Team notebook

International University of Business Agriculture and Technology

November 3, 2023

Contents

1 1.Settings and Script

1.1 Geany Setup

```
// 1. Open any C++ file in Geany, go to Build -> Set
     Build Commands and copy my flags.
//    Compile (F8): g++ -std=c++17 -Wshadow -Wall -o
     "%e" "%f" -O2 -Wno-unused-result
```

```
//   Build (F9): g++ -std=c++17 -Wshadow -Wall -o "%e"
     "%f" -g -fsanitize=address -fsanitize=undefined
     -D_GLIBCXX_DEBUG
// 2. gedit ~/.bashrc > #force_color_prompt=yes > add
     ulimit -s 2000123
// 3. "Tools" > "Configuration Files" >
     "filetypes.common>[styling]\n line_height=0;2;
// 4. Font Monospace Regular 10 > sans 9 > Monospace 9
```

1.2 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;
}
vector<pair<int, int>> gen_tree(int n = 0)
{
    assert(n >= 0);
    vector<pii> res(n ? n - 1 : 0);
    // if you like to have bamboo like tree or star
         like tree uncomment below 8 lines
    /*if (rng() % 5 == 0) { // bamboo like tree
      for (int i = 1; i < n; ++i) res[i-1] = {i, i + 1};
      return res;
    }
    if (rng() % 7 == 0) { // star tree
      for (int i = 2; i <= n; ++i) res[i-2] = {1, i};
      return res;
    }*/
    iota(permutation, permutation + 1 + n, 0);
    shuffle(permutation + 1, permutation + 1 + n, rng);
    for (int i = 2; i <= n; ++i)
    {
        int u = i, v = rand(1, i - 1);
        u = permutation[u], v = permutation[v];
        res[i - 2] = minmax(u, v); // u < v, just for
             convenience while debugging
    }
    shuffle(res.begin(), res.end(), rng);
    return res;
}
vector<pair<int, int>> simple_graph(int n = 0, int m =
     0)
{
    assert(n > 0 && m >= n);
    int max_edges = n * (n - 1) / 2;
    assert(m <= max_edges);
    vector<pii> res = gen_tree(n);
    set<pii> edge(res.begin(), res.end());
    for (int i = n; i <= m; ++i)
    {
        while (true)
        {
            int u = rand(1, n), v = rand(1, n);
            if (u == v)
                continue;
            auto it = edge.insert(minmax(u, v));
            if (it.second)
                break;
        }
    }
    res.assign(edge.begin(), edge.end());
    return res;
}
} // namespace generator
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;
}
ostream &operator<<(ostream &other, const
     vector<pair<int, int>> &v)
{
    for (const auto &x : v)
        other << x.first << ' ' << x.second << '\n';
    return other;
}
// comment the just below line if test cases required
#define SINGLE_TEST
const int max_tests = 1;
// complete this function according to the requirements
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();
    }
}
```

1.3 Main Script

```
#!/bin/bash

# to color the output text in different colours
green=$(tput setaf 71);
red=$(tput setaf 1);
blue=$(tput setaf 32);
orange=$(tput setaf 178);
bold=$(tput bold);
reset=$(tput sgr0);

echo ${bold}${blue}How many testcase you want to test?
     :${reset}
read max_tests
echo ${bold}${blue}Enter file Name? :${reset}
```

```
read file_name

# You can chamge the version of C++ or add the compiler
     flags you wish
g++ gen.cpp -o generator || { echo
     ${bold}${orange}Compilation Error in ${reset}
     gen.cpp; exit 1; }
g++ solution.cpp -o original || { echo
     ${bold}${orange}Compilation Error${reset} in
     $1.cpp; exit 1; }
g++ brute.cpp -o brute || { echo
     ${bold}${orange}Compilation Error${reset} in
     $2.cpp; exit 1; }



diff_found=0
i=1

while [[ $i -le $max_tests ]]
do
    ./generator > input1.txt
    ./original < input1.txt > original_output.txt #||
         {echo failed; exit 1;}
    ./brute < input1.txt > brute_output.txt
    if diff --tabsize=1 -F --label --side-by-side
         --ignore-space-change original_output.txt
         brute_output.txt > dont_show_on_terminal.txt;
         then
        echo "${orange}test_case #$i:
             ${bold}${green}passed${reset}"
    else
        echo "${orange}test_case #$i:
             ${bold}${red}failed${reset}"
        diff_found=1
        break
    fi
    i=$((i+1))
done

if [ $diff_found -eq 1 ]
then
    echo "${blue}Input: ${reset}"
    cat input1.txt
    echo ""

    echo "${blue}Output: ${reset}"
    cat original_output.txt
    echo ""

    echo "${blue}Expected: ${reset}"
    cat brute_output.txt
    echo ""
    cp input1.txt input$file_name.txt
    cp brute_output.txt output$file_name.txt
    notify-send "Wrong Answer"
else
    notify-send "Accepted"
fi

rm input1.txt
rm generator
rm original
rm brute
rm original_output.txt
rm brute_output.txt
rm dont_show_on_terminal.txt
```

1.4 Template

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
template<typename T>
using ordered_set=tree<T,null_type,
less<T>,rb_tree_tag,
tree_order_statistics_node_update>;
typedef tree<int,null_type,less<int>,
rb_tree_tag,
tree_order_statistics_node_update>indexed_set;
priority_queue <int, vector<int>, greater<int>> gq;
#define faster ios::sync_with_stdio(0);
     cin.tie(0);cout.tie(0);
#define pi                          2*acos(0.0)
void init_code() {
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
}
```

2 2.Combinatorics

2.1 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;
}
```

2.2 How Many Digit X^Y

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

2.3 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$[log_{10}(1)+log_{10}(2)+\ldots\ldots\ldots\ldots\ldots+log_{10}(N)]+1$

2.4 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; //Solution Exists
    return -1;
       //No Solution
}
```

2.5 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;
}
```

2.6 Longest Increasing 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();
}
```

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 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;
}
```

2.9 Modular Inverse

```
const ll mx = 1e6+7;
ll fact[mx];
void factorial()
{
    fact[0]=1;
    for (ll i = 1; i < mx; i++)
        fact[i] = (fact[i-1] * i) % MOD;
}
```

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 Pollard RHO

```
const int N = 1000005;
LL Mul(LL a, LL b, LL m) {
    LL ret=0, c=a;
    while(b) {
        if(b&1) ret=(ret+c)%m;
        b>>=1;
        c=(c+c)%m;
    }
    return ret;
}
LL bigmod(LL a, LL n, LL m) {
    LL ret=1, c=a;
    while(n) {
        if(n&1) ret=Mul(ret, c, m);
        n>>=1;
        c=Mul(c, c, m);
    }
    return ret;
}
bool isPrime(long long n) {
    if (n == 2) return 1;
    if (n%2 == 0) return 0;
    long long d = n-1;
    while(d%2 == 0) d >>= 1;
    int test[] = {2,3,5,7,11,13,17,19,23};
    for(int i = 0; i < 9; i++) {
        long long x = test[i]%(n-2), temp = d;
        if (x < 2) x += 2;
        long long a = bigmod(x, d, n);
        while(temp != n-1 && a != 1 && a != n-1) {
            a = Mul(a, a, n);
            temp <<= 1;
        }
        if (a != n-1 && (temp&1) == 0) return 0;
    }
    return 1;
}
long long pollard_rho(long long n, long long c) {
    long long x = 2, y = 2, i = 1, k = 2, d;
    while(true) {
        x = (Mul(x, x, n) + c);
        if (x >= n) x -= n;
        d = __gcd(abs(x-y),n);
        if (d > 1) return d;
        if (++i == k) {
            y = x, k <<= 1;
        }
    }
    return n;
}
```

```
bool stat[N];
vector<LL>primes;
void siv() {
    for(int i = 4; i < N; i += 2)stat[i] = 1;
    int sq = sqrt(N);
    for(int i = 3; i <= sq; i += 2) {
        if(stat[i])continue;
        for(int j = i * i; j < N; j += 2 * i)stat[j] =
            1;
    }
    for(int i = 2; i < N; i++)if(stat[i] ==
        0)primes.push_back(i);
}
void llfactorize(long long n, vector<long long> &f) {
    if (n == 1) return;
    if (n < 1e9) {
        for(int i = 0; primes[i]* primes[i] <= n; i++) {
            while(n%primes[i] == 0) {
                f.push_back(primes[i]);
                n /= primes[i];
            }
        }
        if (n != 1) f.push_back(n);
        return;
    }
    if (isPrime(n)) {
        f.push_back(n);
        return;
    }
    long long d = n;
    for(int i = 2; d == n; i++) {
        d = pollard_rho(n, i);
    }
    llfactorize(d, f);
    llfactorize(n/d, f);
}
void factorize(long long n, vector<pair<long long,long
    long>> &ans) {
    vector<long long> v;
    llfactorize(n, v);
    if(v.size() == 0)return;
    sort(v.begin(), v.end());
    long long a = v[0], b = 1;
    for(int i = 1; i < v.size(); i++) {
        if (v[i] == v[i-1] ) b++;
        else {
            ans.push_back({a,b});
            a = v[i];
            b = 1;
        }
    }
    ans.push_back({a,b});
}
```

```
LL phi(LL n,vector<pair<long long,long long>> &ans) {
    LL ph=n;
    for(auto p:ans) {
        ph/=p.ff;
        ph*=(p.ff-1);
    }
    return ph;
}
int main() {
    siv();
    int t;
    cin >> t;
    while(t--) {
        vector<pair<long long,long long>>v;
        LL a;
        cin >> a;
        if(a!=1 and isPrime(a)) {
            a++;
            while(!isPrime(a)) a++;
            cout << a << '\n';
            continue;
        }
        factorize(a,v);
        LL x=phi(a,v);
        LL b=a+1;
        while(true) {
            vector<pair<long long,long long>>vv;
            factorize(b,vv);
            LL y=phi(b,vv);
            if(y>x) {
                cout << b << '\n';
                break;
            }
            b++;
        }
    }
}
```

2.12 Segment Tree Lazy Propagation

```
int query(int node, int b, int e, int i, int j, int
    carry = 0) {
    if (i > e || j < b) return 0;
    if (b >= i and e <= j)
        return tree[node].sum + carry * (e - b + 1);
    int Left = node << 1;
    int Right = (node << 1) + 1;
    int mid = (b + e) >> 1;
    int p1 = query(Left, b, mid, i, j, carry +
        tree[node].prop);
```

```
    int p2 = query(Right, mid + 1, e, i, j, carry +
        tree[node].prop);
    return p1 + p2;
}
```

2.13 nCr with Big Mod

```
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;
}
```

2.14 nCr

```
#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))
const LL MOD=1e9+7;
const int N=200005;
LL fact[N],inv[N];
LL BigMod(LL B,LL P,LL M) {
    LL R=1;
    while(P>0) {
        if(P & 1) R=(R*B)%M;
        P=P>>1;
        B=(B*B)%M;
    }
    return R%M;
}
LL nCr(int n,int r) {
    LL up=fact[n];
    LL down=(inv[r]*inv[n-r])%MOD;
    return (up*down)%MOD;
}
void pre() {
    fact[0]=1;
    FOR(i,1,N-1) fact[i]=(fact[i-1]*(LL)i)%MOD;
    inv[N-1]=BigMod(fact[N-1],MOD-2,MOD);
    ROF(i,N-2,1) inv[i]=(inv[i+1]*(i+1))%MOD;
    inv[0]=1;
}
```

3 3.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 Triangle

Let a, b, c be length of the three sides of a triangle.

$$p = (a + b + c) * 0.5$$

The inradius is defined by:

$$iR = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$$

The radius of its circumcircle is given by the formula:

$$cR = \frac{abc}{\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}}$$

4 4.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);
    }
}
```

4.2 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.
```

4.3 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;
}
```

4.4 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;
}
```

4.5 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*/
```

4.6 Total Digit of N factorial

4.6.1 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;
}
```

4.6.2 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;
}
```

4.6.3 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, $log_b n = log_{10} n \times log_b 10$ $log_b n = \frac{log_{10} n}{log_{10} b}$

4.7 $n^2 logn PerQuery Matrix Expo$

```
const int MOD = 1e9+7;
typedef vector<int> row;
typedef vector<row> matrix;
matrix operator*(const matrix&a, const matrix &b) {
    int n = a.size();
    int p = b.size();
    int m = b[0].size();

    matrix ans(n, row(m));
    for (int i=0; i<n; i++)
        for (int j=0; j<m; j++)
            for (int k=0; k<p; k++)
                ans[i][j] = (ans[i][j] +
                     1LL*a[i][k]*b[k][j])%MOD;
    return ans;
}
matrix unit(int n) {
    matrix ans(n, row(n));
    for (int i=0; i<n; i++)
        ans[i][i] = 1;
    return ans;
}
matrix power(const matrix &a, long long p) {
    if (p == 0)    return unit(a.size());
    matrix ans = power(a, p/2);
    ans = ans * ans;
    if (p%2)       ans = ans*a;
    return ans;
}
const int K = 31;
matrix pwr[K];
int main() {
    int n, m, q;
    cin>>n>>m>>q;
    matrix base(n, row(n));
    for (int i=0; i<m; i++) {
        int u, v;
        cin>>u>>v;
        u--;
        v--;
        base[u][v]++;
    }
    pwr[0] = base;
    ///Precalculating all the 2th power of base matrix
    for (int i=1; i<K; i++) pwr[i] = pwr[i-1]*pwr[i-1];
    while (q--) {
        int s, t, k;
        cin>>s>>t>>k;
        s--;
        t--;
        matrix col(n, row(1));
        col[t][0] = 1;

        for(int i=0; i<31; i++)
            if (k&(1<<i))
                col = pwr[i] * col;

        cout<<col[s][0]<<'\n';
    }
}
```

5 5.DP

5.1 Digit DP devide by K

```
int len,id,inp[10],k;
int dp[10][85][85][2];
int vis[10][85][85][2];
int call(int pos,bool isSmall,int val,int rem) {
    if(pos==len) {
        if(val==0 and rem==0) return 1;
        else return 0;
    }
    if(vis[pos][val][rem][isSmall]==id) return
         dp[pos][val][rem][isSmall];
    vis[pos][val][rem][isSmall]=id;
    int last=9;
    if(!isSmall) last=inp[pos];
    int ret=0;
    for(int i=0; i<last; i++)
         ret+=call(pos+1,1,(val*10+i)%k,(rem+i)%k);
    if(!isSmall)
         ret+=call(pos+1,0,(val*10+last)%k,(rem+last)%k);
    else
         ret+=call(pos+1,1,(val*10+last)%k,(rem+last)%k);
    return dp[pos][val][rem][isSmall]=ret;
}
int solve(int x) {
    if(x<0) return 0;
    if(k>83) return 0; //Though only 10 digit, digit
         sum can be maximum 82
    if(k==1) return x;
```

```
    len=0;
    while(x) {
        inp[len++]=x%10;
        x/=10;
    }
    reverse(inp,inp+len);
    id++;
    return call(0,0,0,0);
}
int main() {
    int t;
    scanf("%d",&t);
    FOR(tc,1,t) {
        int lo,hi;
        scanf("%d %d %d",&lo,&hi,&k);
        printf("Case %d:
             %d\n",tc,solve(hi)-solve(lo-1));
    }
}
```

5.2 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 6.Graph and Tree

6.1 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;
}
```

6.2 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;
}
```

6.3 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);
```

6.4 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);
```

6.5 DFS LCA

```
int n, l, timer;
vector<vector<int>> adj;
vector<int> tin, tout;
vector<vector<int>> up;
void dfs(int v, int p) {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; ++i)
        up[v][i] = up[up[v][i-1]][i-1];
    for (int u : adj[v]) {
        if (u != p)
            dfs(u, v);
    }
    tout[v] = ++timer;
}
bool is_ancestor(int u, int v) {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v) {
    if (is_ancestor(u, v)) return u;
    if (is_ancestor(v, u)) return v;
    for (int i = l; i >= 0; --i) {
        if (!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}
void preprocess(int root) {
    tin.resize(n);
    tout.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(root, root);
}
```

6.6 DSU

```
typedef pair< int, int >PII;
const int MAXN = 1e5+7;
int bap[MAXN], sz[MAXN];
int parent(int u) {
    if (u == bap[u]) return u;
    return parent(bap[u]);
}
bool unite(int u, int v) {
    int cu = parent(u);
    int cv = parent(v);
    if (cu == cv) return false;
    if (sz[cu] < sz[cv]) swap(cu, cv);
    bap[cv] = cu;
    sz[cu] += sz[cv];
    return true;
}
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) bap[i] = i, sz[i] = 1;
    vector< PII >redundant;
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        if (!unite(u, v)) redundant.emplace_back(u, v);
    }
    vector< int >st;
    for (int i = 1; i <= n; i++)
        st.push_back(parent(i));
    sort(st.begin(), st.end());
    st.erase(unique(st.begin(), st.end()), st.end());
    cout << redundant.size() << endl;
    for (int i = 0; i < redundant.size(); i++) {
        cout << redundant[i].first << " " <<
            redundant[i].second << " "
            << st[i] << " " << st[i+1] << endl;
    }
    return 0;
}
```

6.7 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;
}
```

6.8 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;
}
```

6.9 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);
    }

}
```

6.10 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;
}
```

6.11 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;

}
```

6.12 Mo on Tree

```
const double inf = 0.0000;
const int lx = 1e6 + 10;
const int mod = 998244353;
const int hs = 3797;
inline bool checkBit (int n, int i) { return
    n&(1LL<<i); }
inline int setBit (int n, int i) { return n|(1LL<<i);; }
inline int resetBit (int n, int i) { return
    n&(~(1LL<<i)); }
vector<int> adj[lx];
bool vis[lx];
int st[lx], en[lx], cnt[lx], depth[lx], a[lx];
int parent[lx][20], res[lx];
string colors;
int ans, _time;
int id[lx * 2];
const int B = 320;
struct query {
        int l, r, id;
        bool operator < (const query& x) {
                if(l / B == x.l /B) return r < x.r;
                return l / B < x.l / B;
        }
}Q[lx];
void dfs(int node, int p)
{
        if(colors[node] == '1') a[node] = node;
        else a[node] = a[p];

        st[node] = ++_time;
        id[_time] = node;

        depth[node] = depth[p] + 1;
        parent[node][0] = p;

        for(int k = 1; k < 20; k++) {
                parent[node][k] = parent[parent[node][k
                    - 1]][k - 1];
        }

        for(int child : adj[node]) {
                if(child != p) dfs(child, node);
        }

        en[node] = ++_time;
        id[_time] = node;
}
int getLCA(int u, int v) {
    if(depth[u] < depth[v]) swap(u, v);
    for(int k = 19; k >= 0; k--) if(depth[parent[u][k]]
        >= depth[v]) u = parent[u][k];
    if(u == v) return u;
    for(int k = 19; k >= 0; k--) if(parent[u][k] !=
        parent[v][k]) u = parent[u][k], v =
        parent[v][k];
    return parent[u][0];
```

```
}
inline void add(int u)
{
	int x = a[u];
	if(cnt[x] == 0) ans++;
	cnt[x]++;
}
inline void rem(int u)
{
	int x = a[u];
	cnt[x]--;
	if(cnt[x] == 0) ans--;
}
inline void check(int u)
{
	if(!vis[u]) add(u);
	else rem(u);
	vis[u] ^= 1;
}
void solve()
{
	int n, q;
	cin >> n >> q;
	for(int i = 0; i <= n; i++) {
		adj[i].clear();
		vis[i] = false;
		st[i] = en[i] = cnt[i] = 0;
		depth[i] = 0;
		for(int j = 0; j < 20; j++) parent[i][j]
		    = 0;
	}
	for(int u = 2; u <= n; u++) {
		int _parent;
		cin >> _parent;
		adj[_parent].push_back(u);
	}
	ans = 0;
	cin >> colors;
	colors = "#" + colors;
	_time = 0;
	dfs(1, 0);
	for(int i = 1; i <= q; i++) {
		int u, v;
		cin >> u >> v;

		if(st[u] > st[v]) swap(u, v);
		int lca = getLCA(u, v);
		if(lca == u) Q[i].l = st[u], Q[i].r =
		    st[v];
		else Q[i].l = en[u], Q[i].r = st[v];

		Q[i].id = i;
	}
	sort(Q + 1, Q + q + 1);
	int l = 1, r = 0;
	for(int i = 1; i <= q; i++) {
		int L = Q[i].l, R = Q[i].r;

		if(R < l) {
			while(l > L) check(id[--l]);
			while(l < L) check(id[l++]);
			while(r < R) check(id[++r]);
			while(r > R) check(id[r--]);
		}
		else {
			while(r < R) check(id[++r]);
			while(r > R) check(id[r--]);
			while(l > L) check(id[--l]);
			while(l < L) check(id[l++]);
		}
		int u = id[l], v = id[r], lca =
		    getLCA(u, v);
		if(lca != u and lca != v) check(lca);
		res[Q[i].id] = ans;
		if(lca != u and lca != v) check(lca);
	}
	for(int i = 1; i <= q; i++) cout << res[i] <<
	    endl;
}
```

6.13 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;
}
```

6.14 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;
    }
}
```

7 7.Strings

7.1 Hasing

```
const int N = 2000006;
const ULL hs = 3797;
ULL F[N], FH[N], RH[N];
char str[N];
int n;
ULL Fhash(int x, int y) {
    return FH[y] - F[y - x + 1] * FH[x - 1];
}
ULL Rhash(int x, int y) {
    return RH[x] - F[y - x + 1] * RH[y + 1];
}
int main() {
    F[0] = 1;
    for(int i = 1; i < N; i++) F[i] = F[i - 1] * hs;
    scanf("%s", str + 1);
    n = strlen(str + 1);
    FH[0] = 0;
    FOR(i, 1, n) FH[i] = FH[i - 1] * hs + str[i];
    RH[n+1]=0;
    ROF(i, n, 1) RH[i] = RH[i + 1] * hs + str[i];
}
```

7.2 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;
}
```

7.3 Manacher's Algorithm

```
int p[lx];
```

```
void manachers_algorithm(string s)
{
    //First modify the string
    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];
        }
    }
}
```

7.4 Suffix Array

```
const int N = 2000006;
const int M = 22;
int n, stp, sfxMv, sfx[N], tmp[N], sfxSum[N],
     sfxCnt[N], Rank[M][N], int lcp[N], rnk[N];
char in[N];
char a[N], b[N];
inline bool Equal(const int &u, const int &v) {
    if(!stp) return in[u] == in[v];
    if(Rank[stp-1][u] != Rank[stp-1][v]) return false;
    int a = u + sfxMv < n ? Rank[stp-1][u+sfxMv] : -1;
    int b = v + sfxMv < n ? Rank[stp-1][v+sfxMv] : -1;
    return a == b;
}
void update() {
    int i, rnk;
    for(i = 0; i < n; i++) sfxSum[i] = 0;
    for(i = rnk = 0; i < n; i++) {
        sfx[i] = tmp[i];
        if(i && !Equal(sfx[i], sfx[i-1])) {
            Rank[stp][sfx[i]] = ++rnk;
            sfxSum[rnk+1] = sfxSum[rnk];
        }
        else Rank[stp][sfx[i]] = rnk;
        sfxSum[rnk+1]++;
    }
}
```

```
void Sort() {
    int i;
    for(i = 0; i < n; i++) sfxCnt[i] = 0;
    memset(tmp, -1, sizeof tmp);
    for(i = 0; i < sfxMv; i++) {
        int idx = Rank[stp - 1][n - i - 1];
        int x = sfxSum[idx];
        tmp[x + sfxCnt[idx]] = n - i - 1;
        sfxCnt[idx]++;
    }
    for(i = 0; i < n; i++) {
        int idx = sfx[i] - sfxMv;
        if(idx < 0)continue;
        idx = Rank[stp-1][idx];
        int x = sfxSum[idx];
        tmp[x + sfxCnt[idx]] = sfx[i] - sfxMv;
        sfxCnt[idx]++;
    }
    update();
    return;
}
inline bool cmp(const int &a, const int &b) {
    if(in[a]!=in[b]) return in[a]<in[b];
    return false;
}
void print() {
    for(int i=0; i<n; i++) {
        for(int j=sfx[i]; j<n; j++) printf("%c", in[j]);
        printf("\n");
    }
}
void suffixArray() {
    int i;
    for(i = 0; i < n; i++) tmp[i] = i;
    sort(tmp, tmp + n, cmp);
    stp = 0;
    update();
    ++stp;
    for(sfxMv = 1; sfxMv < n; sfxMv <<= 1) {
        Sort();
        stp++;
    }
    stp--;
    for(i = 0; i <= stp; i++) Rank[i][n] = -1;
}
void kasai() {
    for(int i=0; i<n; i++) rnk[ sfx[i] ] = i;
    for(int i = 0, k = 0; i < n; i++, k ? k-- : 0) {
        if(rnk[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sfx[ rnk[i] + 1 ];
        while(i + k < n && j + k < n && in[i + k] ==
             in[j + k]) k++;
        lcp[ rnk[i] ] = k;
    }
}
int main() {
    scanf("%s",in);
    n=strlen(in);
    suffixArray();
    print();
    kasai();
    for(int i=0; i<n; i++) cout << lcp[i] << '\n';
    return 0;
}
```

7.5 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;
}
```

7.6 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;
        }
    }
}
```

8 8.Xor and And

8.1 Maximum And Pair

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

8.2 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;
}
```

8.3 Minimum XOR Operation

- Minimum xor pair – Trie
    1. Sort the array
    2. Find min of ar[i]$^a r[i+1]$
    - Minimum XOR subarray – Trie
    - Minimum XOR of OR and AND in an array
    1. (x——y) $^(xY) min(x^y) Minimum XOR of all pair$
    - Sum of XOR of all elements of subsets
    1. $2^{n-1} \times$ (OR of whole array)
    - Maximum OR pair = Max element er sathe or jar beshi shetai answer.
    - Sum of or of all subset = $a_i * 2^{n-1}$ for all i

8.4 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;
}
```

8.5 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;
}
```

9 9.Misc

9.1 1,Histogram

```
int main() {
    int t;
    scanf("%d",&t);
    FOR(tc,1,t) {
        ll n;
        scanf("%lld",&n);
        ll arr[n+1];
        REP(i,n) scanf("%lld",&arr[i]);
        stack<ll>st;
        st.push(0);
        ll ans=-1;
        FOR(i,1,n) {
            if(i==n) {
                while(!st.empty()) {
                    ll j=st.top();
                    st.pop();
                    if(!st.empty()) {
            ans=max(ans,arr[j]*(i-st.top()-1));
                    }
                    else {
                        ans=max(ans,arr[j]*i);
                        break;
                    }
                }
```

```
            }
            else {
                if(arr[i]>=arr[st.top()]) {
                    st.push(i);
                }
                else {
                    while(arr[st.top()]>arr[i]) {
                        ll j=st.top();
                        st.pop();
                        if(!st.empty()) {
                ans=max(ans,arr[j]*(i-st.top()-1));
                        }
                        else {
                            ans=max(ans,arr[j]*i);
                            break;
                        }
                    }
                    st.push(i);
                }
            }
        }
        printf("Case %d: %lld\n",tc,ans);
    }
    return 0;
}
```

9.2 2,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;
```

9.3 3,Knight Move in Infinite Grid

```
// Minimum number of knight moves from (x,y) to
// (0,0) in non-negative infinite chessboard
ll knight_move(ll x, ll y) {
    ll cnt = max({(x + 1) / 2, (y + 1) / 2, (x + y + 2)
         / 3});
    while((cnt % 2) != (x + y) % 2) cnt++;
    if(x == 1 && !y) return 3;
    if(y == 1 && !x) return 3;
    if(x == y && x == 2) return 4;
    return cnt;
}
```

9.4 4,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() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    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;
}
```

9.5 5,MO Algorithm

```
int id,n,q,block_size,ans;
int arr[MAX],answer[MAX],freq[MAX],cnt[2*MAX];
pii update[MAX];
pair<pii,pii>qry[MAX];
map<int,int>mp;
bool mo_cmp(pair<pii,pii>x,pair<pii,pii>y) {
    if(x.ff.ff/block_size!=y.ff.ff/block_size)
        return x.ff.ff/block_size<y.ff.ff/block_size;
    if(x.ff.ss/block_size!=y.ff.ss/block_size)
        return x.ff.ss/block_size<y.ff.ss/block_size;
    return x.ss.ff < y.ss.ff;
}
void add(int x) {
    freq[cnt[x]]--;
    cnt[x]++;
    freq[cnt[x]]++;
}
void Remove(int x) {
    freq[cnt[x]]--;
    cnt[x]--;
    freq[cnt[x]]++;
}
void _update(int i,int u,int v) {
    int idx=update[i].ff;
    int val=update[i].ss;
    if(idx>v or idx<u) swap(arr[idx],update[i].ss);
    else {
        Remove(arr[idx]);
        add(val);
        swap(arr[idx],update[i].ss);
    }
}
int main() {
    FastRead cin >> n >> q;
    FOR(i,1,n) {
        cin >> arr[i];
        if(mp[arr[i]]==0) {
            mp[arr[i]]=++id;
            arr[i]=id;
        }
        else arr[i]=mp[arr[i]];
    }
    int up=0,qr=0;
    REP(i,q) {
        int u,v,w;
        cin >> u >> v >> w;
        if(u==1) {
            qry[qr]=mk(pii(v,w),pii(up,qr));
            qr++;
        }
        else {
            if(mp[w]==0) mp[w]=++id;
            update[++up]=pii(v,mp[w]);
        }
    }
    block_size=cbrt(n)*cbrt(n);
    sort(qry,qry+qr,mo_cmp);
    int ml=1,mr=0,mu=0;
    REP(i,qr) {
        int l=qry[i].ff.ff;
        int r=qry[i].ff.ss;
        int u=qry[i].ss.ff;
        while(mu<u) {
            mu++;
            _update(mu,ml,mr);
        }
        while(mu>u) {
            _update(mu,ml,mr);
            mu--;
        }
        while(mr<r) {
            mr++;
            add(arr[mr]);
        }
        while(mr>r) {
            Remove(arr[mr]);
            mr--;
        }
        while(ml<l) {
            Remove(arr[ml]);
            ml++;
        }
        while(ml>l) {
            ml--;
            add(arr[ml]);
        }
        FOR(j,1,700) {
            if(freq[j]==0 and answer[qry[i].ss.ss]==0) {
                answer[qry[i].ss.ss]=j;
                break;
            }
        }
    }
    REP(i,qr) cout << answer[i] << '\n';
    return 0;
}
```

9.6 6,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);
    }
}
```

9.7 7,Order Set

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
template<typename T>
using ordered_set=tree<T,null_type,less<T>,
rb_tree_tag,tree_order_statistics_node_update>;
typedef
     tree<pair<int,int>,null_type,less<pair<int,int>>,
rb_tree_tag,tree_order_statistics_node_update>ordered_set;
template<typename T> using ordered_set=
tree<T,null_type,less<T>,rb_tree_tag,
tree_order_statistics_node_update>;
```

9.8 8,Contest WA

```
Pre-submit:
Write a few simple test cases if sample is not enough.
Are time limits close? If so, generate max cases.
Is the memory usage fine?
Could anything overflow?
Make sure to submit the right file.
Do not use endl. Use "\n"
fastio include on first line of the program.

Wrong answer:
```

```
Print your solution! Print debug output, as well.
Are you clearing all data structures between test cases?
Can your algorithm handle the whole range of input?
Read the full problem statement again.
Do you handle all corner cases correctly?
Have you understood the problem correctly?
Any uninitialized variables?
Any overflows?
Confusing N and M, i and j, etc.?
Are you sure your algorithm works?
What special cases have you not thought of?
Are you sure the STL functions you use work as you
    think?
Add some assertions, maybe resubmit.
Create some testcases to run your algorithm on.
Go through the algorithm for a simple case.
Go through this list again.
Explain your algorithm to a teammate.
Ask the teammate to look at your code.
Go for a small walk, e.g. to the toilet.
```

```
Is your output format correct? (including whitespace)
Rewrite your solution from the start or let a teammate
    do it.

Runtime error:
Have you tested all corner cases locally?
Any uninitialized variables?
Are you reading or writing outside the range of any
    vector?
Any assertions that might fail?
Any possible division by 0? (mod 0 for example)
Any possible infinite recursion?
Invalidated pointers or iterators?
Are you using too much memory?
Debug with resubmits (e.g. remapped signals, see
    Various).

Time limit exceeded:
Do you have any possible infinite loops?
What is the complexity of your algorithm?
```

```
Are you copying a lot of unnecessary data? (References)
How big is the input and output? (consider scanf)
Avoid vector, map. (use arrays/unordered_map)
What do your teammates think about your algorithm?

Memory limit exceeded:
What is the max amount of memory your algorithm should
    need?
Are you clearing all data structures between test cases?

Corner Case:
Graph: Self Loop, Multi Edge, Disconnected Graph
Game Theory: Pattern can appear after few operations so
    do 30/40 with bruteforce recursion
Geomatry: Good OP: +, -, *, /2 .
Do not use bad operations. Find another way.
Use tanQ for more accurate.
floating point accuracy compare difference <= 10^-9
```