ch[p][c])) p = fa[p];
            if (!p) p = 1, tmp = 0;
            else {tmp = len[p] + 1;p = ch[p][c];}
        }
        visit[p]++;
        mxlen[p] = max(mxlen[p],tmp);
    }
    for (int j = tot; j >= 1; --j) {
        int k = id[j];
        anslen[k] = min(anslen[k],mxlen[k]);
        if (anslen[k] && fa[k]) mxlen[fa[k]] = max(mxlen[fa[k]],len[fa[k]]);
        mxlen[k] = 0;
    }

}


int main() {
    scanf("%s", s + 1);
    int l = strlen(s + 1);
    for (int i = 1; i <= l; ++i) {
        add(s[i] - 'a');
    }
    for (int k = 1; k <= tot; ++k) {       //给每个点赋权值
        anslen[len[k]] ++;

    }
    for (int m = 1; m <= tot; ++m) {     //获得长度小于等于m的点的总个数，所以必然是不会相同的，故可以用来标记。
        anslen[m] +=anslen[m-1];
    }
    for (int n = 1; n <= tot; ++n) {  //根据点出现顺序获得拓扑序
        id[anslen[len[n]]--] = n;
    }
    while (scanf("%s", s1 + 1)!=EOF)
    {
        solve();
    }
    for (int j = 1; j <= tot; ++j) {
        mx = max(mx,anslen[j]);
    }
    printf("%d",mx);
}
```

8.4 ac 自动机

```

//多模式串匹配

#include <queue>
#include <cstdlib>
#include <cmath>
#include <cstdio>
#include <string>
#include <cstring>
#include <iostream>
#include <algorithm>
```

```
using namespace std;
const int maxn =  500000+9;
int trie[maxn][26]; //字典树
int cntword[maxn];  //记录该单词出现次数
int fail[maxn];     //失败时的回溯指针
int cnt = 0;
void insertWords(string s){
    int root = 0;
    for(int i=0;i<s.size();i++){
        int next = s[i] - 'a';
        if(!trie[root][next])
            trie[root][next] = ++cnt;
        root = trie[root][next];
    }
    cntword[root]++;
}
void getFail(){
    queue <int>q;
    for(int i=0;i<26;i++){      //将第二层所有出现了的字母扔进队列
        if(trie[0][i]){
            fail[trie[0][i]] = 0;
            q.push(trie[0][i]);
        }
    }
    while(!q.empty()){
        int now = q.front();
        q.pop();
        for(int i=0;i<26;i++){
            if(trie[now][i]){
                fail[trie[now][i]] = trie[fail[now]][i];
                q.push(trie[now][i]);
            }
            else trie[now][i] = trie[fail[now]][i];
        }
    }
}

int query(string s){
    int now = 0,ans = 0;
    for(int i=0;i<s.size();i++){
        now = trie[now][s[i]-'a'];
        for(int j=now;j && cntword[j]!=-1;j=fail[j]){
            ans += cntword[j];
            cntword[j] = -1;
        }
    }
    return ans;
}
void init()
{
    for(int i = 0 ; i<= cnt ; i++)
    {
        memset(trie[i], 0, sizeof(trie[i]));
    }
    memset(cntword,0,sizeof(cntword));
    cnt = 0;
}
```

8.5 kmp

```
//
// Created by acer on 2021/2/8.
//
#define int long long
const int maxn = 1000000;
int nxt[maxn];
void search(string s)
{
    int k=-1;
    nxt[0]=-1;
    int j=0;
    while(j<s.length())
    {
        if(k<0 || s[k]==s[j])
        {
            j++;
            k++;
            nxt[j] = k;
        }
        else {
            k=nxt[k];
        }

    }
}
```

8.6 二维字符串哈希

```
#include <iostream>
#include <algorithm>
#include <unordered_map>

using namespace std;

typedef unsigned long long ULL;

const int N = 1010, M = N * N, P = 131;

int n, m, a, b;
ULL hashv[N][N], p[M];
char str[N];

ULL calc(ULL f[], int l, int r)
{
    return f[r] - f[l - 1] * p[r - l + 1];
}

int main()
{
    scanf("%d%d%d%d", &n, &m, &a, &b);

    p[0] = 1;
    for (int i = 1; i <= n * m; i ++ ) p[i] = p[i - 1] * P;

    for (int i = 1; i <= n; i ++ )
    {
        scanf("%s", str + 1);
        for (int j = 1; j <= m; j ++ ) hashv[i][j] = hashv[i][j - 1] * P + str[j] - '0';
    }

    unordered_map<ULL, int> S;
    for (int i = b; i <= m; i ++ )
    {
        ULL s = 0;
        int l = i - b + 1, r = i;
        for (int j = 1; j <= n; j ++ )
        {
            s = s * p[b] + calc(hashv[j], l, r);
            if (j - a > 0) s -= calc(hashv[j - a], l, r) * p[a * b];
            if (j >= a) S[s] = 1;
        }
    }

    int Q;
    scanf("%d", &Q);
    while (Q -- )
    {
        ULL s = 0;
        for (int i = 0; i < a; i ++ )
        {
            scanf("%s", str);
            for (int j = 0; j < b; j ++ ) s = s * P + str[j] - '0';
        }
        if (S[s]) puts("1");
        else puts("0");
    }

    return 0;
}
```

8.7 后缀自动机

```
//
// Created by acer on 2021/2/16.
//
//判断子串，不同子串个数，所有子串字典序第i大，最长公共子串

#include <cstring>
#include <iostream>
#include "string"


#define mem(x, i) memset(x,i,sizeof(x))
using namespace std;
const int MAXN = 3e5 + 10;

int len[MAXN << 1];
int ch[MAXN << 1][27];
int fa[MAXN << 1];
int last = 1;
int tot = 1;
int p;

```

```
void add(int c) {
    p = last;
    last = ++tot;
    int np = last;
    len[np] = len[p] + 1;
    for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
    if (!p) fa[np] = 1;
    else {
        int q = ch[p][c];
        if (len[q] == len[p] + 1) fa[np] = q;
        else {
            int nq = ++tot;
            memcpy(ch[nq],ch[q],sizeof(ch[nq]));
            fa[nq] = fa[q];
            len[nq] = len[p] + 1;
            fa[np] = fa[q] = nq;
            for (; p && ch[p][c] == q; p = fa[p]) {
                ch[p][c] = nq;
            }
        }

    }

}

char s[MAXN], s1[MAXN];

int solve() {
    int len1 = strlen(s1 + 1);
    int ans =0 ; int tmp = 0;
    for (int i = 1; i <= len1; ++i) {

        int c = s1[i] - 'a';
        if (ch[p][c]) p = ch[p][c], tmp++;
        else {
            while (p && !(ch[p][c]))
                p = fa[p];
            if (!p) p = 1, tmp = 0;
            else {
                tmp = len[p] + 1;
                p = ch[p][c];
            }
        }
        ans = max(ans,tmp);
    }
    return ans;
}
int id[MAXN<<1];
int weig[MAXN << 1];
void getTuopu()
{
    for (int k = 1; k <= tot; ++k) {        //给每个点赋权值
        weig[len[k]] ++;
    }
    for (int m = 1; m <= tot; ++m) {      //获得长度大于等于m的点的总个数，所以必然是不会相同的，故可以用来标记。
        weig[m] += weig[m-1];
    }
    for (int n = 1; n <= tot; ++n) {  //根据点出现顺序获得拓扑序
```

```
        id[weig[len[n]]--] = n;
//        cout << weig[len[n]]+1 <<' ' << id[weig[len[n]]+1] << endl;
    }
//    for (int j = 1; j <= tot; ++j) {
//        cout << id[j] << endl;
//    }
}
int main() {
//    freopen("in.txt","r",stdin);
//    freopen("out.txt","w",stdout);
    scanf("%s", s + 1);
    int l = strlen(s + 1);
    for (int i = 1; i <= l; ++i) {
        add(s[i] - 'a');
    }
    scanf("%s", s1 + 1);
    cout << solve();
}
```

8.8 回文自动机

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 500000 , MAXK = 26;
char str[ MAXN + 5 ];long long weight[MAXN+5];
struct Palindrome_Automaton{
    int Size //总节点
    , Last , Root0 , Root1 , Trans[ MAXN + 5 ][ MAXK + 5 ] , Fail[ MAXN + 5 ];
    long long  Len[ MAXN + 5 ];

    Palindrome_Automaton( ) {
        Root0 = Size ++ , Root1 = Size; Last = Root1;
        Len[ Root0 ] = 0  , Fail[ Root0 ] = Root1;
        Len[ Root1 ] = -1 , Fail[ Root1 ] = Root1;
    }
    void Extend( int ch , int dex ) {
        int u = Last;
        while (str[ dex - Len[ u ] - 1 ] != str[ dex ] ) u = Fail[ u ]; //找到合格的后缀
        if( !Trans[ u ][ ch ] ) {         //无现成的边
            int Newnode = ++ Size , v = Fail[ u ];    //防止取掉整串
            Len[ Newnode ] = Len[ u ] + 2;
            while (str[ dex - Len[ v ] - 1 ] != str[ dex ] ) v = Fail[ v ];    //给他找一个fail指针
            Fail[ Newnode ] = Trans[ v ][ ch ] , Trans[ u ][ ch ] = Newnode;
        }
        Last = Trans[ u ][ ch ];
        weight[Last]++;
    }

    void Build( char *str ) {
        int len = strlen( str );
        for( int i = 0 ; i < len ; i ++ ) {
            Extend( str[ i ] - 'a' + 1 , i );
        }
    }

    void getWeight()
    {
        long long  ma =0 ;
        for (int k = Size; k >= 0; --k) {
            weight[Fail[k]] += weight[k];
        }
        for (int j = 0; j <= Size; ++j) {
            ma = max(ma,Len[j]*weight[j]);
        }
        cout << ma << endl;
    }

}PAM;

signed main( ) {
    scanf("%s", str );
    PAM.Build( str );
    PAM.getWeight();
    return 0;
}
```

8.9 拓展 kmp

```
#include<bits/stdc++.h>
#include "ext/pb_ds/assoc_container.hpp"
//using namespace __gnu_pbds;
using namespace std;
const int N = 1e4 + 10;
//typedef long long ll;
#define int long long
using namespace std;
const int MAXN = 2e7 + 5;
char p[MAXN], s[MAXN];
int pl, sl, z[MAXN], ext[MAXN];
//z[i]:     B串   i  ~  strlen （B） -1      部分与 B自身的最长相同前缀
//
//ext[i]:      A串   i  ~  strlen （A） -1        部分与B的最长相同前缀，也就是我们要求的东西。
void getZ() {
    z[0] = pl;//从0号位置开始，LCP就是全部字符串
    //从1开始，先暴力算
    int now = 0;
    while (now + 1 < pl && p[now] == p[now + 1]) now++;
    z[1] = now;
    int p0 = 1;
    //p0是最远情况的起点
    for (int i = 2; i < pl; ++i) {
        //p0+z[p0]是此时最远处
        //i-p0对应着主串的i,加i就是能到的距离
        if (i + z[i - p0] < p0 + z[p0]) {
            z[i] = z[i - p0];//第一种情况
        } else {
            now = p0 + z[p0] - i;
            now = max(now, 0ll);
            while (now + i < pl && p[now] == p[now + i]) now++;
            z[i] = now;
            p0 = i;
        }
    }
}

```

```
void exkmp() {
    getZ();
    //先暴力算ext[0]
    int now = 0;
    while (now < pl && now < sl && p[now] == s[now]) now++;
    ext[0] = now;
    int p0 = 0;
    for (int i = 1; i < sl; ++i) {
        if (i + z[i - p0] < p0 + ext[p0]) {
            ext[i] = z[i - p0];
        } else {
            now = p0 + ext[p0] - i;
            now = max(now, 0ll);//防止i太大
            while (now < pl && now + i < sl && p[now] == s[now + i]) now++;
            ext[i] = now;
            p0 = i;
        }
    }
}

signed main() {
    scanf("%s%s", s, p);
    pl = strlen(p);
    sl = strlen(s);
    exkmp();
    int ans1 = 0,ans2=0;
    for (int i = 0; i < pl; ++i) {
        ans1 ^= 1LL * (i + 1) * (z[i] + 1);
//         cout << z[i] << ' ';
    }
//    cout << '\n';
    for (int i = 0; i < sl; ++i) {
        ans2 ^= 1LL * (i + 1) * (ext[i] + 1);
//        cout << ext[i] << ' ';
    }

    printf("%lld\n%lld\n", ans1, ans2);
    return 0;
}
```

8.10 字典树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100010;
int tire[26*N][26], n, m;
int cnt, level[N*26];

void insert(string s) {
    int p = 0;
    for (int i = 0; i < s.length(); i ++) {
        int v = s[i] - 'a';
        if (!tire[p][v]) tire[p][v] = ++cnt;
        p = tire[p][v];
    }
    level[p] ++;
}

```

```
int query(string s) {
    int p = 0;
    int res = 0;
    for (int i = 0; i < s.length(); i ++) {
        p = tire[p][s[i]-'a'];
        if (!p) break;
        res += level[p];
    }
    return res;
}
```

8.11 字符串哈希

```

char s[N];
unsigned long long f[N], p[N];

unsigned long long get(int l, int r) {//获取哈希值
    return f[r] - f[l-1] * p[r-l+1];
}

f[i] = f[i-1] * base + s[i];
p[i] = p[i-1] * base;


/*

H(T) = H(S + T) - H(S) - p ^ (length(T))

*/
```

8.12 最小表示法

```
//
// Created by acer on 2021/2/1.
//
//求循环字符串的长度为k的最小子串
int MinimumRepresentation(int *s, int l)
{
    int i,j,k;
    i=0;j=1;k=0;
    while(i<l&&j<l)
    {
        k=0;
        while(s[i+k]==s[j+k]&&k<l) k++;
        if(k==l) return i;
        if(s[i+k]>s[j+k])
            if(i+k+1>j) i=i+k+1;
            else i=j+1;
        else if(j+k+1>i) j=j+k+1;
        else  j=i+1;
    }
    if(i<l) return i;
    else return j;
}
```