IUBAT enCRYpted

International University of Business Agriculture and Technology

May 6, 2025

Contents

1 A.Settings and Script

1.1 Generator

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define accuracy
chrono::steady_clock::now().
time_since_epoch().count()
#define rep(i, a, n) for (int i = a; i <= n; ++i)
#define nl          << "\n"
const int N = 1e6 + 4;
int32_t permutation[N];
mt19937 rng(accuracy);
int rand(int l, int r)
{
    uniform_int_distribution<int> ludo(l, r);
    return ludo(rng);
}
const int inf = 1LL << 31;
using pii = pair<int, int>;
namespace generator
{
string gen_string(int len = 0, bool upperCase =
    false, int l = 1, int r = 26)
{
    assert(len >= 0 && len <= 5e6);
    string str(len, (upperCase ? 'A' : 'a'));
    for (char &ch : str)
    {
        ch += rand(l, r) - 1;
    }
    return str;
}
vector<int> gen_array(int len = 0, int minRange =
    0, int maxRange = inf)
{
    assert(len >= 0 and len <= 5e6);
    vector<int> a(len);
    for (int &x : a)
        x = rand(minRange, maxRange);
    return a;
}
}
using namespace generator;

template <typename T = int> ostream
    &operator<<(ostream &other, const vector<T>
    &v)
{
    for (const T &x : v)
        other << x << ' ';
    other << '\n';
    return other;
}
#define SINGLE_TEST
const int max_tests = 1;
void generate_test()
{
}
signed main()
{
    srand(accuracy);
    int t = 1;
    #ifndef SINGLE_TEST
        t = rand(1, max_tests), cout << t << '\n';
    #endif
    while (t--)
    {
        generate_test();
    }
}
```

1.2 Main Script

```
set -e
g++ code.cpp -o code
g++ gen.cpp -o gen
g++ brute.cpp -o brute
for((i = 1; ; ++i)); do
    ./gen $i > input_file
    ./code < input_file > myAnswer
    ./brute < input_file > correctAnswer
    diff -Z myAnswer correctAnswer > /dev/null ||
        break
    echo "Passed test: " $i
done
echo "WA on the following test:"
cat input_file
echo "Your answer is:"
cat myAnswer
echo "Correct answer is:"
cat correctAnswer
```

1.3 Settings

```
// Layout -> Two column split row
// Wrap -> On, wrapping indent -> true
// Autosave -> After Delay
// Sticky scroll -> off
// gedit ~/.bashrc -> ulimit -s 2000123

// Command lines:
// generator (bash):
//     chmod +x test.sh
//     ./test.sh

// C++:
//     g++ -std=c++17 -O2 -Wno-unused-result
    -static a.cpp -o x
```

```
//    ./x
//    g++ a.cpp -o x
//    ./x
```

1.4 Template

```
#include "bits/stdc++.h"
#pragma GCC target ("avx2")
#pragma GCC optimization ("O3")
#pragma GCC optimization ("unroll-loops")
using namespace std;
typedef long long int ll;
typedef unsigned long long ll;
typedef double db;

#define pi acos(-1)
#define prDouble(x, y) fixed << setprecision(y)
    << x
#define lcm(a, b) (a / __gcd(a, b)) * b
#define fast_io                    \
    ios_base::sync_with_stdio(false); \
    cin.tie(NULL);                 \
    cout.tie(NULL)
#define nl << "\n"
#define debug(x) cout << #x << " = " << x nl
#define casePrint(ans, cn) cout << "Case " << cn
    << ": " << ans nl

int getbit(long long n, long long bit) {
    return (1LL << bit) & n;
}
void setbit(long long &n, long long bit) {
    n |= (1LL << bit);
}
void unsetbit(long long &n, long long bit) {
    n &= ~(1LL << bit);
}

void solve(int caseNumber) {

}
```

```
void init_code() {
#ifndef ONLINE_JUDGE
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    freopen("error.txt", "w", stderr);
#endif
}

signed main() {
    init_code();
    fast_io;
    int testcase = 1;
    // cin >> testcase;
    for (int caseNumber = 1; caseNumber <=
        testcase; ++caseNumber) {
        solve(caseNumber);
    }
    return 0;
}
```

2 B.Combinatorics

2.1 Binomial coefficient

```
const int N = 1010, mod = 1e9 + 7;
int C[N][N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  C[0][0] = 1;
  for (int n = 1; n < N; n++) {
    C[n][0] = 1;
    for (int k = 1; k <= n; k++) {
      C[n][k] = (C[n - 1][k - 1] + C[n - 1][k]) %
          mod;
    }
  }
  cout << C[6][2] << '\n';
  return 0;
}
```

2.2 Extended Euclidean Algorithm

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

2.3 General Equation

- $\sum_{0 \le k \le n} \binom{n-k}{k} = Fib_{n+1}$
- $\binom{n}{k} = \binom{n}{n-k}$
- $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$
- $4 \cdot k\binom{n}{k} = n\binom{n-1}{k-1}$
- $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$
- $\sum_{i=0}^{n} \binom{n}{i} = 2^n$
- $\sum_{i \ge 0} \binom{n}{2i} = 2^{n-1}$
- $\sum_{i \ge 0} \binom{n}{2i+1} = 2^{n-1}$
- $\sum_{i=0}^{k} (-1)^i \binom{n}{i} = (-1)^k \binom{n-1}{k}$
- $\sum_{i=0}^{k} \binom{n+i}{i} = \binom{n+k+1}{k}$
- $1\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + ... + n\binom{n}{n} = n2^{n-1}$
- $\sum_{i=0}^{n} C(n,i) = 2^n$
- $\sum_{i \ge 0} C(n,2i) = 2^{n-1}$
- $\sum_{i \ge 0} C(n,2i+1) = 2^{n-1}$

2.4 How Many Digit X^Y

Let, No. of Digits D. D
=floor $[log_{10}(X^Y)] + 1$
=floor $[Y \times log_{10}(X)] + 1$

2.5 How many digit in N!

Let, Number of Digits D
D=floor$[log_{10}(N!)] + 1$
=floor$[log_{10}(1 \times 2 \times 3 \times 4 \ldots\ldots\ldots\ldots (N-1) \times N)] + 1$
=floor$[log_{10}(1) + log_{10}(2) + \ldots\ldots\ldots\ldots\ldots +$
$log_{10}(N)] + 1$

2.6 Last Non Zero Digit of Factorial

```
int PTwo(int N) {
    int T[]= {6,2,4,8};
     if (N==0) return 1;
    return T[N%4];
}
int LastNZDigit(int N) {
    int A[]= {1,1,2,6,4};
    if(N<5) return A[N];
    return
        (PTwo(N/5)*LastNZDigit(N/5)*LastNZDigit(N%5))%10;
}
```

2.7 Matrix Exponentitation

```
#define ROF(i,a,b)    for(int i=a;i>=b;i--)
#define REP(i,b)      for(int i=0;i<b;i++)
LL mod;
const LL N=6;
void MatMul(LL A[N][N], LL B[N][N]) {
    LL R[N][N];
    MEM(R,0);
    REP(i, N) REP(j, N) REP(k, N) R[i][j] =
        (R[i][j]%mod + (A[i][k] *
        B[k][j])%mod)%mod;
    REP(i, N) REP(j, N) B[i][j] = R[i][j];
    return;
}
void MatPow(LL R[N][N],LL M[N][N],LL P) {
    while(P) {
        if(P & 1) MatMul(M,R);
        MatMul(M,M);
        P = P >> 1;
    }
}
int main() {
    LL n,M[N][N],R[N][N]; // M is Co-efficient
        Matrix,R is Base case Matrix
    //Take input values of M and R matrix
    //Input n,We have to find f(n)
    MatPow(R,M,n-2); // Here n-2 may changes in
        diffrent problems
    //value of f(n) is in R[0][0] position
    return 0;
}
```

2.8 Modular Inverse

```
int binExpIter(int a, int b)
{
    int ans = 1;
    while (b)
    {
        if (b & 1)
            ans = (ans * 1LL * a) % MOD;
        a = (a * 1LL * a) % MOD;
        b >>= 1;
    }
    return ans;
}

int MMI(int n)
{
    return binExpIter(n, MOD - 2);
}
```

2.9 combination1

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

const int N = 1e6, mod = 1e9 + 7;

int power(long long n, long long k) {
  int ans = 1 % mod; n %= mod; if (n < 0) n +=
      mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}
int f[N], invf[N];
int nCr(int n, int r) {
  if (n < r or n < 0) return 0;
  return 1LL * f[n] * invf[r] % mod * invf[n - r]
      % mod;
}
int nPr(int n, int r) {
  if (n < r or n < 0) return 0;
  return 1LL * f[n] * invf[n - r] % mod;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  f[0] = 1;
  for (int i = 1; i < N; i++) {
    f[i] = 1LL * i * f[i - 1] % mod;
  }
  invf[N - 1] = power(f[N - 1], mod - 2);
  for (int i = N - 2; i >= 0; i--) {
    invf[i] = 1LL * invf[i + 1] * (i + 1) % mod;
  }
  cout << nCr(6, 2) << '\n';
  cout << nPr(6, 2) << '\n';
  return 0;
}
```

2.10 nCr with Big Mod

```
ll Big_Mod(ll a, ll b)
{
    if(!b) return 1;
    ll ans = Big_Mod(a, b/2);
    ans %= mod;ans *= ans;ans %= mod;
    if(b&1)
    {
        a %= mod;ans *= a;ans %= mod;
    }
    return ans;
}
ll nCr(ll n, ll k)
{
    return fact(n)*1ll*Big_Mod(fact(k),
        mod-2)%mod*1ll*Big_Mod(fact(n-k),
        mod-2)%mod;
}
```

2.11 star and bar

1. $x_1 + x_2 + \cdots + x_r = n$ where $x_i >$ 0. Solution: $\binom{n-1}{r-1}$
2. $x_1 + x_2 + \cdots + x_r = n$ where $x_i \geq$ 0. Solution: $\binom{n+r-1}{r-1}$
3. $x_1 + x_2 + \cdots + x_r = n$ where $x_i \geq$ *b*. Solution: $\binom{n-rb+r-1}{r-1}$

3 C.Geometry

3.1 Line And Point

```
struct pt {
    double x, y;
};
struct line {
    double a, b, c;
};
const double EPS = 1e-9;
// determinant of a 2x2 matrix
double det(double a, double b, double c, double
     d) {
    return a*d - b*c;
}
bool intersect(line m, line n, pt & res) {
    double zn = det(m.a, m.b, n.a, n.b);
    if (abs(zn) < EPS)
        return false;
    res.x = -det(m.c, m.b, n.c, n.b) / zn;
    res.y = -det(m.a, m.c, n.a, n.c) / zn;
    return true;
}
bool parallel(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b)) < EPS;
}
bool equivalent(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b)) < EPS
          && abs(det(m.a, m.c, n.a, n.c)) < EPS
          && abs(det(m.b, m.c, n.b, n.c)) < EPS;
}
```

3.2 Points Inside Polygon

```
//points inside convex polygon O(logn)
const ll N = 100009;
struct point {
    ll x,y;
}a[N];
ll n;
double cross(const point& p1,const point&
     p2,const point& org) {
    return
        ((p1.x-org.x)*1.0)*(p2.y-org.y)-((p2.x-org.x)*1.0)*(p1.y-org.y);
}
inline bool comp(const point& x,const point& y) {
    return cross(x,y,a[0]) >= 0;
}
bool inside(point& p) {
    if (cross(a[0], a[n-1], p)>=0) return false;
    if (cross(a[0], a[1], p) <=0) return false;
    ll l =1;
    ll r =n-1;
    while(l<r) {
        ll m = l + (r-l)/2;
        if(cross(a[m],p,a[0]) >=0)
            l=m+1;
        else
            r=m;
    }
    if(l == 0)
        return false;
    return cross(a[l-1],a[l],p) >0;
}
sort(a+1,a+n,comp);
```

3.3 Solid-Equations

1. Rectangular Parallelepiped

- Area: $2(ab + bc + ca)$
- Volume: abc
- Diagonal: $\sqrt{a^2 + b^2 + c^2}$

2. Cube

- Total surface area: $6a^2$
- Volume: a^3
- Diagonal: $\sqrt{3} \cdot a$

3. Prism

- Total surface area: 2(base area) + (perimeter of base) × (height)
- Volume: (base area) × (height)

4. Pyramid

- Total surface area: (base area) + $\frac{1}{2}$ × (perimeter of base) × (slant height) (for a right regular pyramid)
- Slant height: $\sqrt{h^2 + r^2}$ (for a right regular pyramid with base inscribed in a circle of radius r)
- Volume: $\frac{1}{3}$ × (base area) × (height)

5. Right Circular Cone

- Surface area: $\pi r(l + r)$
- Volume: $\frac{1}{3}$ × (base area) × (height)

6. Sphere

- Surface area: $4\pi r^2$
- Volume: $\frac{4}{3}\pi r^3$
- Radius of the circle formed by a section at a distance h from the center: $r_h = \sqrt{r^2 - h^2}$

7. Cylinder

- Total surface area: $2\pi r(r + h)$
- Curved surface area: $2\pi rh$
- Volume: $\pi r^2 h$

8. Cone

- Curved surface area: πrl
- Total surface area: $\pi r(r + l)$
- Volume: $\frac{1}{3}\pi r^2 h$

9. Frustum of a Cone

- Curved surface area: $\pi(r_1 + r_2)l$
- Total surface area: $\pi(r_1 + r_2)l + \pi r_1^2 + \pi r_2^2$
- Volume: $\frac{1}{3}\pi h(r_1^2 + r_2^2 + r_1 r_2)$

10. Hemisphere

- Curved surface area: $2\pi r^2$
- Total surface area: $3\pi r^2$
- Volume: $\frac{2}{3}\pi r^3$

11. Torus

- Surface area: $4\pi^2 Rr$
- Volume: $2\pi^2 Rr^2$

12. Ellipsoid

- Surface area: (Complex formula, approximation: $4\pi\left(\frac{a^p b^p + a^p c^p + b^p c^p}{3}\right)^{1/p}$, where $p \approx 1.6075$)
- Volume: $\frac{4}{3}\pi abc$

---

4 D.Number Theory

4.1 1 - N Divisor

```
// Number of Divisor
for (int i = 1; i * i <= n; i++) {
    for (int j = i * i; j <= n; j += i) {
        if ( i * i == j) a[j]++;
        else a[j] += 2;
    }
}
// Note: When we find the remainder of a % m, the
    answer will their GCD (Greatest Common
    Divisor). So, a % m = GCD (a, m)
```

```
// Sum of Divisor
for ( int i = 1; i * i <= n; i++) {
    for (int j = i*i; j < n; j += i) {
        if(j == i*i) a[j] += i;
        else a[j] += i + (j / i);
    }
}

long long SumOfDivisors(long long num) {
    long long total = 1;

    for (int i = 2; (long long)i * i <= num; i++)
        {
        if (num % i == 0) {
            int e = 0;
            do {
                e++;
                num /= i;
            } while (num % i == 0);

            long long sum = 0, pow = 1;
            do {
                sum += pow;
                pow *= i;
            } while (e-- > 0);
            total *= sum;
        }
    }
    if (num > 1) {
        total *= (1 + num);
    }
    return total;
}

long long numberOfDivisors(long long num) {
    long long total = 1;
    for (int i = 2; (long long)i * i <= num; i++)
        {
        if (num % i == 0) {
            int e = 0;
            do {
                e++;
                num /= i;
            } while (num % i == 0);
```

```
            total *= e + 1;
        }
    }
    if (num > 1) {
        total *= 2;
    }
    return total;
}
```

4.2 Catalan Numbers

The Catalan sequence is the sequence $C_0, C_1, C_2, \ldots$, where $C_0 = 1$, $C_1 = 1$, and

$$C_n = C_0C_{n-1} + C_1C_{n-2} + \ldots + C_{n-1}C_0, \quad n \geq 2.$$

Therefore,

$$C_n = \frac{C(2n, n)}{n+1}, \quad n = 0, 1, 2, \ldots;$$

or

$$C_n = \frac{4n-2}{n+1} \times C_{n-1}, \quad n > 1.$$

The Catalan sequence is a frequent counting sequence. For example:

- C_n is the number of stack-sortable permutations of $\{1, \ldots, n\}$.
- C_n is the number of different ways that a convex polygon with $n + 2$ sides can be cut into triangles by connecting vertices with non-crossing line segments.
- C_n is the number of rooted binary trees with n nodes.

4.3 Eular Totient Of Every Number 1-N

```
int euler_phi [mxm];
void Euler_Totient (int n) {
    // Euler Totient of number 1 to n
        euler_phi[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(euler_phi[i] > 0) continue;
        for(int j = i + i; j <= n; j += i) {
            if(euler_phi[j] == 0) euler_phi[j] = j;
            euler_ph[j] -= (euler_phi[j] / i);
        }
    }
}
// Note: Normally euler totient returns the
     amount of number which are co-prime with n.
```

4.4 GCD LCM Equations

- $\gcd(a, 0) = a$
- $\gcd(a, b) = \gcd(b, a \mod b)$
- Every common divisor of a and b is a divisor of $\gcd(a, b)$.
- If m is any integer, then $\gcd(a+mb, b) = \gcd(a, b)$
- The gcd is a multiplicative function in the following sense: if a_1 and a_2 are relatively prime, then $\gcd(a_1a_2, b) = \gcd(a_1, b)\gcd(a_2, b)$.
- $\gcd(a, b) \cdot \mathrm{lcm}[a, b] = |ab|$
- $\gcd(a, \mathrm{lcm}[b, c]) = \mathrm{lcm}[\gcd(a, b), \gcd(a, c)]$
- $\mathrm{lcm}[a, \gcd(b, c)] = \gcd(\mathrm{lcm}[a, b], \mathrm{lcm}[a, c)]$
- For non-negative integers a and b, where a and b are not both zero,

$$\gcd(n1, n1) = n\gcd(a, b)$$

- $\gcd(a, b) = \frac{kla \cdot kib}{k}$
- $\gcd(i, n) = k] = \frac{\varphi(n)}{k} \sum_{i=1}^{n} \alpha(i)$
- $\gcd(k, n) = \sum_{d|n} \alpha(d)\varphi(\frac{n}{d})$
- $\sum_{k=1}^{n} \frac{1}{gcd(k,n)} = \sum_{d|n} \frac{1}{d} \cdot \phi(\frac{n}{d}) = \frac{1}{n} \sum_{d|n} d \cdot \phi(d)$
- $\sum_{k=1}^{n} \frac{k}{gcd(k,n)} = \frac{n}{2} \cdot \sum_{d|n} \frac{1}{d} \cdot \phi(\frac{n}{d}) = \frac{n}{2} \cdot \frac{1}{n} \cdot \sum_{d|n} d \cdot \phi(d)$
- $\sum_{k=1}^{n} \frac{n}{gcd(k,n)} = 2 * \sum_{k=1}^{n} \frac{k}{gcd(k,n)} - 1, for\ n > 1$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} [gcd(i, j) = 1] = \sum_{d=1}^{n} \mu(d) \lfloor \frac{n}{d} \rfloor^2$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} gcd(i, j) = \sum_{d=1}^{n} \phi(d) \lfloor \frac{n}{d} \rfloor^2$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} i \cdot j[gcd(i, j) = 1] = \sum_{i=1}^{n} \phi(i)i^2$
- $F(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i, j) = \sum_{l=1}^{n} (\frac{(1+\lfloor \frac{n}{l} \rfloor)(\lfloor \frac{n}{l} \rfloor)}{2})^2 \sum_{d|l} \mu(d)ld$
- $gcd(lcm(a, b), lcm(b, c), lcm(a, c)) = lcm(gcd(a, b), gcd(b, c), gcd(a, c))$
- $gcd(A_L, A_{L+1}, ..., A_R) = gcd(A_L, A_{L+1} - A_L, ..., A_R - A_{R-1})$
- $Givenn, IfSUM = LCM(1, n) + LCM(2, n) + ... + LCM(n, n)\ thenSUM = n2((d)xd) + 1$
- $xed(k) = x.()$

4.5 LOG with any base

```
// find log with any base
double LOG(double base, double num)
{
    return log(num) / log(base);
}
```

4.6 Large-number-fibonacci

```
const ll M = 1e9 + 7;
#define vpll vector<pair<ll, ll>>
#define pll pair<ll, ll>
vector<pair<ll, ll>> base;
vpll mul(vpll a, vpll b) {
  vpll c;
  ll a1 = (a[0].first * b[0].first + a[0].second
      * b[1].first) % M;
  ll a2 = (a[0].first * b[0].second + a[0].second
      * b[1].second) % M;
  ll b1 = (a[1].first * b[0].first + a[1].second
      * b[1].first) % M;
  ll b2 = (a[1].first * b[0].second + a[1].second
      * b[1].second) % M;
  pll x = {a1, a2};
  pll y = {b1, b2};

  c.push_back(x);
  c.push_back(y);

  return c;
}
vpll power(vpll a, ll b) {
  if (b == 1) return base;

  vpll r = power(a, b / 2);

  r = mul(r, r);

  if (b % 2) r = mul(r, a);

  return r;
}
void fib(vpll vp) { cout << vp[0].first +
    vp[0].second << endl; }
int main() {
  ll n;
  cin >> n;

  base.push_back({1, 1});
  base.push_back({1, 0});

  vpll ans = power(base, n - 2);

  fib(ans);

  return 0;
}
```

4.7 Sieve of Eratosthenes

```
// remember to call the function
vector<bool> prime(1e7+10, true);

void SieveOfEratosthenes(){
    prime[0]=prime[1]=false;
    for (int p = 2; p * p <= 1e7+10; p++) {
        if (prime[p] == true) {
            for (int i = p * p; i <= 1e7+10; i +=
                p)
                prime[i] = false;
        }
    }
}
```

4.8 Total Digit of N factorial

5 Data Structures

5.1 MO's algo coordinate compression

```
vector<int>compressed, a;
const int N = 2e5 + 8;
vector<int>cnt(N, 0);
int cur_result = 0;
struct query {
    int l, r, idx;
};
int block;
void add(int i) {
    cnt[a[i]]++;
    if(cnt[a[i]] == 1) {
        cur_result++;
    }
}
void Remove(int i) {
    cnt[a[i]]--;
    if(cnt[a[i]] == 0) {
        cur_result--;
    }

}
bool comp1(query p , query q) {
    if (p.l / block != q.l / block) {
        if (p.l == q.l) return p.r < q.r;
        return p.l < q.l;
    }
    return (p.l / block & 1) ? (p.r < q.r) : (p.r
        > q.r);
}
void mos_algorithm(int n, vector<query>&queries) {
    vector<int>answers(queries.size());
    block = (int)sqrt(n);
    sort(queries.begin(), queries.end(), comp1);
    int cur_l = 0;
    int cur_r = -1;
    for (query q : queries) {
        while (cur_l > q.l) {
            cur_l--;
            add(cur_l);
        }
        while (cur_r < q.r) {
            cur_r++;
            add(cur_r);
        }
        while (cur_l < q.l) {
            Remove(cur_l);
            cur_l++;
        }
        while (cur_r > q.r) {
            Remove(cur_r);
            cur_r--;
        }
        answers[q.idx] = cur_result;
```

```
    }
    for (int i : answers) {
        cout << i << nl;
    }
}
void solve() {
    int n, q; cin >> n >> q;
    a.assign(n, 0);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    vector<query>queries(q);
    for(int i = 0; i < q; i++) {
        cin >> queries[i].l;
        queries[i].l--;
        cin >> queries[i].r;
        queries[i].r--;
        queries[i].idx = i;
    }
    compressed = a;
    sort(compressed.begin(), compressed.end());
    compressed.resize(unique(compressed.begin(),
        compressed.end()) - compressed.begin());
    for (int i = 0; i < n; i++) {
        a[i] = lower_bound(compressed.begin(),
            compressed.end(), a[i]) -
            compressed.begin() + 1;
    }
    mos_algorithm(n, queries);
}
int main() {
    fast_IO;
    int t = 1;
    // cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

5.2 Segment Tree Lazy

```
struct node
{
  ll sum;
  ll setAll;
  ll increment;
  bool setAllValid;
  node()
  {
    sum = 0;
    setAllValid = 0;
    increment = 0;
  }
  void Reset()
  {
    setAllValid = increment = 0;
  }
};
// 0-based indexing
class segtree
{
public:
  int range;
  vector<node> tree;
  void build(vector<int> &v)
  {
    range = v.size();
    tree.assign(4 * range, node());
    build_recur(v, 0, range - 1, 0);
  }
  void build_recur(vector<int> &v, int l, int r,
      int node_no)
  {
    if (l == r)
    {
      if (l < v.size())
        tree[node_no].sum = v[l];
      else
        tree[node_no].sum = 0;
      return;
    }
    int mid = (l + r) / 2;
    build_recur(v, l, mid, 2 * node_no + 1);
    build_recur(v, mid + 1, r, 2 * node_no + 2);
    tree[node_no].sum = tree[2 * node_no + 1].sum
        + tree[2 * node_no + 2].sum;
  }
  ll range_query(int L, int R)
  {
    return range_query_recur(0, 0, range - 1, L,
        R);
  }
  void incUpdate_recur(int node, int l, int r,
      int &L, int &R, int &X)
  {
    if (r < L || R < l || l >= range)
      return;
    if (L <= l && R >= r)
    {
      tree[node].increment += X;
      return;
    }
    applyAggr(node, l, r);
    int mid = (l + r) / 2;
    incUpdate_recur(2 * node + 1, l, mid, L, R,
        X);
    incUpdate_recur(2 * node + 2, mid + 1, r, L,
        R, X);
    applyAggr(2 * node + 1, l, mid);
    applyAggr(2 * node + 2, mid + 1, r);
    tree[node].sum = tree[2 * node + 1].sum +
        tree[2 * node + 2].sum;
  }
  void incUpdate(int L, int R, int X)
  {
    incUpdate_recur(0, 0, range - 1, L, R, X);
  }
  void setUpdate_recur(int node, int l, int r,
      int &L, int &R, int &X)
  {
    if (r < L || R < l || l >= range)
      return;
    if (L <= l && R >= r)
    {
      tree[node].setAllValid = 1;
      tree[node].setAll = X;
      tree[node].increment = 0;
      return;
```

```
  }
  applyAggr(node, l, r);
  int mid = (l + r) / 2;
  setUpdate_recur(2 * node + 1, l, mid, L, R,
      X);
  setUpdate_recur(2 * node + 2, mid + 1, r, L,
      R, X);
  applyAggr(2 * node + 1, l, mid);
  applyAggr(2 * node + 2, mid + 1, r);
  tree[node].sum = tree[2 * node + 1].sum +
      tree[2 * node + 2].sum;
}
void setUpdate(int L, int R, int X)
{
  setUpdate_recur(0, 0, range - 1, L, R, X);
}
void compose(int par, int child)
{
  if (tree[par].setAllValid)
  {
    tree[child].setAllValid = 1;
    tree[child].setAll = tree[par].setAll;
    tree[child].increment = tree[par].increment;
  }
  else
    tree[child].increment += tree[par].increment;
}
void applyAggr(int node, int l, int r)
{
  if (tree[node].setAllValid)
    tree[node].sum = (r - l + 1) *
        tree[node].setAll;
  tree[node].sum += (r - l + 1) *
      tree[node].increment;
  if (l != r)
  {
    compose(node, 2 * node + 1);
    compose(node, 2 * node + 2);
  }
  tree[node].Reset();
}
ll range_query_recur(int node, int l, int r,
    int &L, int &R)
{
```

```
  if (r < L || R < l || l >= range)
    return 0;
  applyAggr(node, l, r);
  if (L <= l && R >= r)
    return tree[node].sum;
  int mid = (l + r) / 2;
  return range_query_recur(2 * node + 1, l,
      mid, L, R) + range_query_recur(2 * node +
      2, mid + 1, r, L, R);
 }
};
```

5.3 Segment Tree

```
class SegmentTree
{
public:
  vector<vector<int>> tree; // 0-mn, 1-mx, 2-sum
  SegmentTree(int n)
  {
    tree.assign(3, vector<int>(4 * n + 1, 0));
  }
  void build(vector<int> &a, int l, int r, int
      node)
  {
    if (l > r)
      return;
    if (l == r)
    {
      tree[0][node] = a[l];
      tree[1][node] = a[l];
      tree[2][node] = a[l];
      return;
    }
    int mid = (l + r) / 2;
    build(a, l, mid, 2 * node);
    build(a, mid + 1, r, 2 * node + 1);

    tree[0][node] = min(tree[0][2 * node],
        tree[0][2 * node + 1]);
    tree[1][node] = max(tree[1][2 * node],
        tree[1][2 * node + 1]);
    tree[2][node] = tree[2][2 * node] +
        tree[2][2 * node + 1];
  }

  void update(int node, int l, int r, int idx,
      int value)
  {
    if (l > r)
      return;
    if (l == r)
    {
      tree[0][node] = value;
      tree[1][node] = value;
      tree[2][node] = value;
      return;
    }
    int mid = (l + r) / 2;
    if (idx >= l and idx <= mid)
    {
      update(2 * node, l, mid, idx, value);
    }
    else
    {
      update(2 * node + 1, mid + 1, r, idx,
          value);
    }
    tree[0][node] = min(tree[0][2 * node],
        tree[0][2 * node + 1]);
    tree[1][node] = max(tree[1][2 * node],
        tree[1][2 * node + 1]);
    tree[2][node] = tree[2][2 * node] +
        tree[2][2 * node + 1];
  }

  // b=begin, e=end
  int query_mn(int node, int l, int r, int b,
      int e)
  {
    if (l >= b and r <= e)
      return tree[0][node];
    if (l > e or r < b)
      return 1e18; // return invalid value;
    int mid = (l + r) / 2;
```

```
        int left = query_mn(2 * node, l, mid, b,
            e);
        int right = query_mn(2 * node + 1, mid +
            1, r, b, e);
        return min(left, right);
    }
    int query_mx(int node, int l, int r, int b,
        int e)
    {
        if (l >= b and r <= e)
            return tree[1][node];
        if (l > e or r < b)
            return -1e18; // return invalid value;
        int mid = (l + r) / 2;
        int left = query_mx(2 * node, l, mid, b,
            e);
        int right = query_mx(2 * node + 1, mid +
            1, r, b, e);
        return max(left, right);
    }
    int query_sum(int node, int l, int r, int b,
        int e)
    {
        if (l >= b and r <= e)
            return tree[2][node];
        if (l > e or r < b)
            return 0; // return invalid value;
        int mid = (l + r) / 2;
        int left = query_sum(2 * node, l, mid, b,
            e);
        int right = query_sum(2 * node + 1, mid +
            1, r, b, e);
        return left + right;
    }
};
```

5.4 ordered set

```
// Ordered Set
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

template <typename T> using o_set = tree<T,
    null_type, less<T>,rb_tree_tag,
    tree_order_statistics_node_update>;
int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    o_set<int> se;
    cout << se.order_of_key(5) << '\n'; // number
        of elements < 5
    se.erase(3); // erase by value
    cout << (*se.find_by_order(1)) << '\n'; // if
        you imagine this as a 0-indexed vector,
        what is se[1]?
    return 0;
}
```

6 E.DP

6.1 Digit DP

```
int len,id,inp[10];
LL dp[10][10][2][2];
int vis[10][10][2][2];
LL call(int pos,bool isSmall,bool isStart,int
    total) {
    if(pos==len) return total;
    if(vis[pos][total][isSmall][isStart]==id)
        return dp[pos][total][isSmall][isStart];
    vis[pos][total][isSmall][isStart]=id;
    int last=9;
    if(!isSmall) last=inp[pos];
    LL ret=0;
    if(isStart) {
        nfor(int i=0; i<=last; i++) {
            ret+=call(pos+1,isSmall |
                i<inp[pos],1,(i==0)+total);
        }
    }
    else {
        for(int i=1; i<=last; i++) {
            ret+=call(pos+1,isSmall |
                i<inp[pos],1,(i==0)+total);
        }
        ret+=call(pos+1,1,0,0);
    }
    return dp[pos][total][isSmall][isStart]=ret;
}
LL solve(LL x) {
    if(x<0) return 0;
    len=0;
    while(x) {
        inp[len++]=x%10;
        x/=10;
    }
    reverse(inp,inp+len);
    id++;
    return call(0,0,0,0)+1;
}
int main() {
    int t;
    scanf("%d",&t);
    FOR(tc,1,t) {
        LL lo,hi;
        scanf("%lld %lld",&lo,&hi);
        printf("Case %d:
            %lld\n",tc,solve(hi)-solve(lo-1));
    }
```

6.2 Knapsack 1

```
int n, w; cin>>n>>w;
vector<int> dp(w+1, 0);
for(int i=0; i<n; i++){
    int weight, val; cin>>weight>> val;
    for(int prev_weight=w-weight; prev_weight>=0;
        prev_weight--){
        dp[prev_weight+weight] =
            max(dp[prev_weight+weight],
            dp[prev_weight]+val);
    }
```

```
}
cout<<dp[w]<<nl;
```

6.3 Longest Incresing Subsequence

```
int longest(int n) {
    vector<int>v;
    for(int i=1; i<=n; i++) {
        int pos = lower_bound(v.begin(), v.end(),
            a[i]) - v.begin();
        if(pos == v.size()) v.push_back(a[i]);
        else v[pos] = a[i];
    }
    return v.size();
}

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

const int N = 3030;
string a, b;
int dp[N][N];
int lcs(int i, int j) {
  if (i >= a.size() or j >= b.size()) return 0;
  if (dp[i][j] != -1) return dp[i][j];
  int ans = lcs(i + 1, j);
  ans = max(ans, lcs(i, j + 1));
  if (a[i] == b[j]) {
    ans = max(ans, lcs(i + 1, j + 1) + 1);
  }
  return dp[i][j] = ans;
}
void print(int i, int j) {
  if (i >= a.size() or j >= b.size()) return;
  if (a[i] == b[j]) {
    cout << a[i];
    print(i + 1, j + 1);
    return;
  }
  int x = lcs(i + 1, j);
  int y = lcs(i, j + 1);
  if (x >= y) {
    print(i + 1, j);
  }
  else {
    print(i, j + 1);
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cin >> a >> b;
  memset(dp, -1, sizeof dp);
  // cout << lcs(0, 0) << '\n';
  print(0, 0);
  return 0;
}
```

7 F.Graph

7.1 Bellman Ford Negetive Cycle Detection

```
struct node {
    int u;
    int v;
    int wt;
    node(int first, int second, int weight)
    {
        u = first;
        v = second;
        wt = weight;
    }
};
const int inf = 10000000;
const int SZ = 1e6+5;
int dist[SZ];
vector<node> edges;
void bellmenFord(int n) {
    for(int i = 0; i<=n-1; i++) {
        for(auto it: edges) {
            if(dist[it.u] + it.wt < dist[it.v]) {
                dist[it.v] = dist[it.u] + it.wt;
            }
        }
    }
}
int isNegCycle() {
    for(auto it: edges) {
        if(dist[it.u] + it.wt < dist[it.v]) {
            return 1;
        }
    }
    return 0;
}
```

7.2 DFS Articulation Point

```
const ll SZ = 1e6 + 5;
ll tin[SZ], low[SZ], vis[SZ], isArticulation[SZ];
vector<ll> adj[SZ];
ll timer;
void dfs(ll node, ll parent, ll timer) {
    vis[node] = 1;
    tin[node] = low[node] = timer++;
    ll childCnt = 0;
    for (auto child : adj[node]) {
        if (child == parent) {
            continue;
        }
        if (vis[child] == 0) {
            dfs(child, node, timer);
            low[node] = min(low[node], low[child]);
            if (low[child] >= tin[node] && parent
                != -1) {
                isArticulation[node] = 1;
            }
        }
        else {
            low[node] = min(low[node], tin[child]);
        }
    }
    if (parent == -1 && childCnt > 1) {
        isArticulation[node] = 1;
```

```
    }
}
void init(ll n) {
    mem(vis, 0);
    mem(isArticulation, 0);
    mem(tin, -1);
    mem(low, -1);
    for (int i = 1; i <= n; i++) {
        adj[i].clear();
    }
}
//dfs call -> dfs(i, -1, timer);
```

7.3 DFS Bridge Graph

```
const ll SZ = 1e6 + 5;
ll tin[SZ], low[SZ], vis[SZ];
vector<ll> adj[SZ];
ll timer;
void dfs(ll node, ll parent, ll timer) {
    vis[node] = 1;
    tin[node] = low[node] = timer++;
    for (auto child : adj[node]) {
        if (child == parent) continue;
        if (vis[child] == 0) {
            dfs(child, node, timer);
            low[node] = min(low[node], low[child]);
            if (low[child] > tin[node])
                vp.push_back({min(child, node),
                max(node, child)});
        }
        else low[node] = min(low[node],
            tin[child]);
    }
}

void init()
{
    memset(vis, 0, sizeof(vis));
    memset(tin, -1, sizeof(tin));
    memset(low, -1, sizeof(low));
    for (int i = 0; i < n; i++) adj[i].clear();
}
// dfs call -> dfs(i, -1, timer);
```

7.4 DFS LCA

```
const int N = 3e5 + 9, LG = 18;
vector<int> g[N];
int par[N][LG + 1], dep[N], sz[N];
// Call at first
void dfs(int u, int p = 0) {
  par[u][0] = p;
  dep[u] = dep[p] + 1;
  sz[u] = 1;
  for (int i = 1; i <= LG; i++) par[u][i] =
      par[par[u][i - 1]][i - 1];
  for (auto v: g[u]) if (v != p) {
    dfs(v, u);
    sz[u] += sz[v];
  }
}
// Lowest Common Ancestor
int lca(int u, int v) {
  if (dep[u] < dep[v]) swap(u, v);
  for (int k = LG; k >= 0; k--) if
      (dep[par[u][k]] >= dep[v]) u = par[u][k];
  if (u == v) return u;
  for (int k = LG; k >= 0; k--) if (par[u][k] !=
      par[v][k]) u = par[u][k], v = par[v][k];
  return par[u][0];
}
// K-th root from u
int kth(int u, int k) {
  assert(k >= 0);
  for (int i = 0; i <= LG; i++) if (k & (1 << i))
      u = par[u][i];
  return u;
}
// Calculate distance between u and v
int dist(int u, int v) {
  int l = lca(u, v);
  return dep[u] + dep[v] - (dep[l] << 1);
}
//kth node from u to v, 0th node is u
int go(int u, int v, int k) {
  int l = lca(u, v);
  int d = dep[u] + dep[v] - (dep[l] << 1);
  assert(k <= d);
  if (dep[l] + k <= dep[u]) return kth(u, k);
  k -= dep[u] - dep[l];
  return kth(v, dep[v] - dep[l] - k);
}
int32_t main() {
  int n; cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs(1);
  int q; cin >> q;
  while (q--) {
    int u, v; cin >> u >> v;
    cout << dist(u, v) << '\n';
  }
  return 0;
}
```

7.5 DSU

```
// 1-based indexing
class DSU
{
public:
    vector<int> parent, rating;
    int cc_count;
    DSU(int n)
    {
        cc_count = n;
        parent.assign(n + 10, 0);
        rating.assign(n + 1, 1);
        for (int i = 1; i <= n; i++)
        {
            parent[i] = i;
            rating[i] = 1;
```

```
        }
    }
    int find_parent(int node)
    {
        return parent[node] = (parent[node] ==
            node ? node :
            find_parent(parent[node]));
    }
    bool is_in_same(int u, int v)
    {
        int par1 = find_parent(u);
        int par2 = find_parent(v);
        if (par1 == par2)
        {
            return true;
        }
        return false;
    }
    // Returns size of component
    int get_size(int u) { return
        rating[find_parent(u)]; }
    // False means already connected
    bool merge(int u, int v)
    {
        if (is_in_same(u, v))
            return false;
        int par1 = find_parent(u);
        int par2 = find_parent(v);
        if (rating[par1] >= rating[par2])
        {
            parent[par2] = par1;
            rating[par1] += rating[par2];
        }
        else
        {
            parent[par1] = par2;
            rating[par2] += rating[par1];
        }
        --cc_count;
        return true;
    }
}
```

7.6 Dijkstra by GM

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

const int N = 3e5 + 9, mod = 998244353;

int n, m;
vector<pair<int, int>> g[N], r[N];
vector<long long> dijkstra(int s, int t,
    vector<int> &cnt) {
    const long long inf = 1e18;
    priority_queue<pair<long long, int>,
        vector<pair<long long, int>>,
        greater<pair<long long, int>>> q;
    vector<long long> d(n + 1, inf);
    vector<bool> vis(n + 1, 0);
    q.push({0, s});
    d[s] = 0;
    cnt.resize(n + 1, 0); // number of shortest
        paths
    cnt[s] = 1;
    while(!q.empty()) {
        auto x = q.top();
        q.pop();
        int u = x.second;
        if(vis[u]) continue;
        vis[u] = 1;
        for(auto y: g[u]) {
        int v = y.first;
        long long w = y.second;
        if(d[u] + w < d[v]) {
            d[v] = d[u] + w;
            q.push({d[v], v});
            cnt[v] = cnt[u];
        } else if(d[u] + w == d[v]) cnt[v] =
            (cnt[v] + cnt[u]) % mod;
        }
    }
    return d;
}

int u[N], v[N], w[N];

int32_t main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    int s, t;
    cin >> n >> m >> s >> t;
    for(int i = 1; i <= m; i++) {
        cin >> u[i] >> v[i] >> w[i];
        g[u[i]].push_back({v[i], w[i]});
        r[v[i]].push_back({u[i], w[i]});
    }
    vector<int> cnt1, cnt2;
    auto d1 = dijkstra(s, t, cnt1);
    auto d2 = dijkstra(t, s, cnt2);

    long long ans = d1[t];
    for(int i = 1; i <= m; i++) {
        int x = u[i], y = v[i];
        long long nw = d1[x] + w[i] + d2[y];
        if(nw == ans && 1LL * cnt1[x] * cnt2[y] %
            mod == cnt1[t]) cout << "YES\n";
        else if(nw - ans + 1 < w[i]) cout << "CAN
            " << nw - ans + 1 << '\n';
        else cout << "NO\n";
    }
    return 0;
}
```

7.7 Eular Path on Undirected Graph

```
/*
all the edges should be in the same connected
    component
#undirected graph: euler path: all degrees are
    even or exactly two of them are odd.
#undirected graph: euler circuit: all degrees are
    even
*
find_euler_tour(u):
    for each node v in adj[u]
        remove edge u, v
        find_euler_tour(b)
```

```
    push(u) // Stored in reverse order
*/
//euler path in an undirected graph
//it also finds circuit if it exists
vector<pair<int, int>> g[N];
vector<int> ans;
int done[N];
int vis[N * N]; //number of edges
void dfs(int u) {
  while (done[u] < g[u].size()) {
    auto e = g[u][done[u]++];
    if (vis[e.second]) continue;
    vis[e.second] = 1;
    dfs(e.first);
  }
  ans.push_back(u);
}
int solve(int n) {
  int edges = 0;
  ans.clear();
  memset(done, 0, sizeof done);
  memset(vis, 0, sizeof vis);
  vector<int> deg(n + 1, 0);
  for (int u = 1; u <= n; u++) {
    for (auto e : g[u]) {
      deg[e.first]++, deg[u]++, edges++;
    }
  }
  int odd = 0, root = 0;
  for (int i = 1; i <= n; i++) {
    if (deg[i] & 1) odd++, root = i;
  }
  if (odd > 2) return 0;
  if (root == 0) {
    for (int i = 1; i <= n; i++) if (deg[i]) root
        = i;
  }
  if (root == 0) return 1; //empty graph
  dfs(root);
  if (ans.size() != edges / 2 + 1) return 0;
  reverse(ans.begin(), ans.end());
  return 1;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int t;
  cin >> t;
  while (t--) {
    int n, m;
    cin >> n >> m;
    vector<int> deg(n + 1, 0);
    for (int i = 1; i <= m; i++) {
      int u, v;
      cin >> u >> v;
      g[u].push_back({v, i});
      g[v].push_back({u, i});
      deg[u]++, deg[v]++;
    }
    int sz = m;
    for (int i = 1; i <= n; i++) {
      if (deg[i] & 1) {
        ++sz;
        g[n + 1].push_back({i, sz});
        g[i].push_back({n + 1, sz});
      }
    }
    int ok = solve(n + 1);
    assert(ok);
    vector<int> in(n + 2, 0), out(n + 2, 0);
    for (int i = 0; i + 1 < ans.size(); i++) {
      if (ans[i] != n + 1 && ans[i + 1] != n + 1) {
        in[ans[i + 1]]++;
        out[ans[i]]++;
      }
    }
    int res = 0;
    for (int i = 1; i <= n; i++) res += in[i] ==
        out[i];
    cout << res << '\n';
    for (int i = 0; i + 1 < ans.size(); i++) if
        (ans[i] != n + 1 && ans[i + 1] != n + 1)
        cout << ans[i] << ' ' << ans[i + 1] <<
        '\n';
    for (int i = 0; i <= n + 1; i++) g[i].clear();
  }
  return 0;
}
```

7.8 Minimum Spanning Tree from Each Egde

```
const double inf = 0.0000;
const int lx = 2e5 + 5;
const int mod = 998244353;
const int hs = 3797;
struct graph {
    int u;
    int v;
    int w;
    int idx;

    bool operator < (const graph ob) const {
        return w < ob.w;
    }
};

int cost[lx];
int bap[lx], n, m;
graph temp;
vector<graph> adj;
set<int> e[lx];

int find(int x)
{
    if(x == bap[x]) return x;
    return bap[x] = find(bap[x]);
}
void answer(int u, int v, int w)
{
    u = find(u);
    v = find(v);

    if(e[u].size() < e[v].size())
        swap(u, v);
    bap[v] = u;

    while(e[v].begin() != e[v].end()) {
```

```
        if(e[u].find(*e[v].begin()) != e[u].end())
            {
            cost[*e[v].begin()] -= w;
        }
        else e[u].insert(*e[v].begin());
        e[v].erase(e[v].begin());
    }
}
void solve()
{
    cin >> n >> m;
    for(int i = 1; i <= n; i++) bap[i] = i;
    for(int i = 1; i <= m; i++) {
        cin >> temp.u >> temp.v >> temp.w;
        temp.idx = i;
        cost[i] = temp.w;
        adj.push_back(temp);

        e[temp.u].insert(i);
        e[temp.v].insert(i);
    }

    sort(adj.begin(), adj.end());
    ll ans = 0;
    for(int i = 0; i < m; i++) {
        int u = adj[i].u;
        int v = adj[i].v;
        int w = adj[i].w;
        int idx = adj[i].idx;

        if(find(u) == find(v)) continue;

        answer(u, v, w);
        ans += w;
    }
    for(int i = 1; i <= m; i++) cout << ans + 1LL
        * cost[i] << ' ';
    cout << endl;
}
```

7.9 Notes

1. To make a tree cyclic, add $\frac{\text{number of leaves} + 1}{2}$ number of edges.
2. An Euler path exists in an undirected graph if and only if the degree of each node is even. In a directed graph, an Euler path exists if and only if the indegree and outdegree of each node are equal.
3. A graph is a biconnected component (block) if it has no articulation points, but may have bridges (like two nodes). A graph is a bridge-tree if it has no bridges, but may have articulation points.
4. For an Euler path to exist in an undirected graph, the graph must have exactly two nodes with odd degree. The path will start at one node and end at the other. Additionally, exactly one node should have an indegree equal to its outdegree plus one, and exactly one node should have an outdegree equal to its indegree plus one.
5. An Euler path or cycle exists in a graph if and only if the graph is connected.

7.10 Topological Sort

```
vector<vector<int>> adjList;
vector<int> topoLogicalSort(vector<pair<int,
    int>> edges, int nodeCount) {
    vector<int> inDegree(nodeCount + 1, 0);
    vector<vector<int>> adjList(nodeCount + 1);
    for(auto edge : edges) {
        inDegree[edge.second] += 1;
        adjList[edge.first].push_back(edge.second);
    }
    queue<int> nodesWithIndegreeZero;
    for(int i = 0; i <nodeCount; i++) {
        if(inDegree[i] == 0) {
            nodesWithIndegreeZero.push(i);
        }
    }
    vector<int> topoLogicallySortedNodes;
    while (!nodesWithIndegreeZero.empty()) {
        int node = nodesWithIndegreeZero.front();
        topoLogicallySortedNodes.push_back(node);
        nodesWithIndegreeZero.pop();
        for (auto x: adjList[node]) {
            inDegree[x]--;
            if (inDegree[x] == 0) {
                nodesWithIndegreeZero.push(x);
            }
        }
    }
    return topoLogicallySortedNodes;
}
```

7.11 dijkstra simple

```
// Dijkstra for weighted graph with non-negative
    weights
void dijkstra(int src, vector<vector<pair<int,
    int>>> &adj, vector<int> &dist)
{
  int n = adj.size(); // Nodes are 0-based
  dist.assign(n, INT_MAX);
  priority_queue<pair<int, int>, vector<pair<int,
      int>>, greater<>> pq;
  dist[src] = 0;
  pq.push({0, src});
  while (!pq.empty())
  {
    auto [d, u] = pq.top();
    pq.pop();
    if (d > dist[u])
      continue; // Outdated entry
    for (auto [v, w] : adj[u])
    {
      if (dist[u] + w < dist[v])
      {
        dist[v] = dist[u] + w;
        pq.push({dist[v], v});
```

```
      }
    }
  }
}
// Assumes adj[u] = vector of {neighbor, weight}
// Make sure graph has no negative weights
```

8 G.Tree

8.1 2D Fenwick Tree

```
const ll sz = 1040;
ll fre[sz][sz], ar[sz][sz];
ll n;
void update(ll x, ll y, ll val) {
    ++x; ++y;
    for(ll i=x; i<=n; i += (i&(-i))) {
        for(ll j=y; j<=n ; j += (j&(-j))) {
            fre[i][j] += val;
        }
    }
}
ll query(ll x, ll y) {
    ++x; ++y;
    ll sum=0;
    for(ll i=x; i>0; i -= (i&(-i))) {
        for(ll j=y; j>0 ; j -= (j&(-j))) {
            sum += fre[i][j];
        }
    }
    return sum;
}
ll areaSum(ll x1, ll y1, ll x2, ll y2) {
    ll ans = query(x2, y2)
             - query(x2, y1-1) - query(x1-1, y2)
             + query(x1-1, y1-1);
    return ans;
}
```

8.2 MST Kruskal's

```
// DSU is required
// sort the edge according to your problem(asc,
    dsc)
// Asc -> Minimum spanning tree
// Dsc -> Maximum spanning tree
    int n, m; cin >> n >> m;
    vector<array<int, 3>> ed;
    for(int i = 1; i <= m; i++){
        int u, v, w; cin >> u >> v >> w;
        ed.push_back({w, u , v});
    }
    sort(ed.begin(), ed.end());
    long long ans = 0;
    dsu d(n);
    for (auto e: ed){
        int u = e[1], v = e[2], w = e[0];
        if (d.same(u, v)) continue;
        ans += w;
        d.merge(u, v);
    }
    cout << ans << '\n';
```

8.3 MST To find Kth Sortest Number in Range

```
#define ULL          unsigned long long
#define ff           first
#define ss           second
#define pb           push_back
#define pii          pair<int,int>
#define all(a)       a.begin(),a.end()
#define MEM(a,x)     memset(a,x,sizeof(a))
#define FOR(i,a,b)   for(int i=a;i<=b;i++)
#define ROF(i,a,b)   for(int i=a;i>=b;i--)
#define REP(i,b)     for(int i=0;i<b;i++)
const int N=1e6+5;
pii arr[N];
int flag[N];
vector<int>tree[4*N];
void build(int node,int L,int R)
{
    if(L==R){
        tree[node].pb(arr[L].ss);
        return;
    }
    int mid=(L+R)/2;
    build(node*2,L,mid);
    build(2*node+1,mid+1,R);
    merge(all(tree[2*node]),all(tree[2*node+1]),back_inse
}
int query(int node,int L,int R,int l,int r,int
    val)
{
    if(L==R) return tree[node][0];
    int
        cnt=upper_bound(all(tree[2*node]),r)-lower_bound(
    int mid=(L+R)/2;
    if(val>cnt) return
        query(node*2+1,mid+1,R,l,r,val-cnt);
    else return query(node*2,L,mid,l,r,val);
}
int main()
{
    int n,q;
    while(scanf("%d %d",&n,&q)!=EOF){
        FOR(i,1,4*n) tree[i].clear();
        FOR(i,1,n) scanf("%d",&arr[i].ff);
        FOR(i,1,n) arr[i].ss=i,flag[i]=arr[i].ff;
        sort(arr+1,arr+n+1);
        build(1,1,n);
        while(q--){
            int l,r,k;
            scanf("%d %d %d",&l,&r,&k);
            printf("%d\n",flag[query(1,1,n,l,r,k)]);
        }
    }
}
```

8.4 Merger Sort Tree Using Segment Tree

```
const int N=1e6+5;
vector<int>V[N],tree[4*N];
void build(int node,int L,int R) {
    if(L==R) {
        sort(all(V[L]));
        tree[node]=V[L];
        return;
    }
    int mid=(L+R)/2;
    build(node*2,L,mid);
    build(2*node+1,mid+1,R);
    merge(all(tree[2*node]),all(tree[2*node+1]),
    back_inserter(tree[node]));
}
int query(int node,int L,int R,int l,int r,int
     val) {
    if(r<L or R<l or tree[node].empty()) return 0;
    if(l<=L and R<=r) {
        int cnt=upper_bound(all(tree[node]),val)
        -tree[node].begin();
        return cnt;
    }
    int mid=(L+R)/2;
    int x=query(node*2,L,mid,l,r,val);
    int y=query(node*2+1,mid+1,R,l,r,val);
    return x+y;
}
```

8.5 Minimum Spanning Tree – MST using Prim's Algo

```
int main() {
    int N=5,m=6;
    vector<pair<int,int> > adj[N];
    int parent[N];
    int key[N];
    bool mstSet[N];
    for (int i = 0; i < N; i++)
        key[i] = INT_MAX, mstSet[i] = false;
    priority_queue< pair<int,int>, vector
        <pair<int,int>>, greater<pair<int,int>> >
         pq;
    key[0] = 0;
    parent[0] = -1;
    pq.push({0, 0});
    while(!pq.empty()) {
        int u = pq.top().second;
        pq.pop();
        mstSet[u] = true;
        for (auto it : adj[u]) {
            int v = it.first;
            int weight = it.second;
            if (mstSet[v] == false && weight <
                 key[v]) {
                parent[v] = u;
                key[v] = weight;
                pq.push({key[v], v});
            }
        }
    }
    for (int i = 1; i < N; i++)
        cout << parent[i] << " - " << i <<" \n";
    return 0;

}
```

9 H.Strings

9.1 KMP Algorithm

```
// finding frequency of P in S, as a substring.
//  call build_failure_function() first
#define MAX 1000005
int failure[MAX];
// longest prefix that also mactchs current suffix
void build_failure_function(string pattern, int m)
{
    failure[0] = 0;
    failure[1] = 0; // base case

    for (int i = 2; i <= pattern.size(); i++)
    {
        int j = failure[i - 1];
        while (true)
        {
            if (pattern[j] == pattern[i - 1])
            {
                failure[i] = j + 1;
                break;
            }
            if (j == 0)
            {
                failure[i] = 0;
                break;
            }
            j = failure[j];
        }
    }
}

int frequency_of_P_in_S(string &s, stirng &p)
{
    build_failure_function(p, p.size());
    int total_mtc = 0;
    int ans = 0;
    for (int i = 0; i < s.size(); i++)
    {
        if (s[i] == p[total_mtc])
            total_mtc++;
        else
        {
            while (total_mtc > 0)
            {
                total_mtc = failure[total_mtc];
                if (s[i] == p[total_mtc])
                {
                    total_mtc++;
                    break;
                }
            }
        }
        if (p.size() == total_mtc)
        {
            ans++;
            total_mtc = failure[total_mtc];
        }
```

```
    }
    return ans;
}
```

9.2 LCS

```
int L[m + 1][n + 1];
for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0) L[i][j] = 0;
        else if (X[i - 1] == Y[j - 1]) L[i][j] =
            L[i - 1][j - 1] + 1;
        else L[i][j] = max(L[i - 1][j], L[i][j -
            1]);
    }
}
```

9.3 Lexiographically smallest string by cycle shift

```
int minlex(char[] s) {
    int len = s.size();
    int n = 2*len, i=0, j=1, k=0, a, b;
    while(i+k<n && j+k<n) {
        a=(i+k>=len)?s[i+k-len]:s[i+k];
        b=(j+k>=len)?s[j+k-len]:s[j+k];
        if(a==b) k++;
        else if(a>b) {
            i=i+k+1;
            if(i<=j) i=j+1;
            k=0;
        }
        else {
            j=j+k+1;
            if(j<=i) j=i+1;
            k=0;
        }
    }
    return min(i, j);
}
```

9.4 Longest Palindromic Subsequence

```
for (int i = 1; i <= S.length(); i++) {
    for (int j = 1; j <= R.length(); j++) {
        if (S[i - 1] == R[j - 1]) dp[i][j] = 1 +
            dp[i - 1][j - 1];
        else dp[i][j] = max(dp[i][j - 1], dp[i -
            1][j]);
    }
}
```

9.5 Manacher's Algorithm

```
int p[lx];
void manachers_algorithm(string s)
{
    string t = "#";
    for(char c : s) {
        t += c;
        t += '#';
    }
    t = "-" + t + "?";
    int l = 1, r = 1, n = s.size();
    for(int i = 1; i < n; i++) {
        p[i] = max(0, min(r - i, p[l + r - i]));

        while(s[i - p[i]] == s[i + p[i]]) {
            ++p[i];
        }
        if(i + p[i] > r) {
            l = i - p[i];
            r = i + p[i];
        }
    }
}
```

9.6 Trie

```
struct Node {
    Node *next[26];
    int frequency;
    Node() {
        frequency = 0;
        for (int i = 0; i < 26; i++) next[i] =
            nullptr;
    }
};
void addString(Node *root, string s) {
    Node *cur = root;
    for (char c : s) {
        if (cur->next[c-'a'] == nullptr)
            cur->next[c-'a'] = new Node();
        cur = cur->next[c-'a'];
        cur->frequency++;
    }
}
int queryString(Node *root, string s) {
    Node *cur = root;
    for (char c : s) {
        if (cur->next[c-'a'] == nullptr) return 0;
        cur = cur->next[c-'a'];
    }
    return cur->frequency;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n, q;
    cin >> n >> q;
    Node *root = new Node();
    for (int i = 0; i < n; i++) {
        string s;
        cin >> s;
        addString(root, s);
    }
    for (int i = 0; i < q; i++) {
        string s;
        cin >> s;
```

```
        cout << queryString(root, s) << "\n";
    }
    return 0;
}
```

9.7 Z function

```
int z[lx];
void Z_Algorithm(string s)
{
    z[0] = 0;
    int l = 0, r = 0, n = s.size();
    for(int i = 1; i < n; i++) {
        if(i <= r) {
            z[i] = min(z[i - l], r - i + 1); //
                Mirror Index of i will be i - l
        }
        while(i + z[i] < n and s[z[i]] == s[i +
            z[i]]) {
            ++z[i];
        }
        if(i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
}
```

9.8 string hashing

```
// Don't Use int=long long
// call prec() funtion from the main function
// create an object of Hashing with a
    (string)parameter first
const int N = 1e6 + 9;
int power(long long n, long long k, const int
    mod) {
    int ans = 1 % mod;
    n %= mod;
    if (n < 0) n += mod;
    while (k) {
        if (k & 1) ans = (long long) ans * n % mod;
        n = (long long) n * n % mod;
        k >>= 1;
    }
    return ans;
}
// extra mod 999999989, 1e9+9, 1e9+7
const int MOD1 = 127657753, MOD2 = 987654319;
const int p1 = 137, p2 = 277;
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
    pw[0] = {1, 1};
    for (int i = 1; i < N; i++) {
        pw[i].first = 1LL * pw[i - 1].first * p1 %
            MOD1;
        pw[i].second = 1LL * pw[i - 1].second * p2
            % MOD2;
    }
    ip1 = power(p1, MOD1 - 2, MOD1);
    ip2 = power(p2, MOD2 - 2, MOD2);
    ipw[0] = {1, 1};
    for (int i = 1; i < N; i++) {
        ipw[i].first = 1LL * ipw[i - 1].first *
            ip1 % MOD1;
        ipw[i].second = 1LL * ipw[i - 1].second *
            ip2 % MOD2;
    }

}
struct Hashing {
    int n;
    string s; // 0 - indexed
    vector<pair<int, int>> hs; // 1 - indexed
    Hashing() {}
    Hashing(string _s) {
        n = _s.size();
        s = _s;
        hs.emplace_back(0, 0);
        for (int i = 0; i < n; i++) {
        pair<int, int> p;
        p.first = (hs[i].first + 1LL * pw[i].first
            * s[i] % MOD1) % MOD1;
        p.second = (hs[i].second + 1LL *
            pw[i].second * s[i] % MOD2) % MOD2;
        hs.push_back(p);
        }
    }
    pair<int, int> get_hash(int l, int r) { // 1
        - indexed
        assert(1 <= l && l <= r && r <= n);
        pair<int, int> ans;
        ans.first = (hs[r].first - hs[l - 1].first
            + MOD1) * 1LL * ipw[l - 1].first %
            MOD1;
        ans.second = (hs[r].second - hs[l -
            1].second + MOD2) * 1LL * ipw[l -
            1].second % MOD2;
        return ans;
    }
    pair<int, int> get_hash() {
        return get_hash(1, n);
    }
};
```

10 I.Bit

10.1 Equation

* a+b = a$\oplus b + 2(a\&b)$
* a+b = a—b+a&b
* a$\oplus b = a|b - a\&b$
* k$_{th}$ bit is set in x iff $x \bmod 2^{k-1} > 2^k$
*

k$_{th}$ bit is set in x iff $x \bmod 2^{k-1} - x \bmod 2^k \neq 0 (= 2^k)$

* n mod$2^i = n\&(2^i - 1)$
* $0.1 \oplus 2 \oplus 3 \oplus \cdots \oplus (4k-1) = 0$ for any $k \geq 0$

10.2 Maximum And Pair

```
Int checkbit(int p, int n) {
    cnt = 0;
    for(i=0; i<n; i++) {
```

```
        if(p&(ai==p)cnt++;
    }
return cnt;
}
for(b=32; b>=0; b--) {
    cnt = checkbit(res |<<b, n);
    if(cnt >=2) res|=1<<b;
}
```

10.3 Maximum XOR of all subsequence

```
#define INT_BITS 32
int maxSubarrayXOR(int set[], int n) {
    for (int i = INT_BITS-1; i >= 0; i--) {
        int maxInd = index;
        int maxEle = INT_MIN;
        for (int j = index; j < n; j++) {
            if ( (set[j] & (1 << i)) != 0 &&
                 set[j] > maxEle )
                maxEle = set[j], maxInd = j;
        }
        if (maxEle == INT_MIN) continue;
        swap(set[index], set[maxInd]);
        maxInd = index;
        for (int j=0; j<n; j++) {
            if (j != maxInd && (set[j] & (1 << i))
                 != 0)
                set[j] = set[j] ^ set[maxInd];
        }
        index++;
    }
    int res = 0;
    for (int i = 0; i < n; i++)
        res ^= set[i];
    return res;
}
```

10.4 Minimum XOR Operation

Minimum xor pair - Trie

1. Sort the array.
2. Find the minimum of $ar[i] \oplus ar[i+1]$, where $\oplus$ denotes the XOR operation.

Minimum XOR subarray - Trie

Minimum XOR of OR and AND in an array

1. Minimum XOR of $(x \vee y) \oplus (x \wedge y)$ is equal to the minimum XOR of all pairs in the array, where $\vee$ denotes the OR operation and $\wedge$ denotes the AND operation.

Sum of XOR of all elements of subsets

1. $2^{n-1} \times$ (OR of whole array), where n is the number of elements in the array.

Maximum OR pair

= Maximum element $\vee$ next greater element (excluding itself)

Sum of OR of all subsets

$= \sum_{i=0}^{n-1} a_i \cdot 2^{n-1}$, where a_i is the i-th element in the array and n is the number of elements.

10.5 Number of Subarrays with XOR 0

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

const int N = 1e6 + 9;
int a[N], p[N];

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
    p[i] = p[i - 1] ^ a[i];
  }
  map<int, int> mp;
  long long ans = 0;
  mp[p[0]]++; // don't forget to add this (why?)
  for (int i = 1; i <= n; i++) {
    ans += mp[p[i]];
    mp[p[i]]++;
  }
  cout << ans << '\n';
  return 0;
}
```

10.6 Sum of XOR of All subset in Array

```
int xorSum(int arr[], int n) {
    int bits = 0;
    for (int i=0; i < n; ++i) bits |= arr[i];
    int ans = bits * pow(2, n-1);
    return ans;
}
```

10.7 Sum of all and of all subset

```
ans = 0;
for(i=0; i<32; i++) {
    cnt=0;
    for(j=0; j<n; j++) {
        if(aj&(1<<i)cnt++;
    }
    subsets = (1<<cnt)-1;
    subsets = subset^(1<<i) ans += subset;
}
```

11 J.Misc

11.1 Base conversion

```
#pragma once
string const DIGITS = "0123456789ABCDEF";
```

```
unsigned long long int
    basis_string_to_number(string &s, int b) {
    unsigned long long int result = 0ULL;
    for (char d : s) {
        result = b * result
                + (find(DIGITS.begin(),
                    DIGITS.end(), d) -
                    DIGITS.begin());
    }
    return result;
}
string number_to_basis_string(unsigned long long
    int n, int b) {
    vector<char> ds;
    do {
        ds.push_back(DIGITS[n % b]);
        n = n / b;
    } while (n != 0);
    return string(ds.rbegin(), ds.rend());
}
```

11.2 Equation Solve

```
// a1*x + b1*y + c1*z = d1 | a2*x + b2*y + c2*z =
    d2 | a3*x + b3*y + c3*z = d3
bool eq3(double &x, double &y, double &z,
        double a1, double b1, double c1, double p,
        double a2, double b2, double c2, double q,
        double a3, double b3, double c3, double
            r) {
    double detr = a1 * (b2 * c3 - c2 * b3) - b1 *
        (a2 * c3 - c2 * a3) + c1 * (a2 * b3 - b2
        * a3);
    if (fabs(detr) < 1e-9) return false;
    x = (p * (b2 * c3 - c2 * b3) + q * (c1 * b3 -
        b1 * c3) + r * (b1 * c2 - c1 * b2)) /
        detr;
    y = (p * (c2 * a3 - a2 * c3) + q * (a1 * c3 -
        c1 * a3) + r * (c1 * a2 - a1 * c2)) /
        detr;
    z = (p * (a2 * b3 - b2 * a3) + q * (b1 * a3 -
        a1 * b3) + r * (a1 * b2 - b1 * a2)) /
        detr;
    return true;
}

// a1*x + b1*y = c1 | a2*x + b2*y = c2
bool eq2(double &x, double &y,
        double a, double b, double p,
        double c, double d, double q) {
    double det = a * d - b * c;
    if (fabs(det) < 1e-9) return false; // Check
        for near-zero determinant
    x = (p * d - b * q) / det;
    y = (a * q - c * p) / det;
    return true;
}
```

11.3 LCS - longest common subsequences

```
int L[m + 1][n + 1];
for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
        if (i == 0 || j == 0) L[i][j] = 0;
        else if (X[i - 1] == Y[j - 1]) L[i][j] =
            L[i - 1][j -1] + 1;
        else L[i][j] = max(L[i - 1][j], L[i][j -
            1]);
    }
}
```

11.4 LIS - Longest Increasing Subsequence

```
// LIS - Longest increasing subsequence
// O(n*logn)
const int INF=INT_MAX;
vector<int> LIS(vector<int> & inp){
    int n=inp.size();
    vector<int> a(n+2, INF);
    vector<int> res(n,1);
    a[0]=-INF;
    int lis=0;
    for(int i=0; i<n; i++){
        int low = lower_bound(a.begin(), a.end(),
            inp[i])-a.begin();
        a[low] = inp[i];
        lis = low;
        res[i] = lis;
    }
    return res;
}
```

11.5 Mex of all Subarray

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 9, inf = 1e9;
struct ST {
    int t[4 * N];
    ST() {}
    void build(int n, int b, int e) {
        t[n] = 0;
        if (b == e) {
            return;
        }
        int mid = (b + e) >> 1, l = n << 1, r = l
            | 1;
        build(l, b, mid);
        build(r, mid + 1, e);
        t[n] = min(t[l], t[r]);
    }
    void upd(int n, int b, int e, int i, int x) {
        if (b > i || e < i) return;
        if (b == e && b == i) {
            t[n] = x;
            return;
        }
        int mid = (b + e) >> 1, l = n << 1, r = l
            | 1;
        upd(l, b, mid, i, x);
        upd(r, mid + 1, e, i, x);
```

```
        t[n] = min(t[l], t[r]);
    }
    int get_min(int n, int b, int e, int i, int
        j) {
        if (b > j || e < i) return inf;
        if (b >= i && e <= j) return t[n];
        int mid = (b + e) >> 1, l = n << 1, r = l
            | 1;
        int L = get_min(l, b, mid, i, j);
        int R = get_min(r, mid + 1, e, i, j);
        return min(L, R);
    }
    int get_mex(int n, int b, int e, int i) { //
        mex of [i... cur_id]
        if (b == e) return b;
        int mid = (b + e) >> 1, l = n << 1, r = l
            | 1;
        if (t[l] >= i) return get_mex(r, mid + 1,
            e, i);
        return get_mex(l, b, mid, i);
    }
} t;
int a[N], f[N];
int32_t main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        --a[i];
    }
    t.build(1, 0, n);
    set<array<int, 3>> seg; // for cur_id = i,
        [x[0]...i], [x[0] + 1...i], ...
        [x[1]...i] has mex x[2]
    for (int i = 1; i <= n; i++) {
        int x = a[i];
        int r = min(i - 1, t.get_min(1, 0, n, 0, x
            - 1));
        int l = t.get_min(1, 0, n, 0, x) + 1;
        if (l <= r) {
            auto it = seg.lower_bound({l, -1, -1});
            while (it != seg.end() && (*it)[1] <=
                r) {
                auto x = *it;
                it = seg.erase(it);
            }
        }
        t.upd(1, 0, n, x, i);
        for (int j = r; j >= l; ) {
            int m = t.get_mex(1, 0, n, j);
            int L = max(l, t.get_min(1, 0, n, 0,
                m) + 1);
            f[m] = 1;
            seg.insert({L, j, m});
            j = L - 1;
        }
        int m = !a[i];
        seg.insert({i, i, m});
        f[m] = 1;
    }
    int ans = 0;
    while (f[ans]) ++ans;
    cout << ans + 1 << '\n';
    return 0;
}
```

11.6 Mo's Algorithm

```
const int M = 1e9+7;
int n, q;
vector<int> v;
struct query{int l,r,idx;};
int block;
bool comp1(query p,query q){
    if (p.l / block != q.l / block) {
        if(p.l==q.l) return p.r<q.r;
        return p.l < q.l;
    }
    return (p.l / block & 1) ? (p.r < q.r) : (p.r
        > q.r);
}
vector<int> fre(1000001, 0);
int no_of_distinct=0;
void add(int idx){
    fre[v[idx]]++;
    if(fre[v[idx]]==1) no_of_distinct++;
}
void rmv(int idx){
    fre[v[idx]]--;
    if(fre[v[idx]]==0) no_of_distinct--;
}
int get_answer(){
    return no_of_distinct;
}
void mos_algorithm(int n, vector<query>&queries){
    vector<int> answers(queries.size());
    block = (int)sqrt(n);
    sort(queries.begin(), queries.end(),comp1);
    int cur_l = 0;
    int cur_r = -1;
    for (query q : queries) {
        while (cur_l > q.l) {cur_l--; add(cur_l);}
        while (cur_r < q.r) {cur_r++; add(cur_r);}
        while (cur_l < q.l) {rmv(cur_l);cur_l++;}
        while (cur_r > q.r) {rmv(cur_r);cur_r--;}
        answers[q.idx] = get_answer();
    }
    for(int i:answers) {cout<<i<<"\n";}
}
void srfahad2021(){
    cin>>n;
    v.assign(n, 0);
    for(int i=0; i<n; i++){
        cin>>v[i];
    }
    vector<query> queries;
    cin>>q;
    for(int i=0; i<q; i++){
        int l, r; cin>>l>>r;
        l--; r--;
        queries.push_back({l, r, i});
    }
    mos_algorithm(q, queries);
}
int32_t main(){
    fast();
    int test=1;
    // cin>>test;
    for(int i=1; i<=test; i++){
        // cout<<"Case "<<i<<": ";
```

```
        srfahad2021();
    }
    return 0;
}
```

11.7 Nim Game 2d

```
int main() {
    int t;
    scanf("%d",&t);
    FOR(tc,1,t) {
        int r,c;
        scanf("%d %d",&r,&c);
        int nim=0;
        FOR(i,1,r) {
            FOR(j,1,c) {
                int tmp;
                scanf("%d",&tmp);
                if(((r-i)+(c-j))%2) {
                    nim^=tmp;
                }
            }
        }
        if(nim) printf("Case %d: win\n",tc);
        else printf("Case %d: lose\n",tc);
    }
}
```

11.8 Number of Subsegment Equal to K

```
int a[lx], t[lx];
int n, k;
map<ll, int> mp;
int o[lx], Plus[lx], Minus[lx], cnt[lx];
int l, r;
ll p[lx];
const int block = 320;
struct Query {
        int l, r, idx;
        bool operator < (Query q) const {
                int b1 = l / block;
                int b2 = q.l / block;

                if(b1 == b2) return r < q.r;
                return b1 < b2;
        }
};
ll res[lx], ans = 0;
ll query(int x, int y)
{
        while(x < l) {
                --l;
                ans += cnt[Plus[l]];
                cnt[o[l]]++;
        }
        while(r < y) {
                ++r;
                ans += cnt[Minus[r]];
                cnt[o[r]]++;
        }

        while(l < x) {
                cnt[o[l]]--;
                ans -= cnt[Plus[l]];
                l++;
        }
        while(y < r) {
                cnt[o[r]]--;
                ans -= cnt[Minus[r]];
                r--;
        }
        return ans;
}
void solve()
{
        cin >> n >> k;
        for(int i = 1; i <= n; i++) cin >> t[i];
        for(int i = 1; i <= n; i++) cin >> a[i];
        for(int i = 1; i <= n; i++) {
                if(t[i] == 1) p[i] = p[i - 1] +
                    a[i];
                else p[i] = p[i - 1] - a[i];
        }
        for(int i = 0; i <= n; i++) {
                mp[p[i]];
                mp[p[i] + k];
                mp[p[i] - k];
        }
        int idx = 0;
        for(auto &it : mp) {
                it.second = ++idx;
        }
        for(int i = 0; i <= n; i++) {
                o[i] = mp[p[i]];
                Plus[i] = mp[p[i] + k];
                Minus[i] = mp[p[i] - k];
        }
        cnt[o[0]]++;
        l = 0; r = 0;
        int qr;
        vector<Query> Q;
        cin >> qr;
        for(int i = 0; i < qr; i++) {
                int x, y;
                cin >> x >> y;
                Query q;
                q.l = x - 1;
                q.r = y;
                q.idx = i;
                Q.push_back(q);
        }

        sort(Q.begin(), Q.end());
        for(auto it : Q) {
                res[it.idx] = query(it.l, it.r);
        }
        for(int i = 0; i < qr; i++) cout << res[i]
            << endl;
}
```

11.9 Ternary Search

```
double ternary_search(double l, double r) {
    double eps = 1e-9; //set the error limit here
```

```
while (r - l > eps) {
    double m1 = l + (r - l) / 3;
    double m2 = r - (r - l) / 3;
    double f1 = f(m1);//evaluates the function
        at m1
    double f2 = f(m2); //evaluates the
        function at m2
    if (f1 < f2)
        l = m1;
    else
        r = m2;
}
return f(l);//return the maximum of f(x) in
    [l, r]
```

```
}
```

11.10 Tower of Hanoi

```
void solve(int count, char source, char
    destination, char intermediate) {
    if (count == 1)
    printf("Move top disc from pole %c to pole
        %c\n", source, destination);
    else {
    solve(count-1, source, intermediate,
        destination);
    solve(1, source, destination, intermediate);
    solve(count-1, intermediate, destination,
        source);
    }
    }
// bitset
bitset<17>BS;
BS[1] = BS[7] = 1;
cout<<BS._Find_next(1)<<','<<BS._Find_next(3)<<endl;
    // prints 7,7
```