Team notebook

International University of Business Agriculture and Technology

December 31, 2024

Contents

1 A.Settings and Script

1.1 Generator

```cpp
#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

```bash
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 Template

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

#pragma GCC target("avx2")
#pragma GCC optimization("O3")
#pragma GCC optimization("unroll-loops")

typedef long long ll;
typedef long double ld;

#define pi acos(-1)
#define prDouble(x, y) fixed << setprecision(y) << x
#define nl << "\n"
#define lcm(a, b) (a * b) / __gcd(a, b)
#define getBit(n, i) (n & (1 << i))
#define setBit(n, i) (n | (1 << i))
#define isPowerOfTwo(n) (n && !(n & (n - 1)))
#define MultiTestCase


void solve(int test_case)
{
}

void file_io()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    freopen("error.txt", "w", stderr);
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    file_io();

    int test_cases = 1;
#ifdef MultiTestCase
    cin >> test_cases;
#endif

    for (int test_case = 1; test_case <= test_cases;
         test_case++)
    {
        solve(test_case);
    }

    return 0;
}

// Compile command - g++ -std=c++17 -O2 -Wno-unused-result
     -static filename.cpp -o filename
```

1.4 settings

```
// 1. Layout - 2 Column Split Row
// 2. Wrap - On, Warping Indent - True
// 3. AutoSave - After Delay
// 4. Sticky Scroll - Off
// 5. gedit ~/.bashrc -> ulimit -s 2000123
```

2 B.Combinatorics

2.1 Binomial coefficient

```cpp
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 Inverse Modulo Using Extended Euclidean Algorithm

```
int Extended_Euclidean(int a,int b,int &x,int &y) {
    if(b==0) {
        x=1;
        y=0;
         return  a;
    }
    int d=Extended_Euclidean(b,a%b,y,x);
    y=y-(a/b)*x;
    return d;
}
int Inverse_Modulo(int a,int m) {
    int x,y,d=Extended_Euclidean(a,m,x,y);
    if(d==1)     return (x+m)%m;
}
```

2.7 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;
}
```

2.8 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;
}
```

2.9 Modular Inverse

```
long long poww(long long a,long long b)
{
    if(b==0)
        return 1;
    long long r=poww(a,b/2);
    r=(r*r)%mod;
    if(b&1)
    {
        r=(r*a)%mod;
    }
    return r;
}
long long inv(long long n)
{
    return poww(n,mod-2);
}
```

2.10 Number of Trailing Zeroes of N Factorial Base B

```
#define SIZE_N 1000 #define SIZE_P 1000
bool flag[SIZE_N+5];
int primes[SIZE_P+5];
int seive() {
    int i,j,total=0,val;
    for(i=2; i<=SIZE_N; i++) flag[i]=1;
      val  =sqrt(SIZE_N)+1;
    for(i=2; i<val; i++) if(flag[i]) for(j=i; j*i<=SIZE_N; j++)
          flag[i*j]=0;
    for(i=2; i<=SIZE_N; i++) if(flag[i]) primes[total++]=i;
    return total;
}
int factors_in_factorial(int N,int p) {
    int sum=0;
    while(N) {
        sum+=N/p;
        N/=p;
    }
    return sum;
}
int Trailingzero_Base_B(int N,int B) {
    int i,ans,freq,power;
    ans=1000000000;
    for(i=0; primes[i]<=B; i++) {
        if(B%primes[i]==0) {
            freq=0;
            while(B%primes[i]==0) {
                freq++;
                B/=primes[i];
            }
            power=factors_in_factorial(N,primes[i]);
            ans=min(ans,power/freq);
        }
    }
    return ans;
}
int main() {
    int total=seive();
    int i,N,B,zero;
    while(scanf("%d %d",&N,&B)==2) {
        zero=Trailingzero_Base_B(N,B);
        printf("%d\n",zero);
    }
    return 0;
}
```

2.11 ^nC_r *and* ^nP_r

```
#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;
}
```

2.12 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 Convex Hull

```
Convex Hull
struct point {
    LL x,y;
    bool operator < (const point &p) const {
        return x<p.x || (x==p.x && y<p.y);
    }
} P[MAX],C[MAX];
inline LL Cross(point &o,point &a,point &b) {
    return (a.x-o.x)*(b.y-o.y)-(a.y-o.y)*(b.x-o.x);
}
void ConvexHull(int np,int &nc) {
    sort(P,P+np);
    REP(i,np) {
        while(nc>=2 and Cross(C[nc-2],C[nc-1],P[i])<=0)
            nc--;
        C[nc++]=P[i];
    }
    int t=nc+1;
    ROF(i,np-1,1) {
        while(nc>=t and Cross(C[nc-2],C[nc-1],P[i-1])<=0)
            nc--;
        C[nc++]=P[i-1];
    }
    nc--;
    return;
}
int main() {
    int nc=0,np;
    scanf("%d",&np);
    REP(i,np) {
        scanf("%lld %lld",&P[i].x,&P[i].y);
    }
    ConvexHull(np,nc);
    REP(i,nc) {
        printf("%lld %lld\n",C[i].x,C[i].y);
    }
    return 0;
}
```

3.2 Geometry All Template

```
#define MAXD 4
#define eps 1e-9
double cosineRule3Side ( double a, double b, double c ) {
    double res = (SQ(a)+SQ(b)-SQ(c)) / (2*a*b);
    if ( res < -1 ) res = -1;
    if ( res > 1 ) res = 1;
    return acos ( res );
}
struct myVec {
    int d; //Dimension
    double val[MAXD];//Contains value of each component
    myVec add ( myVec b ) {
        myVec res;
        FOR(i,0,d) res.val[i] = val[i] + b.val[i];
        return res;
    }
    myVec sub ( myVec b ) {
        myVec res;
        FOR(i,0,d) res.val[i] = val[i] - b.val[i];
        return res;
    }
    myVec mul ( double t ) {
        myVec res;
        FOR(i,0,d)res.val[i] = val[i] * t;
        return res;
    }
    myVec div ( double t ) {
        myVec res;
        FOR(i,0,d) res.val[i] = val[i] / t;
        return res;
    }
    bool operator == ( myVec b ) {
        FOR(i,0,d) if ( fabs ( val[i] - b.val[i] ) > eps )
              return false;
        return true;
    }
    myVec perp2D() {
        myVec res = (*this);
        swap ( res.val[0], res.val[1] );
        res.val[0] *= -1;
        return res;
    }
    double dot ( myVec v ) { //Finds *this (dot) v
        double res = 0;
        for ( int i = 0; i < d; i++ ) res += val[i] * v.val[i];
        return res;
    }
    double length () { //Finds length of current vector
        return sqrt ( this->dot( *this ) );
    }
    myVec unitVec () {
        return (*this).div ( length() ); // v / ||v||
    }
    double angleBetween ( myVec b ) { //Angle between two
          vectors
        double res = dot( b ) / ( length() * b.length() );
        if ( res > 1 ) res = 1;
        if ( res < -1 ) res = -1;
        return acos (res);
    }
    double polarAngle2D() { //Angle from x-axis
        double res = atan2 ( val[1], val[0] );
        if ( res + eps < 0 ) res += 2 * pi;
        return res;
    }
    double cross2D ( myVec v ) { //Cross the two values. Only
          for 2D. Z compo 0.
        return val[0]*v.val[1] - val[1]*v.val[0];
    }
};
struct myLine {
    myVec a, b; //a is displacement, b is direction.
    //Builds a line from two points
    myLine lineFromPoints ( myVec x, myVec y ) {
        myLine m;
        m.a = x;
        m.b = y.sub ( x );
        return m;
    }
    //Finds point on line, given t.
    myVec atPos ( double t ) {
        return a.add ( b.mul ( t ) ); // a + tb;
    }
```

```
    double lineToPointDistance ( myVec p, double t ) {
        p = p.sub ( a ); //Take it to origin
        t = b.dot ( p ) / ( b.length() * b.length() ); //point
              of intersection
        myVec x = b.mul ( t ); //tb
        return ( p.sub(x).length() ); //xp length()
    }
    double segmentToPointDistance ( myVec p, double &t ) {
        p = p.sub ( a ); //Take it to origin
        t = b.dot ( p ) / ( b.length() * b.length() );
        if ( t + eps < 0 || t > 1 + eps ) { //Not on segment
            return min ( p.length(), p.sub(b).length() );
        }
        myVec x = b.mul ( t ); //tb
        return ( p.sub(x).length() ); //xp length()
    }
    bool overlapParallel ( myLine l ) {
        double p, q, r, s;
        if ( b.val[0] == 0 ) {
            p = a.val[1];
            q = atPos(1).val[1];
            r = l.a.val[1];
            s = l.atPos ( 1 ).val[1];
            if ( min ( r, s ) > max ( p, q ) ) return false;
            if ( max ( r, s ) < min ( p, q ) ) return false;
            return true;
        }
        else {
            p = a.val[0];
            q = atPos(1).val[0];
            r = l.a.val[0];
            s = l.atPos ( 1 ).val[0];
            if ( min ( r, s ) > max ( p, q ) ) return false;
            if ( max ( r, s ) < min ( p, q ) ) return false;
            return true;
        }
    }
    char lineAndLineIntersection2D ( myLine l, double &t,
          double &s ) {
        if ( b.cross2D ( l.b) == 0 ) {
            if ( l.a.sub(a).cross2D(l.b) == 0 ) {
                if ( overlapParallel ( l ) ) return 'o';
                      //overlaps
                else return 'p'; //parallel
            }
            else return 'd'; //disjoint and parallel
        }
        myVec w = a.sub ( l.a );
        myVec p = l.b.perp2D(), z = b.perp2D();
        t = -(w.dot(p))/p.dot(b); //for current line
        s = w.dot(z)/z.dot(l.b); //for line l
        return 'i';
    }
    double lineAndLineDistance2D ( myLine l ) {
        double t, s; //First check if the intersect
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r == 'i' ) return 0; //Intersects. 0 distance.
        //Parallel Lines
        return lineToPointDistance ( l.a, t );
    }
```

```
    double lineAndSegmentDistance2D ( myLine l ) {
        double t, s;
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r == 'i' && s + eps > 0 && s < 1 + eps ) {
            return 0; //Valid intersection
        }
        double res = lineToPointDistance ( l.a, t );
        res = min ( res, lineToPointDistance ( l.a.add(l.b), t
              ) );
        return res;
    }
    double segmentAndSegmentDistance2D ( myLine l ) {
        double t, s;
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r =='i' && t+eps > 0 && t < 1 + eps && s + eps > 0
              && s < 1 + eps ) {
            return 0; //Valid intersection
        }
        double res = segmentToPointDistance ( l.a, t );
        res = min ( res, segmentToPointDistance ( l.a.add(l.b),
              t ) );
        res = min ( res, l.segmentToPointDistance ( a, t ) );
        res = min ( res, l.segmentToPointDistance ( a.add ( b
              ), t ) );
        return res;
    }
    myLine reflect ( myVec p, myVec norm ) {
        myVec ap = p.sub ( a ); //Starting to Point of
              Reflection
        norm = norm.unitVec();

        double d = fabs ( ap.dot ( norm ) );

        myVec m = p.add ( norm.mul ( d ) );
        myVec h = m.sub ( a ).mul ( 2 );
        m = a.add ( h );

        myLine ray = ray.lineFromPoints ( p, m );
        return ray;
    }
};
struct myCir {
    myVec a;
    double r;
    myVec atPos ( double t ) {
        myVec res;
        res.val[0] = a.val[0] + r * cos ( t );
        res.val[1] = a.val[1] + r * sin ( t );
        return res;
    }
    char circleAndLineIntersection2D ( myLine l, double &t1,
          double &t2 ) {
        double t3;
        double d = l.lineToPointDistance ( a, t3 );
        if ( d > r + eps ) return 'd';
        if ( fabs ( d - r ) <= eps ) return 't';
        myVec m = l.atPos ( t3 );
        myVec am = m.sub ( a );
        //Need to handle when line passes through center
        double x = am.polarAngle2D();
    double temp = d / r;
    if ( temp > 1 ) temp = 1;
    if ( temp < -1 ) temp = -1;
    double theta = pi / 2 - asin ( temp ); //Using sin law
        find internal angle.
    t1 = x + theta;
    t2 = x - theta;
    return 'i';
}
char sphereAndLineIntersect ( myLine l, double &t1, double
     &t2 ) {
    double tp = 0;
    double d = l.lineToPointDistance ( a, tp );
    if ( d > r + eps ) return 'd';
    if ( fabs ( d - r ) < eps ) {
        t1 = tp;
        return 't';
    }
    double chord = sqrt ( r * r - d * d );
    t1 = tp - chord / l.b.length();
    t2 = tp + chord / l.b.length();
    return 'i';
}
char circleAndCircleIntersection2D ( myCir c2, double &t1,
     double &t2 ) {
    myVec d = c2.a.sub ( a );
    if ( d.length() > r + c2.r + eps ) return 'd'; //Case 1
    if ( d.length() + c2.r + eps < r ) return 'd'; //Case 2
    if ( a == c2.a && fabs ( r - c2.r ) <= eps ) {
        if ( r == 0 ) {
            t1 = 0;
            return 't'; //Case 7
        }
        return 's'; //Case 6
    }
    if ( fabs ( d.length() - r - c2.r ) <= eps ||
            fabs ( d.length() + c2.r - r ) <= eps ) {
        t1 = d.polarAngle2D();
        return 't'; //Case 3 and 4
    }
    double theta = cosineRule3Side ( r, d.length(), c2.r );
    double m = d.polarAngle2D ();
    t1 = m - theta;
    t2 = m + theta;
    return 'i'; //Case 5
}
int circleToCircleTangentLine (myCir c2,myLine &l1,myLine
     &l2,myLine &l3,myLine &l4) {
    //First circle must be smaller or equal to second circle
    if (r>c2.r + eps ) return c2.circleToCircleTangentLine
         ( *this, l1, l2, l3, l4 );
    myVec oo = c2.a.sub ( a );
    double d = oo.length();

    if ( fabs ( d ) < eps && fabs ( r - c2.r ) < eps )
         //Infinite tangents
        return -1;
    if ( d + r + eps < c2.r ) //No tangents
        return 0;
    double base = oo.polarAngle2D();
        if ( fabs ( d + r - c2.r ) < eps ) { //Contains Circle
            l1 = l1.lineFromPoints ( atPos ( base + pi ), atPos
                 ( base + pi ) );
            return 1;

            double ang = pi - acos ( (c2.r - r ) / d );
            l1 = l1.lineFromPoints ( atPos ( base + ang ),
                 c2.atPos ( base + ang ) );
            l2 = l2.lineFromPoints ( atPos ( base - ang ),
                 c2.atPos ( base - ang ) );

            if ( d + eps < r + c2.r ) return 2; //Circle
                 intersects

            if ( fabs ( d - r - c2.r ) < eps ) { //Circle tangent
                l3 = l3.lineFromPoints ( atPos ( base ), atPos (
                     base ) );
                return 3;
            }
            //Disjoint Circle
            ang = acos ( ( c2.r + r ) / d );
            l3 = l3.lineFromPoints ( atPos ( base + ang ),
                 c2.atPos ( base + ang + pi ) );
            l4 = l4.lineFromPoints ( atPos ( base - ang ),
                 c2.atPos ( base - ang + pi ) );

            return 4;
        }
    };
    bool collinear ( myVec a, myVec b, myVec c ) {
        myVec ab = b.sub(a), ac = c.sub(a);
        double d = fabs ( ab.dot(ac) );
        if ( fabs ( d - ab.length() * ac.length() ) <= eps )
            return true;
        return false;
    }
```

3.3 Line And Point

```
struct pt {
    double x, y;
};
struct line {
    double a, b, c;
};
const double EPS = 1e-9;
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;
}
```

3.4 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);
```

3.5 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) + $1\frac{}{2\times(\text{perimeter of base})\times(\text{slant height})`-Slantheight:`\sqrt{h^2+r^2}`-Volume:`\frac{1}{3}\times(\text{ba}}$

5. Right Circular Cone: - Surface area: $‘1\frac{}{2\times(\text{base circumference})\times(\text{slant height})+\pi\times r\times(r+l)‘-Volume:‘\frac{1}{3}\times(\text{base area})\times(\text{height})‘}$

6. Sphere: - Surface area: $‘4\ \times\pi\times r^{2}‘-Volume:‘\frac{4}{3}\times\pi\times r^{3}‘-Radiusofthecircleformedbyasectionwithheight‘h‘:‘r_{h}=\sqrt{r^{2}-h^{2}}‘$

7. Cylinder: - Total surface area: $‘2\ \times\pi\times r\times(r+h)‘-Curvedsurfacearea:‘2\times\pi\times r\times h‘-Volume:‘\pi\times r^{2}\times h‘$

8. Cone: - Curved surface area: $‘\pi\times r\times l‘-Totalsurfacearea:‘\pi\times r\times(r+l)‘-Volume:‘\frac{1}{3}\times\pi\times r^{2}\times h‘$

9. Frustum of a Cone: - Curved surface area: $‘\pi\times(r_{1}+r_{2})\times l‘-Totalsurfacearea:‘\pi\times(r_{1}+r_{2})\times l+\pi\times r_{1}^{2}+\pi\times r_{2}^{2}‘-Volume:‘\frac{1}{3}\times\pi\times h\times(r_{1}^{2}+r_{2}^{2}+r_{1}\times r_{2})‘$

10. Hemisphere: - Curved surface area: $‘2\ \times\pi\times r^{2}‘-Totalsurfacearea:‘3\times\pi\times r^{2}‘-Volume:‘\frac{2}{3}\times\pi\times r^{3}‘$

11. Torus: - Surface area: $‘4\ \times\pi^{2}\times Rr‘-Volume:‘2\times\pi^{2}\times Rr^{2}‘$

12. Ellipsoid: - Surface area: $‘4\ \times\pi\sqrt{\frac{(a^{2}\cdot b^{2}+b^{2}\cdot c^{2}+a^{2}\cdot c^{2})}{3}}‘-Volume:‘\frac{4}{3}\times\pi\times a\times b\times c‘$

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 Find the sum of Binomial Coefficient

```
// Returns value of Binomial Coefficient Sum
int binomialCoeffSum(int n)
{
    int C[n + 1][n + 1];
    // Calculate value of Binomial Coefficient
    // in bottom up manner
    for (int i = 0; i <= n; i++) {
        for (int j = 0; j <= min(i, n); j++) {
            // Base Cases
            if (j == 0 || j == i)
                C[i][j] = 1;
            // Calculate value using previously
            // stored values
            else
                C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
        }
    }
    // Calculating the sum.
    int sum = 0;
    for (int i = 0; i <= n; i++)
        sum += C[n][i];
    return sum;
}
```

4.5 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 $\gcd(a_1 a_2, b) = \gcd(a_1, b)\gcd(a_2, b)$.
- $\gcd(a, b) \cdot \operatorname{lcm}[a, b] = |ab|$
- $\gcd(a, \operatorname{lcm}[b, c]) = \operatorname{lcm}[\gcd(a, b), \gcd(a, c)]$
- $\operatorname{lcm}[a, \gcd(b, c)] = \gcd(\operatorname{lcm}[a, b], \operatorname{lcm}[a, c])$
- For non-negative integers a and b, where a and b are not both zero,
$$\gcd(n1, n1) = n\gcd(a, b)$$
- $\gcd(a, b) = \dfrac{kla \cdot kib}{k}$
- $\gcd(i, n) = k] = \dfrac{\varphi(n)}{k} \sum_{i=1}^{n} \alpha(i)$
- $\gcd(k, n) = \sum_{d|n} \alpha(d)\varphi(\frac{n}{d})$
- $\sum_{k=1}^{n} \frac{1}{gcd(k,n)} = \sum_{d|n} \frac{1}{d} \cdot \phi(\frac{n}{d}) = \frac{1}{n} \sum_{d|n} d \cdot \phi(d)$
- $\sum_{k=1}^{n} \frac{k}{gcd(k,n)} = \frac{n}{2} \cdot \sum_{d|n} \frac{1}{d} \cdot \phi(\frac{n}{d}) = \frac{n}{2} \cdot \frac{1}{n} \cdot \sum_{d|n} d \cdot \phi(d)$
- $\sum_{k=1}^{n} \frac{n}{gcd(k,n)} = 2 * \sum_{k=1}^{n} \frac{k}{gcd(k,n)} - 1, for\ n > 1$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} [gcd(i, j) = 1] = \sum_{d=1}^{n} \mu(d) \lfloor \frac{n}{d} \rfloor^2$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} gcd(i, j) = \sum_{d=1}^{n} \phi(d) \lfloor \frac{n}{d} \rfloor^2$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} i \cdot j[gcd(i, j) = 1] = \sum_{i=1}^{n} \phi(i) i^2$
- $F(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i, j) = \sum_{l=1}^{n} (\frac{(1+\lfloor \frac{n}{l} \rfloor)(\lfloor \frac{n}{l} \rfloor)}{2})^2 \sum_{d|l} \mu(d) ld$
- $gcd(lcm(a, b), lcm(b, c), lcm(a, c)) = lcm(gcd(a, b), gcd(b, c), gcd(a, c))$
- $gcd(A_L, A_{L+1}, ..., A_R) = gcd(A_L, A_{L+1} - A_L, ..., A_R - A_{R-1})$
- $Givenn, IfSUM = LCM(1, n) + LCM(2, n) + ... + LCM(n, n)\ thenSUM = n2((d)xd) + 1$
- $xed(k) = x.()$

4.6 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 Linear Seive

```
vector<int> sieve(const int N, const int Q = 17, const int L =
    1 << 15) {
    static const int rs[] = {1, 7, 11, 13, 17, 19, 23, 29};
    struct P {
        P(int p) : p(p) {}
        int p;
        int pos[8];
    };
    auto approx_prime_count = [] (const int N) -> int {
        return N > 60184 ? N / (log(N) - 1.1)
        : max(1., N / (log(N) - 1.11)) + 1;
    };

    const int v = sqrt(N), vv = sqrt(v);
    vector<bool> isp(v + 1, true);
    for (int i = 2; i <= vv; ++i) if (isp[i]) {
            for (int j = i * i; j <= v; j += i) isp[j] = false;
        }

    const int rsize = approx_prime_count(N + 30);
    vector<int> primes = {2, 3, 5};
    int psize = 3;
    primes.resize(rsize);

    vector<P> sprimes;
    size_t pbeg = 0;
    int prod = 1;
    for (int p = 7; p <= v; ++p) {
        if (!isp[p]) continue;
        if (p <= Q) prod *= p, ++pbeg, primes[psize++] = p;
        auto pp = P(p);
        for (int t = 0; t < 8; ++t) {
            int j = (p <= Q) ? p : p * p;
            while (j % 30 != rs[t]) j += p << 1;
            pp.pos[t] = j / 30;
        }
        sprimes.push_back(pp);
    }

    vector<unsigned char> pre(prod, 0xFF);
    for (size_t pi = 0; pi < pbeg; ++pi) {
        auto pp = sprimes[pi];
        const int p = pp.p;
        for (int t = 0; t < 8; ++t) {
            const unsigned char m = ~(1 << t);
            for (int i = pp.pos[t]; i < prod; i += p) pre[i] &=
                m;
        }
    }

    const int block_size = (L + prod - 1) / prod * prod;
    vector<unsigned char> block(block_size);
    unsigned char* pblock = block.data();
    const int M = (N + 29) / 30;

    for (int beg = 0; beg < M; beg += block_size, pblock -=
        block_size) {
        int end = min(M, beg + block_size);
        for (int i = beg; i < end; i += prod) {
            copy(pre.begin(), pre.end(), pblock + i);
        }
        if (beg == 0) pblock[0] &= 0xFE;
        for (size_t pi = pbeg; pi < sprimes.size(); ++pi) {
            auto& pp = sprimes[pi];
            const int p = pp.p;
            for (int t = 0; t < 8; ++t) {
                int i = pp.pos[t];
                const unsigned char m = ~(1 << t);
                for (; i < end; i += p) pblock[i] &= m;
                pp.pos[t] = i;
            }
        }
        for (int i = beg; i < end; ++i) {
            for (int m = pblock[i]; m > 0; m &= m - 1) {
                primes[psize++] = i * 30 + rs[__builtin_ctz(m)];
            }
        }
    }
    assert(psize <= rsize);
    while (psize > 0 && primes[psize - 1] > N) --psize;
    primes.resize(psize);
    return primes;
}
```

4.8 Prime Factorization of a Factorial Number

```
int factorial_factorization(int n) {
    for (int i = 2; i <= n; i++) {
        int a = i;
        for (int j = 0; j < prime.size(); j++) {
            if(prime[j] > i) break;
            int cnt = 0;
            while (a % prime[j] == 0) {
                a /= prime[j];
                cnt++;
            }
            m[prime[j]] += cnt;
        }
    }
}
/*Note:
Prime factorization of a factorial n means you have to do prime
      factorization of every number 1 to n. And the merge that*/
```

4.9 Total Digit of N factorial

4.9.1 Find of total digit of n! in b base number system

```
int factorialDigitExtended(int n,int base){
    double x=0;
    for (int i = 1; i <= n; i++) {
        x += log10(i)/log10(base); // Base Conversion
    }
    int res =x+1+eps;
    return res;
}
```

4.9.2 find Total Digit

```
int factorialDigit ( int n ) {
    double x = 0;
    for ( int i = 1; i <= n; i++ ) {
        x += log10 ( i );
    }
    int res = x + 1 + eps;
    return res;
}
```

4.9.3 note

- In 10 based number system total digits of a number n will be $n + 1$.
  - Number of digits in a number = $log10(n) + 1 + eps$ where $eps = 10^{-9}$
  - Number of digits of x in base $B = logB(x) + 1 + eps$
  - Similarly in b based number system total digits of a number n will be $n + 1$. So, $log_b n = log_{10} n \times log_b 10$ $log_b n = \frac{log_{10} n}{log_{10} b}$

4.10 chinese remainder theorem

```
int gcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}
bool find_any_solution(int a, int b, int c, int &x0, int &y0,
     int &g) {
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g) {
        return false;
    }
    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}
```

5 Data Structures

5.1 Hashing

```
// Hashvalue(l...r) = hsh[l] - hsh[r + 1] * base ^ (r - l + 1);
// Must call preprocess
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAX = 100009;
ll mods[2] = {1000000007, 1000000009};
//Some back-up primes: 1072857881, 1066517951, 1040160883
ll bases[2] = {137, 281};
ll pwbase[3][MAX];

void Preprocess(){
    pwbase[0][0] = pwbase[1][0] = 1;
    for(ll i = 0; i < 2; i++){
        for(ll j = 1; j < MAX; j++){
            pwbase[i][j] = (pwbase[i][j - 1] * bases[i]) %
                  mods[i];
        }
    }
}

struct Hashing{
    ll hsh[2][MAX];
    string str;

    Hashing(){}
    Hashing(string _str) {str = _str; memset(hsh, 0,
         sizeof(hsh)); build();}

    void Build(){
        for(ll i = str.size() - 1; i >= 0; i--){
            for(int j = 0; j < 2; j++){
                hsh[j][i] = (hsh[j][i + 1] * bases[j] + str[i])
                      % mods[j];
                hsh[j][i] = (hsh[j][i] + mods[j]) % mods[j];
            }
        }
    }

    pair<ll,ll> GetHash(ll i, ll j){
        assert(i <= j);
        ll tmp1 = (hsh[0][i] - (hsh[0][j + 1] * pwbase[0][j - i
             + 1]) % mods[0]) % mods[0];
        ll tmp2 = (hsh[1][i] - (hsh[1][j + 1] * pwbase[1][j - i
             + 1]) % mods[1]) % mods[1];
        if(tmp1 < 0) tmp1 += mods[0];
        if(tmp2 < 0) tmp2 += mods[1];
        return make_pair(tmp1, tmp2);
    }
};
```

5.2 MO's Algo

```
// Write add(), remove() and get_answer()
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);
}
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] = get_answer();
 }
 for(int i:answers) {cout<<i<<"\n";}
}
```

5.3 Segnemnt Tree another

```
class SegmentTreeLazy
{
    int len;
    vector<ll> tree, lazy;

public:
    SegmentTreeLazy(int n)
    {
        len = n;
        tree.resize(4 * n + 1, 0);
        lazy.resize(4 * n + 1, 0);
    }

    void buildTree(int node, int start, int end, vector<int>
         &arr)
    {
        if (start == end)
        {
            tree[node] = arr[start];
            return;
        }
        int mid = (start + end) / 2;
        buildTree(2 * node, start, mid, arr);
        buildTree(2 * node + 1, mid + 1, end, arr);
        tree[node] = tree[2 * node] + tree[2 * node + 1];
    }

    void rangeUpdate(int node, int start, int end, int l, int
         r, int val)
    {
        if (lazy[node] != 0)
        {
            tree[node] += (end - start + 1) * lazy[node];
            if (start != end)
            {
                lazy[2 * node] += lazy[node];
                lazy[2 * node + 1] += lazy[node];
            }
            lazy[node] = 0;
        }
        if (r < start || end < l)
            return;
        if (l <= start && end <= r)
        {
            tree[node] += (end - start + 1) * val;
            if (start != end)
            {
                lazy[2 * node] += val;
                lazy[2 * node + 1] += val;
            }
            return;
        }
        int mid = (start + end) / 2;
        rangeUpdate(2 * node, start, mid, l, r, val);
        rangeUpdate(2 * node + 1, mid + 1, end, l, r, val);
        tree[node] = tree[2 * node] + tree[2 * node + 1];
    }

    ll rangeQuery(int node, int start, int end, int l, int r)
    {
        if (lazy[node] != 0)
        {
            tree[node] += (end - start + 1) * lazy[node];
            if (start != end)
            {
                lazy[2 * node] += lazy[node];
                lazy[2 * node + 1] += lazy[node];
            }
            lazy[node] = 0;
        }
        if (r < start || end < l)
            return 0;
        if (l <= start && end <= r)
            return tree[node];
        int mid = (start + end) / 2;
        ll p1 = rangeQuery(2 * node, start, mid, l, r);
        ll p2 = rangeQuery(2 * node + 1, mid + 1, end, l, r);
        return p1 + p2;
    }
};
```

5.4 Sparse Table

```
class SparseTable
{
public:
    vector<vector<ll>> st;
    vector<ll> logTable;
    int n;
    int maxLog;

    SparseTable(int nn)
    {
        n = nn;
        maxLog = log2(n) + 1;
        st.assign(n + 1, vector<ll>(maxLog + 1, 0));
        logTable.assign(n + 1, 0);
        for (int i = 2; i <= n; i++)
        {
            logTable[i] = logTable[i / 2] + 1;
        }
    }
    void buildTable(vector<ll> &arr)
    {
        for (int i = 1; i <= n; i++)
        {
            st[i][0] = arr[i];
        }
        for (int j = 1; j <= maxLog; j++)
        {
            for (int i = 1; i + (1 << j) - 1 <= n; i++)
            {
                st[i][j] = st[i][j - 1] + st[i + (1 << (j -
                     1))][j - 1];
            }
        }
    }
    // ll queryMin(int l, int r)
    // {
    //     int len = r - l + 1;
    //     int k = logTable[len];
    //     return min(st[l][k], st[r - (1 << k) + 1][k]);
    // }
    ll query(int l, int r)
    {
        ll ans = 0;
        for (int j = maxLog; j >= 0; j--)
        {
            if ((1 << j) <= r - l + 1)
            {
                ans += st[l][j];
                l += (1 << j);
            }
        }
        return ans;
    }
};
```

5.5 orderedset

```
// 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;
}
```

5.6 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;
    }
};

class segtree
{
    int range;
    vector<node> tree;
public:
    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);
    }
};
```

5.7 segment$_{t}ree$

```
class SegmentTree
{
    int len;
    int suru;
    vector<int> tree;

public:
    SegmentTree(int n, vector<int> &arr, int start)
    {
        len = n;
        suru = start;
        tree.assign(4 * n + 5, 0);
        build(1, suru, len - 1 + suru, arr);
    }

    void build(int node, int start, int end, vector<int> &arr)
    {
        if (start == end)
        {
            tree[node] = arr[start];
            return;
        }
        int mid = (start + end) / 2;
        build(2 * node, start, mid, arr);
        build(2 * node + 1, mid + 1, end, arr);
        tree[node] = tree[2 * node] + tree[2 * node + 1];
    }
    void updateTree(int node, int start, int end, int idx, int
         val)
    {
        if (start > idx || end < idx)
            return;
        if (start == end)
        {
            tree[node] = val;
            return;
        }
        int mid = (start + end) / 2;
        updateTree(2 * node, start, mid, idx, val);
        updateTree(2 * node + 1, mid + 1, end, idx, val);
        tree[node] = tree[2 * node] + tree[2 * node + 1];
    }
    int queryTree(int node, int start, int end, int l, int r)
    {
        if (start > r || end < l)
            return 0;
        if (start >= l && end <= r)
            return tree[node];
        int mid = (start + end) / 2;
        return queryTree(2 * node, start, mid, l, r) +
             queryTree(2 * node + 1, mid + 1, end, l, r);
    }
    void update_function(int idx, int val)
    {
        updateTree(1, suru, len - 1 + suru, idx, val);
    }
    int query_function(int l, int r)
    {
        return queryTree(1, suru, len - 1 + suru, l, r);
    }
};
```

6 E.DP

6.1 Coin Change V1

```
possible[0] = 1;
for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= k; j++) {
        if(i >= coin[j]) {
            possible[i] |= possible[i - coin[j]];
        }
    }
}
```

6.2 Coin Change V2

```
way[0] = 1;
for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= k; j++) {
        if(i >= coin[j]) {
            way[i] += way[i - coin[j]];
        }
    }
}
```

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();
}
```

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

```
vector<vector<int>> graph;
vector<int> depth;
vector<vector<int>> up;

void dfs(int u, int p)
{
    up[u][0] = p;
    depth[u] = depth[p] + 1;
    for (int i = 1; i < 20; i++)
    {
        up[u][i] = up[up[u][i - 1]][i - 1];
    }
    for (auto v : graph[u])
    {
        if (v == p)
            continue;
        dfs(v, u);
    }
}

int lca(int u, int v)
{
    if (depth[u] > depth[v])
        swap(u, v);
    int diff = depth[v] - depth[u];
    for (int i = 0; i < 20; i++)
    {
        if ((diff >> i) & 1)
        {
            v = up[v][i];
        }
    }
    if (u == v)
        return u;
    for (int i = 19; i >= 0; i--)
    {
        if (up[u][i] != up[v][i])
        {
            u = up[u][i];
            v = up[v][i];
        }
    }
    return up[u][0];
}

// kth root from u
int kth(int u, int k)
{
    if (k < 0)
        return -1;
    for (int i = 0; i <= 20; i++)
        if (k & (1 << i))
            u = up[u][i];
    return u;
}

// kth node from u to v, 0th node is u
int kthNode(int u, int v, int k)
{
    int l = lca(u, v);
    int distance = depth[u] + depth[v] - (depth[l] << 1);
    if (k > distance)
        return -1;
    if (depth[l] + k <= depth[u])
        return kth(u, k);
    k -= depth[u] - depth[l];
    return kth(v, depth[v] - depth[l] - k);
}
```

7.5 DSU

```
int findParent(int x)
{
    if (parent[x] == x)
        return x;
    return parent[x] = findParent(parent[x]);
}

bool isConnected(int u, int v)
{
    return findParent(u) == findParent(v);
}

void unionSet(int u, int v)
{
    int pu = findParent(u);
    int pv = findParent(v);
    if (pu == pv)
        return;
    if (len[pu] < len[pv])
        swap(pu, pv);
    parent[pv] = pu;
    len[pu] += len[pv];
}
```

7.6 Dijkstra on Segment Tree

```
const int N = 1e5 + 9;
vector<pair<int, int>> g[N * 9];
inline void add_edge(int u, int v, int w) {
  g[u].push_back({v, w});
}
int add;
void build(int n, int b, int e) {
  if (b == e) {
    add_edge(b, n + add, 0);
    add_edge(n + add * 5, b, 0);
    return;
  }
  int mid = b + e >> 1;
  add_edge(2 * n + add, n + add, 0);
  add_edge(2 * n + 1 + add, n + add, 0);
  add_edge(n + 5 * add, 2 * n + 5 * add, 0);
  add_edge(n + 5 * add, 2 * n + 1 + 5 * add, 0);
  build(2 * n, b, mid);
  build(2 * n + 1, mid + 1, e);
}
void upd(int n, int b, int e, int i,
int j, int dir, int u, int w) {
  if (j < b || e < i) return;
  if (i <= b && e <= j) {
    if (dir) add_edge(u, n + 5 * add, w); // from u to this
        range
    else add_edge(n + add, u, w); // from this range to u
    return;
  }
  int mid = (b + e) >> 1;
  upd(2 * n, b, mid, i, j, dir, u, w);
  upd(2 * n + 1, mid + 1, e, i, j, dir, u, w);
}
vector<long long> dijkstra(int s) {
  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(9 * N + 1, inf);
  vector<bool> vis(9 * N + 1, 0);
  q.push({0, s});
  d[s] = 0;
  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});
      }
    }
  }
  return d;
}
long long ans[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, q, s; cin >> n >> q >> s;
  add = n;
  build(1, 1, n);
  while (q--) {
    int ty; cin >> ty;
    int u, l, r, w;
    if (ty == 1) {
      cin >> u >> l >> w;
      r = l;
    }
    else {
      cin >> u >> l >> r >> w;
    }
    upd(1, 1, n, l, r, ty <= 2, u, w);
  }
  auto ans = dijkstra(s);
  for (int i = 1; i <= n; i++) {
    if (ans[i] == 1e18) ans[i] = -1;
    cout << ans[i] << ' ';
  }
  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 Kosaraju's Algorithm for Strongly Connected Components

```
const ll SZ = 1e6 + 5;
ll vis[SZ];
vector<int> adj[SZ];
stack<ll> st;
vector<ll> transpose[SZ];
vector<pair<ll, ll>> vp;
void topo(int node) {
    vis[node] = 1;
    for (auto child : adj[node]) {
        if (vis[child] == 0) topo(child);
    }
    st.push(node);
}
void revDfs(ll node) {
    cout << node << " ";
    vis[node] = 1;
    for (auto it : transpose[node]) {
        if (!vis[it]) revDfs(it);
    }
}
void trans(ll m)
{
    memset(vis, 0, sizeof(vis));
    for (int i = 0; i < m; i++) {
        transpose[vp[i].second].push_back(vp[i].first);
    }

}
```

7.9 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;
}
```

7.10 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.

7.11 Number of Shortest Path dijkstra

```
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;
  }
```

7.12 hamaltonian Path

```
/*

Using Visited Array and Recursion and Backtracking TC: O(n!)
So iss Visited Array jo state hain usko Bitmask kardo 'n' bits
     ka mask bcoz 'n' Nodes hain
1-->2-->3 and 1-->3-->2 so here mask is same 111 But Ending
     cities are different firstly 3 and latter 2,
so we use dp[mask][ending city]
*/
ll n,m;
vector<int> graph[25];
ll dp[1<<20][20]; // changes done here
// time complexity: O(2^n*(n+m))
// for complexte graph m = n^2;
// time complexity: O(2^n*n^2)
ll dfs(int node, int mask){
    // reached at node n after visiting all nodes once
    if(__builtin_popcount(mask)==n) return 1;
    // reached at node n not visiting all nodes
    else if(node==n-1) return 0;
    if(dp[mask][node]!=-1) return dp[mask][node];
    ll ans = 0;
    for(auto it: graph[node]){
        if(mask&(1<<it)) continue;
        ans=(ans+dfs(it,mask|(1<<it)))%MOD;
    }

    return dp[mask][node]=ans%MOD;
}

// we have to do in 0-based indexing
```

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 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_inserter(tree[node]));
}
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(all(tree[2*node]),l);
    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)]);
        }
    }
}
```

8.3 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;
}
```

8.4 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;

}
```

8.5 Persistance Segment Tree

```
struct Node {
    Node *left;
    Node *right;
    int cnt;

    Node() {
        left = NULL;
        right = NULL;
        cnt = 0;
    }
};
Node *root = new Node();
Node *x[lx];
int sum(Node *temp)
{
    if(temp) return temp->cnt;
    return 0;
}
Node *Insert(Node *node, int b, int e, int idx)
{
    Node *temp = new Node();
    if(node) *temp = *node;

    if(b == idx and e == idx) {
        temp->cnt += 1;
        return temp;
    }
    int mid_idx = (b + e) / 2;
    if(idx <= mid_idx) {
        temp->left = Insert(temp->left, b, mid_idx, idx);
    }
    else {
        temp->right = Insert(temp->right, mid_idx + 1, e, idx);
    }
    temp->cnt = sum(temp->left) + sum(temp->right);
    return temp;
}
int query(Node *node, int b, int e, int k)
{
    if(b == e) return b;
    int mid = (b + e) / 2;
    int koyta = sum(node->left);
    if(koyta >= k) {
        return query(node->left, b, mid, k);
    }
    else {
        return query(node->right, mid + 1, e, k - koyta);
    }
}

void solve()
{
    x[100001] = root;
    for(int i = 100000; i >= 1; i--) {
        x[i] = x[i + 1];
        for(int j = 0; j < (int)indices[i].size(); j++) {
            x[i] = Insert(x[i], 1, 100000, indices[i][j]);
        }
    }
    while(q--) {
        int l, k;
        cin >> l >> k;
        cout << a[query(x[l], 1, 100000, k)] << endl;
    }
}
```

9 H.Strings

9.1 2D hashing

```
#define MAX 2005
#define ll long long int

int Base[2] = {773,709};
int Mod[2] = {281559881,398805713};
int InvBase[2][2];

int Pow[MAX][2][2],Inv[MAX][2][2];
int PowCell[MAX][MAX][2],InvCell[MAX][MAX][2];

int A[MAX][MAX],B[MAX][MAX];
int cA[MAX][MAX][2],cB[MAX][MAX][2];

inline int cMod(int x,int mod){
    while(x >= mod) x-=mod;
    while(x < 0)   x+=mod;
    return x;
}

int bigMod(int n,int r,int mod){
    if(r==0) return 1;
    ll ret = bigMod(n, r/2, mod);
    ret = (ret*ret) % mod;
    if(r&1) ret = (ret*n)% mod;
    return ret;
}
int invMod(int n,int mod) {return bigMod(n,mod-2,mod);}

void Preprocess(){
    for(int b=0;b<2;b++) for(int m=0;m<2;m++)
        Pow[0][b][m]    = Inv[0][b][m]  = 1;
    for(int b=0;b<2;b++) for(int m=0;m<2;m++) for(int
          i=1;i<MAX;i++)
        Pow[i][b][m]    = (Pow[i-1][b][m]*1LL*Base[b]) % Mod[m];
    for(int m=0;m<2;m++) for(int i=0;i<MAX;i++) for(int
          j=0;j<MAX;j++)
        PowCell[i][j][m]  = (Pow[i][0][m]*1LL*Pow[j][1][m]) %
              Mod[m];

    for(int b=0;b<2;b++) for(int m=0;m<2;m++)
        InvBase[b][m]   = invMod(Base[b],Mod[m]);
    for(int b=0;b<2;b++) for(int m=0;m<2;m++) for(int
          i=1;i<MAX;i++)
        Inv[i][b][m]    = (Inv[i-1][b][m]*1LL*InvBase[b][m]) %
              Mod[m];
    for(int m=0;m<2;m++) for(int i=0;i<MAX;i++) for(int
          j=0;j<MAX;j++)
        InvCell[i][j][m] =
              (Inv[i][0][m]*1LL*Inv[j][1][m])%Mod[m];
}

void Build(int M[MAX][MAX],int cM[MAX][MAX][2],int r,int c){
    for(int m=0;m<2;m++){
        for(int i=1;i<=r;i++){
            for(int j=1;j<=c;j++){
                int Now=(M[i][j]*1LL*PowCell[i][j][m])%Mod[m];
                cM[i][j][m]=cM[i-1][j][m]+cM[i][j-1][m]-cM[i-1][j-1][m]+No
                cM[i][j][m] = cMod(cM[i][j][m], Mod[m]);
            }
        }
    }
}

pair<int,int> Query(int cM[MAX][MAX][2],int x1,int y1,int
      x2,int y2){
    ll Sum[2];
    for(int m=0;m<2;m++){
        Sum[m] =
              cB[x2][y2][m]-cB[x2][y1-1][m]-cB[x1-1][y2][m]+cB[x1-1][y1-1
        Sum[m] = cMod(Sum[m],Mod[m]);
        Sum[m] = (InvCell[x1-1][y1-1][m]*1LL*Sum[m]) % Mod[m];
    }
    return {Sum[0],Sum[1]};
}
```

9.2 Hasing

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

const int N = 2e6 + 9;
const int p1 = 137, mod1 = 1e9 + 7, p2 = 277, mod2 = 1e9 + 9;


int power(long long n, long long k, 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;
}

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;
  }
}

pair<int, int> string_hash(string s) {
  int n = s.size();
  pair<int, int> hs({0, 0});
  for (int i = 0; i < n; i++) {
    hs.first += 1LL * s[i] * pw[i].first % mod1;
    hs.first %= mod1;
    hs.second += 1LL * s[i] * pw[i].second % mod2;
    hs.second %= mod2;
  }
  return hs;
}
pair<int, int> pref[N];
void build(string s) {
  int n = s.size();
  for (int i = 0; i < n; i++) {
    pref[i].first = 1LL * s[i] * pw[i].first % mod1;
    if (i) pref[i].first = (pref[i].first + pref[i - 1].first)
         % mod1;
    pref[i].second = 1LL * s[i] * pw[i].second % mod2;
    if (i) pref[i].second = (pref[i].second + pref[i -
         1].second) % mod2;
  }
}
pair<int, int> get_hash(int i, int j) {
  assert(i <= j);
  pair<int, int> hs({0, 0});
  hs.first = pref[j].first;
  if (i) hs.first = (hs.first - pref[i - 1].first + mod1) %
       mod1;
  hs.first = 1LL * hs.first * ipw[i].first % mod1;
  hs.second = pref[j].second;
  if (i) hs.second = (hs.second - pref[i - 1].second + mod2) %
       mod2;
  hs.second = 1LL * hs.second * ipw[i].second % mod2;
  return hs;
}
int lcp(int i, int j, int x, int y) { // O(log n)
  int l = 1, r = min(j - i + 1, y - x + 1), ans = 0;
  while (l <= r) {
    int mid = l + r >> 1;
    if (get_hash(i, i + mid - 1) == get_hash(x, x + mid - 1)) {
      ans = mid;
      l = mid + 1;
    }
    else {
      r = mid - 1;
    }
  }
  return ans;
}
```

9.3 KMP Algorithm

```
int lps[lx];
void calLPS(string pattern)
{
    int len = pattern.size();
    lps[0] = 0;
    for(int i = 1, k = 0; i < len; i++) {
        while(k > 0 and pattern[i] != pattern[k]) {
            k = lps[k - 1];
        }
        if(pattern[i] == pattern[k]) k++;
        lps[i] = k;
    }
}
int KMP(string text, string pattern)
{
    calLPS(pattern);
    int matches = 0;

    for(int i = 0, k = 0; i < (int)text.size(); i++) {
        while(k > 0 and pattern[k] != text[i]) {
            k = lps[k - 1];
        }
        if(pattern[k] == text[i]) k++;
        if(k == pattern.size()) {
            matches++;
            k = lps[k - 1];
        }
    }
    return matches;
}
```

9.4 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]);
    }
}
```

9.5 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);
}
```

9.6 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]);
    }
}
```

9.7 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];
        }
    }
}
```

9.8 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;
}
```

9.9 Z function

```
//0-indexed string s[0...n-1]
//fail[i] = maximum x such that s[0....x-1] matches with
     s[i-x+1...i]
int fail[MAX];
void build_failure(string s){
    int n=s.size();
    for(int i=1;i<n;i++){
        int j=fail[i-1];
        while(j>0 && s[i]!=s[j]) j=fail[j-1];
        if(s[i]==s[j]) j++;
        fail[i]=j;
    }
}

int KMPSearch(string txt, string pat){
    int Count = 0;
    int i = 0;
    int j = 0;
    while(i < txt.size()){
        if(txt[i] == pat[j]) i++, j++;

        if(j == pat.size()) Count++, j = fail[j - 1];
        else if(i < txt.size() && txt[i] != pat[j]){
            if(j != 0) j = fail[j - 1];
            else i++;
        }
    }
    return Count;
}

//0-indexed string s[0...n-1]
//z[i] = maximum x such that s[0....x-1] matches with
     s[i..i+x-1]
int z[MAX];
void zAlgo(string s){
    int L=0,R=0;
    int n=s.size();
    for(int i=1;i<n;i++){
        if(i>R) {L=R=i; while(R<n && s[R-L]==s[R]) R++;
             z[i]=R-L; R--;}
        else{
            int k=i-L;
            if(z[k]<R-i+1) z[i]=z[k];
            else {L=i; while(R<n && s[R-L]==s[R]) R++; z[i]=R-L;
                 R--;}
        }
    }
}
```

10 I.Bit

10.1 Equation

* a+b $= \text{a}\oplus b + 2(a\&b)$
$* \, a + b = a|b + a\&b$
$* \, a \oplus b = a|b - a\&b$
$* \, k_{th}$ bit is set in x iff x mod $2^{k-1} > 2^k$
$* \, k_{th}$ bit is set in x iff x mod $2^{k-1} - x$ mod $2^k \neq 0 (= 2^k)$
$* \, n$ 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 B,Custom Hash Unorder Map

```
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
            chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};
unordered_map<long long, int, custom_hash> safe_map;
```

11.2 D,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;
}
```

11.3 F,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);
    }
}
```

11.4 H,Dinik Max Flow

```
#define ll          long long
#define pb          push_back
#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++)
#define MEM(a,x)    memset(a,x,sizeof(a))
struct edge
{
    int to,rev,f,cap;
};
const int maxnodes=10005;
int s,t,lev[maxnodes],q[maxnodes],work[maxnodes];
vector<edge>g[maxnodes];
inline void addEdge(int u,int v,int w)
{
    edge a= {v,g[v].size(),0,w};
    edge b= {u,g[u].size(),0,0};
    g[u].pb(a);
    g[v].pb(b);
}
bool dinic_bfs()
{
    MEM(lev,-1);
    lev[s]=0;
    int idx=0;
    q[idx++]=s;
    REP(i,idx)
    {
        int u=q[i];
        REP(j,g[u].size())
        {
            edge &e=g[u][j];
            if(lev[e.to]<0 and e.f<e.cap)
            {
                lev[e.to]=lev[u]+1;
                q[idx++]=e.to;
            }
        }
    }
    return lev[t]>=0;
}
int dinic_dfs(int u,int f)
{
    if(u==t) return f;
    for(int &i=work[u];i<g[u].size();i++)
    {
        edge &e=g[u][i];
        if(e.cap<=e.f) continue;
        if(lev[e.to]==lev[u]+1)
        {
            int flow=dinic_dfs(e.to,min(f,e.cap-e.f));
            if(flow>0)
            {
                e.f+=flow;
                g[e.to][e.rev].f-=flow;
                return flow;
            }
        }
    }
    return 0;
}
int maxFlow()
{
    int ret=0;
    while(dinic_bfs())
    {
        MEM(work,0);
        while(int flow=dinic_dfs(s,INT_MAX))
            ret+=flow;
    }
    return ret;
}
int main()
{
    int n,m;
    cin >> n >> m >> s >> t;
    REP(i,m)
    {
        int u,v,w;
        cin >> u >> v >> w;
        addEdge(u,v,w);
        addEdge(v,u,w); //If bidirectional
    }
    cout << maxFlow();
    return 0;
}
```

11.5 I, 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]
}
```

11.6 J,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;
}
```

11.7 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
```

11.8 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());
}
```

11.9 common equations

- **Modular Arithmetic:** [(a + b) $\bmod m$ = [($a \bmod m$) + ($b \bmod m$)] $\bmod m$][($a \times b$) $\bmod m$ = [($a \bmod m$) $\times$ ($b \bmod m$)] $\bmod m$]

- **Exponentiation by Squaring:** [$a^b \bmod m$ = $\begin{cases} 1 & \text{if } b = 0 \\ (a^{b/2} \bmod m)^2 & \text{if } b \text{ is even]} \\ a \times (a^{b-1} \bmod m) & \text{if } b \text{ is odd} \end{cases}$

- **Fibonacci Numbers:** [F(n) = $\begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$

  ]

- **Combinatorial Counting:** [C(n, k) = n!$_{\overline{k!(n-k)!}}$]

- **Inclusion-Exclusion Principle:** [—A $\cup B| = |A| + |B| - |A \cap B|$]

- **Binary Search:** To find x such that $f(x)$ = target: [while l ¡ r:
  m = $\left\lfloor \frac{l+r}{2} \right\rfloor$
  if $f(m)$ < target then $l = m + 1$
  else $r = m$]

- **Dijkstra's Algorithm:** To find the shortest path: [Initialize: dist[s] = 0, for all other v, dist[v] = ∞][while priority queue is not empty:][u = extract min from priority queue][for each neighbor v of u :][if $dist[u] + w(u, v) < dist[v]$: $dist[v] = dist[u] + w(u, v)$]

- **Greatest Common Divisor (GCD):** [$\gcd(a, b) = \gcd(b, a \bmod b)$]

- **Lowest Common Multiple (LCM):** [lcm(a, b) = —a $\cdot b|_{\overline{\gcd(a,b)}}$]

- **Sieve of Eratosthenes:** To find all primes up to n:

  - Initialize a list of boolean values, ‘is$_{p}rime‘, of size n+1$ to ‘true‘.
  - For $p = 2$ to $\sqrt{n}$:
    * If ‘is$_{p}rime[p]‘ is true, mark all multiples of p$ as false.

- **Geometric Series:** The sum of the first n terms:

$$S_n = a\frac{1 - r^n}{1 - r} \quad (r \neq 1)$$

- **Harmonic Series:** The sum of the first n terms:

$$H_n = \sum_{k=1}^{n} \frac{1}{k} \approx \ln(n) + \gamma$$

  Where γ is the Euler-Mascheroni constant.

- **Sum of the First n Natural Numbers:**

$$S_n = \frac{n(n+1)}{2}$$

- **Sum of the First n Squares:**

$$S_n = \frac{n(n+1)(2n+1)}{6}$$

- **Sum of the First n Cubes:**

$$S_n = \left(\frac{n(n+1)}{2}\right)^2$$

11.10 expectation

- Expected Value Definition:

$$E(X) = \sum (x_i \times P(x_i))$$

  Where x_i are the possible outcomes and $P(x_i)$ is the probability of x_i.

- Linearity of Expectation:

$$E(X + Y) = E(X) + E(Y)$$

  This holds even if X and Y are not independent.

- Expected Value of a Uniform Distribution: For a uniform distribution from a to b:

$$E(X) = \frac{a + b}{2}$$

- Geometric Distribution: If X is the number of trials until the first success with probability p:

$$E(X) = \frac{1}{p}$$

- Binomial Distribution: If X is the number of successes in n independent trials with probability p:

$$E(X) = n \times p$$

- Expected Value of a Dice Roll: For a fair n-sided die:

$$E(X) = \frac{n + 1}{2}$$

- Conditional Expectation: If you know some condition A:

$$E(X \mid A) = \sum (x_i \times P(x_i \mid A))$$

- Expected Value of a Random Variable with Indicator Function: If I is an indicator variable:

$$E(I) = P(\text{event occurs})$$

- Expected Value in Card Games: Consider the probabilities of drawing each card and the outcomes associated with them.

- Expected Value in Random Walks: Use symmetry and boundary conditions to simplify calculations.

11.11 geo

```
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(detr == 0) 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;
}

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(det == 1) return false;
   x = (p*d - b*q) / det;
   y = (a*q - c*p) / det;
   return true;
}
```