CoU_Unpredictable_3207

Comilla University

- Symmetry rule: $\binom{n}{k} = \binom{n}{n-k}$
- Factoring in: $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Number of elements e, $gcd(e, n) = d$ equal to $\phi(\frac{n}{d})$.

$$\sum_{m=0}^{n} \binom{m}{k} = \binom{n+1}{k+1}$$

$$\binom{n}{0}^2 + \binom{n}{1}^2 + \cdots + \binom{n}{n}^2 = \binom{2n}{n}$$

$$\sum_{k=0}^{m} \binom{n+k}{k} = \binom{n+m+1}{m}$$

$$1\binom{n}{1} + 2\binom{n}{2} + \cdots + n\binom{n}{n} = n2^{n-1}$$

Number of squares in an m*n squares (m>n) = $\frac{n(n+1)(3m-n+1)}{6}$

onacci numbers:

$$\binom{n}{0} + \binom{n-1}{1} + \cdots + \binom{n-k}{k} + \cdots + \binom{0}{n} = F_{n+1}$$

$$\varphi(p^k) = p^{k-1} * \varphi(p)$$

so on till a square of dimensions m*m is reached that is the outer square. Hence the total number of squares becomes:

$m^2 + (m-1)^2 + (m-2)^2 + \ldots + 3^2 + 2^2 + 1^2 = {}^{m(m+1)(2m+1)}/_{6}$

Hence for any m*n grid, the number of ways to diagonally from one end to the another is given by ${}^{(m+n)!}/_{m!n!}$

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_m}\right).$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

70. $\sum_{i=0}^{n} i \cdot i! = (n+1)! - 1.$

Prove that the sum of the positive integer divisors of

$$\prod_{i=1}^{r} \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

Number of pairs with given LCM (L) and GCD (G)

```
So finally,
If L/G has k unique prime factors, answer will be (2^k) / 2.
```

$$\sum_{d|n} \phi(d) = n.$$

```
//#include <ext/pb_ds/assoc_container.hpp>
//#include <ext/pb_ds/tree_policy.hpp>
#include<bits/stdc++.h>

//using    namespace __gnu_pbds;
using namespace std;

/*** Typedef ***/
typedef long long ll;
typedef unsigned long long ull;

/*** Loops ***/
#define foR0(num) for(ll i = 0; i < num; i++)
#define foR1(num) for(ll i = 1; i <= num; i++)
#define foRev(num) for(ll i = num - 1; i >= 0; i--)
#define rep(i, x, n) for (ll i = x, _n = (n); i < _n; ++i)
#define forIn(arr, num) for(ll i = 0; i < num; i++) cin>>arr[i];
#define forIn1(arr, num) for(ll i = 1; i <= num; i++) cin>>arr[i];
#define vpnt(ans) for(ll i = 0; i < ans.size(); i++) cout << ans[i] << (i + 1 < ans.size() ? ' ' : '\n');
#define apnt(arr, num) for(ll i = 0; i < num; i++) cout << arr[i] << (i + 1 < num ? ' ' : '\n');

/*** Input ***/
#define sci1(a) scanf("%d",&a)
#define sci2(a,b) scanf("%d %d",&a,&b)
#define scln1(a) scanf("%lld",&a)
#define scln2(a,b) scanf("%lld %lld",&a,&b)
#define scln3(a,b,c) scanf("%lld %lld %lld",&a,&b,&c)

/*** Output ***/
#define pf1(a) printf("%d\n",a)
#define pf2(a,b) printf("%d %d\n",a,b)
#define pfln1(a) printf("%lld\n",a)
#define pfln2(a,b) printf("%lld %lld\n",a,b)

/*** Define Values ***/
#define    ff           first
#define    ss           second
#define    re           return
#define    MP           make_pair
#define    pb           push_back
#define    SZ(x)        ((ll) (x).size())
#define    EPS          10E-10
#define    mxx          100005

#define    Ceil(a,b)       (a+b-1)/b
#define    gcd(a, b)       __gcd(a,b)
#define    min3(a,b,c)     min(a,min(b,c))
#define    max3(a,b,c)     max(a,max(b,c))
#define    lcm(a, b)       ((a)/gcd(a,b))*(b)
#define    min4(a,b,c,d)   min(d,min(a,min(b,c)))
#define    max4(a,b,c,d)   max(d,max(a,max(b,c)))
#define    input           freopen("input.txt","rt", stdin)

#define    MOD          1000000007
#define    iseq(a,b)    (fabs(a-b)<EPS)
#define    PI           3.1415926535897932384626 43
#define    output       freopen("output.txt","wt", stdout)
#define    all(v)       v.begin(),v.end()
#define    mem(nam,val)    memset(nam, val, sizeof(nam))
#define    ps(x,y)      fixed<<setprecision(y)<<x
#define    for2D0(n,m)     for(ll i=0;i<n;i++)for(ll j=0;j<m;j++)
#define    for2D1(n,m)     for(ll i=1;i<=n;i++)for(ll j=1;j<=m;j++)
#define    Unique(X)       (X).resize(unique(all(X)) -(X).begin())
#define    get_pos(c,x)    (lower_bound(c.begin(),c.end(),x)-c.begin())
#define    get_pos_up(c,x)  (upper_bound(c.begin(),c.end(),x)-c.begin())
#define    IOS          ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
#define    for2Dpnt(arr,n,m) for(ll i=0;i<n;i++){for(ll j=0;j<m;j++)cout<<arr[i][j]<<" ";cout<<endl;}

typedef vector <ll> vll;
typedef multiset <ll> msll;
typedef queue <ll> qll;
typedef stack <ll> stll;
typedef map <ll, ll> mll;
typedef pair <ll, ll> pll;
typedef vector <pair <ll , ll> > vpll;

/*** BitWise Operations
///int Set(int N,int pos){return N=N | (1<<pos);}
///int reset(int N,int pos){return N= N & ~(1<<pos);}
///bool check(int N,int pos){return (bool)(N & (1<<pos));}
***/
```

```
///const int fx[] = {+1,-1,+0,+0};
///const int fy[] = {+0,+0,+1,-1};
///const int fx[] = {+0,+0,+1,-1,-1,+1,-1,+1}; ///King's move
///const int fy[] = {-1,+1,+0,+0,+1,+1,-1,-1}; ///king's Move
///const int fx[] = {-2,-2,-1,-1,+1,+1,+2,+2}; ///knight's move
///const int fy[] = {-1,+1,-2,+2,-2,+2,-1,+1}; ///knight's move

///transform(data.begin(), data.end(), data.begin(),[](unsigned char c){ return std::tolower(c); });
///typedef tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> ordered_set;
///ll toint(string s){ll n=0,k=1;for(int i=s.size()-1; i>=0; i--){n += ((s[i]-'0')*k);k*=10;}return n;}
///string tostring(ll x){string s="";while(x){s += (x%10)+'0';x/=10;}reverse(s.begin(),s.end());return s;}
///bool sortinrev(const pair<ll,ll> &a,const pair<ll,ll> &b)return (a.first > b.first);
///prime[]={2,4,2,4,6,2} //start loop from 29 to do prime factorization
///auto it = lower_bound(my_multiset.begin(), my_multiset.end(), 3);
///const auto pos = distance(my_multiset.begin(), it);
///priority_queue< pll ,vector<pll>,greater<pll> >p;
///lower_bound(all(v),r+1)-lower_bound(all(v),l);
///cout<<*X.find_by_order(0)<<endl;
///cout<<X.order_of_key(-5)<<endl;
///set< pair<int,int> >s;
///pair<int,int> p={3,2};
///auto lb=lower_bound(s.begin(),s.end(),p);
///cout<<(*lb).first<<" "<<(*lb).second<<endl;
***/
 //__uint128_t
```

Number Of Divisors from 1 to N:

```
ll Divisors[1000000];
void Div()
{
   for(ll i=0;i<=1000000;i++)Divisors[i]  = 1;
   for (ll i=2;i<=1000000;i++)
   {
      for (ll j = 1; j * i<=1000000;  j++)Divisors[i*j]++;
   }
}
```

Compare Structure

```
struct comp
{
   template<typename T>
   bool operator()(const T& l, const T& r) const
   {
      if (l.first == r.first)
         return l.second > r.second;

      return l.first < r.first;
   }
};
```

Bitwise Sieve

```
bool Check(int N,int pos){return (bool)(N & (1<<pos));}
int Set(int N,int pos){ return N=N | (1<<pos);}
int mxx=100000009,prime[5761500],cnt=1;
int status[3125500];bitset<100000009>store;
void sieve()
{
   int i, j, sqrtN;
   sqrtN = int( sqrt( mxx ) );
   store.set();

   for( i = 3; i <= sqrtN; i += 2 )
   {
      if( Check(status[i>>5],i&31)==0)
      {
         for( j = i*i; j <= mxx; j += (i<<1) )
         {
            status[j>>5]=Set(status[j>>5],j & 31)   ;
         }
      }
   }
   prime[0]=2;
   j=1;store[2]=false;
   for(i=3; i<=mxx; i+=2)
   {
      if( Check(status[i>>5],i&31)==0)
         prime[j++]=i,cnt++,store[i]=false;
   }
   printf("%d\n",cnt);
}
```

SUM of Phi(n)

10 ------------------------------ 32
100 ---------------------------- 3044
1000 --------------------------- 304192
10000 ------------------------- 30397486
100000 ------------------------ 3039650754
1000000 ----------------------- 303963552392
10000000 ---------------------- 30396356427242

Prime Numbers

n	x	$\pi(x)$
1	10	4
2	100	25
3	1,000	168
4	10,000	1,229
5	100,000	9,592
6	1,000,000	78,498
7	10,000,000	664,579
8	100,000,000	5,761,455
9	1,000,000,000	50,847,534
10	10,000,000,000	455,052,511
11	100,000,000,000	4,118,054,813
12	1,000,000,000,000	37,607,912,018
13	10,000,000,000,000	346,065,536,839
14	100,000,000,000,000	3,204,941,750,802
15	1,000,000,000,000,000	29,844,570,422,669
16	10,000,000,000,000,000	279,238,341,033,925
17	100,000,000,000,000,000	2,623,557,157,654,233
18	1,000,000,000,000,000,000	24,739,954,287,740,860
19	10,000,000,000,000,000,000	234,057,667,276,344,607
20	100,000,000,000,000,000,000	2,220,819,602,560,918,840
21	1,000,000,000,000,000,000,000	21,127,269,486,018,731,928
22	10,000,000,000,000,000,000,000	201,467,286,689,315,906,290
23	100,000,000,000,000,000,000,000	1,925,320,391,606,803,968,923
24	1,000,000,000,000,000,000,000,000	18,435,599,767,349,200,867,866
25	10,000,000,000,000,000,000,000,000	176,846,309,399,143,769,411,680

Maximum Number of Divisors

1. 10^1 = 4
2. 10^2 = 12
3. 10^3 = 32
4. 10^4 = 64
5. 10^5 = 128
6. 10^6 = 240
7. 10^7 = 448
8. 10^8 = 768
9. 10^9 = 1344
10. 10^10 = 2304
11. 10^11 = 4032
12. 10^12 = 6720
13. 10^13 = 10752
14. 10^14 = 17280
15. 10^15 = 26880
16. 10^16 = 41472
17. 10^17 = 64512
18. 10^18 = 103680
19. 10^19 = 161280

$$lcmSum(n) = \sum_{i=1}^{n} lcm(i,n)$$

$$lcmSum(n) = \frac{n}{2} \times \left(\sum_{d|n} \phi(d).d + 1\right)$$

GCD Sum Function – $g(n)$

$$g(n) = \prod_{i=0}^{k} (a_i + 1)p_i^{a_i} - a_i p^{a_i - 1}$$

$$g(n) = n \sum_{d|n} \frac{\phi(d)}{d}$$

Number of Values that GCD(N,x)<=Y

$$= \sum_{d|Y} \varphi(N/d)$$

Finding More Solutions

Suppose we found a solution (x, y) for $Ax + By = C$.
$(x + k\frac{B}{g}, y - k\frac{A}{g})$, where k is any integer.

Catalan Numbers

$$C_n = \binom{2n}{n} - \binom{2n}{n-1} = \frac{1}{n+1}\binom{2n}{n}, n \geq 0$$

Theorem: If we have K distinguishable containers and N indistinguishable balls, then we can distribute them in $\binom{N+K-1}{N}$ ways.

What if every partition needs to have at least one star?

Theorem: For any pair of positive integers N and K, the number of K-tuples of positive integers whose sum is N is equal to the binomial coefficient $\binom{N-1}{K-1}$.

Theorem: For any pair of positive integers N and K, the number of K-tuples of non-negative integers whose sum is less than or equal to N is $\binom{N+K}{K}$.

Modular Inverse

M is prime:

```
int  x = bigmod( a, M - 2, M );
```

From 1 to N:

```
ll inv[mxx];

void inv_fun(ll n,ll m){

   inv[1] = 1;

   for ( int i = 2; i <= n; i++ ){

      inv[i] = (-(m/i) * inv[m%i] ) % m;

      inv[i] = inv[i] + m;

   }

}
```

M is Not Prime:

```
int modInv ( int a, int m )

{

   int x, y;

   int g = gcdExtended( a, m, &x, &y );

   if(g!=1)return -1;

   x %= m;

   if ( x < 0 ) x += m;

   return x;

}
```

```
/// Sum of digits
long long int DP[12][180][2];
vector<int> num;
long long int solve(int pos,int sum,int f){
   if(pos==num.size())
      return sum;
   if (DP[pos][sum][f] !=-1) return DP[pos][sum][f];
   long long int res=0;
   int lmt;
   if(f==0)
      lmt=num[pos];
   else lmt=9;
   for(int dgt=0; dgt<=lmt; dgt++)
   {
      int nf=f;
      if(f==0 && dgt<lmt) nf=1;
      res+=solve(pos+1,sum+dgt,nf);
   }
   return DP[pos][sum][f]=res;
}
```

```
/// Articulation Point
bitset<10017> is_visited;
vector<long long> low, dtime;
set<long long>artipoint;
vector<vector<long long>> adjlist;
int minutes;
void articulationpoints(long long u, long long p = -1){
   ++minutes; is_visited[u] = true;
   low[u] = dtime[u] = minutes;
   int child = 0;
   for(auto i:adjlist[u]) {
      if(i == p)
         continue;
      if(is_visited[i]) {
         low[u] = min(low[u], dtime[i]);
      }
      else {
         articulationpoints(i, u);
         low[u] = min(low[u], low[i]);
         if (dtime[u] <= low[i] && p != -1)
            artipoint.insert(u);
         child++;
      }
   }
   if (p == -1 && child > 1)
      artipoint.insert(u);
}
```

Prime Factorization

```
vll prime_fact(ll n)
{
   vll fact;
   for (int d :{2, 3, 5})
   {
      while (n % d == 0)
      {
         fact.pb(d);
         n /= d;
      }
   }
   static array<int, 8> inc =
 {4, 2, 4, 2, 4, 6, 2, 6};
   int i = 0;
   for (ll d = 7; d * d <= n; d += inc[i++])
   {
      while (n % d == 0)
      {
         fact.pb(d);
         n /= d;
      }
      if (i == 8)i = 0;
   }
   if (n > 1)
      fact.pb(n);
   return fact;
}
```

Euler Totient(Seive)

```
ll phi[mxx+2];
void calculatePhi()
{
   for(ll i=1; i<=mxx; i++)phi[i] = i;

   for(ll i =2; i<=mxx; i++)
   {
      if(phi[i]==i)
      {
         for(ll j=i; j<=mxx; j+=i)
            phi[j]-=phi[j]/i;
      }
   }
}
```

NOD(1/3)

```
void prime()
bool MillerRabin(u64 n)
ll divisor(ll n)
{
   ll res=1,cnt,x=n,y=sqrt(n);
   for(ll i=0;  ;i++)
   {
      ll p=arr[i];
      if(p*p*p>n)break;
      if(n%p==0)
      {
         cnt=1;
         while(n%p==0)n/=p,cnt++;
         res*=cnt;
      }

   }
   if(MillerRabin(n))res*=2;
   else
   {
      ll val=sqrt(n);
      if(val*val==n&&MillerRabin(val))
            res*=3;
      else if(n!=1)
            res*=4;
   }
   return res;
}
```

Kadane’s Algorithm

```
int maxSumSubArray(int a[],int n)
{
   int cnt_max[n];
   int res = INT_MIN;
   cnt_max[0] = a[0];
   for(int i=1;i<n;i++)
   {
      cnt_max[i] = max(a[i],cnt_max[i-1]+a[i]);
      if(cnt_max[i]>res)
         res = cnt_max[i];
   }
   return res;
}
```

Binomial Coefficient

```
ll NcR(ll n, ll r)
{
   if(r > n - r) r = n - r;
   ll ans = 1,i;
   for(i = 1; i <= r; i++)
   {
      ans *= n - r + i;
      ans /= i;
   }
   return ans;
}
```

Euler Totient

```
ll phi(ll n)
{
   ll result = n;
   for (ll p = 2; p * p <= n; ++p) {

      if (n % p == 0) {

         while (n % p == 0)
            n /= p;
         result -= result / p;
      }
   }

   if (n > 1)
      result -= result / n;
   return result;
}
```

SNOD

```
int SNOD( int n ) {
   int res = 0;
   int u = sqrt(n);
   for ( int i = 1; i <= u; i++ ) {
      res += ( n / i ) - i; //Step 1
   }
   res *= 2; //Step 2
   res += u; //Step 3
   return res;
}
```

Miller Rabin

```
using u64 = uint64_t;
using u128 = __uint128_t;
u64 binpower(u64 base, u64 e, u64 mod)
{
   u64 result = 1;
   base %= mod;
   while (e)
   {
      if (e & 1)result = (u128)result * base % mod;
      base = (u128)base * base % mod;
      e >>= 1;
   }
   return result;
}
bool check_composite(u64 n, u64 a, u64 d, int s)
{
   u64 x = binpower(a, d, n);
   if (x == 1 || x == n - 1)return false;
   for (int r = 1; r < s; r++)
   {
      x = (u128)x * x % n;
      if (x == n - 1)return false;
   }
   return true;
}
bool MillerRabin(u64 n)
{
   if (n < 2)return false;
   int r = 0;
   u64 d = n - 1;
   while ((d & 1) == 0)
   {
      d >>= 1; r++;
   }
   for (int a :{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37})
   {
      if (n == a)return true;
      if (check_composite(n, a, d, r))return false;
   }
   return true;
}
int main()
{
   u64 a;
   cin>>a;
   bool check=MillerRabin(a);
   if(check)cout<<"Prime"<<endl;
   return 0;
}
```

Inclusion Exclusion

```
int ans;
void recurs(int ara[], int i, int j, int num, int numofele, int n)
{
   if (i+1 == numofele) return ;
   int x, y;
   for (x = i; x <numofele; x++)
   {
      y = lcm(ara[x], j);

      if ((num+1)%2==1) ans+=(n/y);
      else ans-=(n/y);
      recurs(ara, x+1, y, num+1, numofele, n);
   }
}
int main(){
   int ara[] = {2, 3, 5, 6, 7, 11, 13, 15, 17},n=1000,m=9;
   recurs(ara, 0, 1, 0, m, n);
}
```

Linear Diophantine Equation

```
bool linearDiophantine ( int A, int B, int C, int &x, int &y ) {
   int g = gcd ( A, B );
   if ( C % g != 0 ) return false;

   int a = A / g, b = B / g, c = C / g;
   extended_euclid( a, b, x, y );

   if ( g < 0 ) {
      a *= -1; b *= -1; c *= -1;
   }
   x *= c; y *= c;
   return true;
}
```

Extend Euclid

```
ll extended_euclid(ll a, ll b, ll &x, ll &y)
{
   if(b==0)
   {
      x=1;y=0;
      return a;
   }
   ll x1,y1;
   ll temp=extended_euclid(b,a%b,x1,y1);
   x=y1;
   y=x1-y1*(a/b);
   return temp;
}
```

Extended Euclid In Range

```
void shift_solution(ll &x,ll &y,ll a,ll b,ll cnt)
{
    x += cnt * b;
    y -= cnt * a;
}
ll gcd(ll a,ll b,ll &x,ll &y)
{
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    ll x1, y1;
    ll d = gcd(b%a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;
}
bool find_any_solution(ll a, ll b, ll c, ll &x0 ,ll &y0, ll
&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;
}
ll find_all_solutions(ll a, ll b, ll c, ll minx, ll maxx, ll
miny, ll maxy)
{
    ll x, y, g;
    if (!find_any_solution(a, b, c, x, y, g))return 0;
    a /= g;
    b /= g;
    ll sign_a = a > 0 ? +1 : -1;
    ll sign_b = b > 0 ? +1 : -1;
    shift_solution(x, y, a, b, (minx - x) / b);
    if (x < minx)shift_solution(x, y, a, b, sign_b);
    if (x > maxx)return 0;
    ll lx1 = x;
    shift_solution(x, y, a, b, (maxx - x) / b);
    if (x > maxx)shift_solution(x, y, a, b, -sign_b);
    ll rx1 = x;
  shift_solution(x, y, a, b, -(miny - y) / a);
    if (y < miny)shift_solution(x, y, a, b, -sign_a);
    if (y > maxy)return 0;
    ll lx2 = x;
    shift_solution(x, y, a, b, -(maxy - y) / a);
    if (y > maxy)shift_solution(x, y, a, b, sign_a);
    ll rx2 = x;
    if (lx2 > rx2)swap(lx2, rx2);
    ll lx = max(lx1, lx2);
    ll rx = min(rx1, rx2);
    if (lx > rx)return 0;
    return (rx - lx) / abs(b) + 1;
}
int main()
{
    ll t;
    scanf("%lld",&t);
    for(ll i=1; i<=t; i++) {
        ll a,b,c,x1,x2,y1,y2;
        scanf("%lld %lld %lld %lld %lld
%lld%lld",&a,&b,&c,&x1,&y1,&x2,&y2);
        printf("Case %lld: ",i);
        c*=-1;
        if(a==0||b==0) {
            if(a==0&&b==0) {
                if(c==0)printf("%lld\n",(abs(y1-
x1)+1)*(abs(y2-x2)+1));
                else printf("0\n");
            }
            else if(a==0) {
                if(c%b!=0)printf("0\n");
                else{
                    c/=b;
                    if(c>=x2&&c<=y2)printf("%lld\n",abs(y1-
x1)+1);
                    else printf("0\n");
                }
            }
            else{
                if(c%a!=0)printf("0\n");
                else{
                    c/=a;
                    if(c>=x1&&c<=y1)printf("%lld\n",abs(y2-
x2)+1);
                    else printf("0\n");
                }
            }
            continue;
        }
        printf("%lld\n",find_all_solutions(a,b,c,x1,y1,x2,
y2));
    }
}
```

```
vector<long long>Node[100005],cost[100005];
long long n,m,i,j,cc=0,k;
long long dis[100005],parent[100005];
long long inf=10e9;
void bellmenford(long long s,long long f) {
   for(i=1; i<=n; i++) {
      if(i==s)dis[i]=0;
      else dis[i]=inf;
      parent[i]=-1;
   }
   for(i=1; i<n; i++) {
      bool done=true;
      for(j=1; j<=n; j++) {
         for(k=0; k<Node[j].size(); k++) {
            long long u=j,v=Node[j][k],uv=cost[j][k];
            if(dis[u]+uv<dis[v]) {
               dis[v]=dis[u]+uv;
               parent[v]=u;
               done=false;
            }
         }
      }
      if(done)break;/// there was nothing to update ;
   }
/// Looking for Cycle ;
   bool found=true;
   for(i=1; i<=n; i++) {
      for(j=0; j<Node[i].size(); j++) {
         long long u=i,v=Node[i][j],uv=cost[i][j];
         if(dis[u]+uv<dis[v]) {
            cout<<"Found Negative Cycle"<<endl;
            found=false;
            return;
         }
      }
      if(!found)break; }
   for(i=1; i<=n; i++)
      cout<<"NODE : "<<i<<" distance : "<<dis[i]<<endl;
}
```

A number is Fibonacci if and only if one or both of $(5 \cdot n^2 + 4)$ or $(5 \cdot n^2 - 4)$ is a perfect square

Every third number of the sequence is even and more generally, every k^{th} number of the sequence is a multiple of F_k

$gcd(F_m, F_n) = F_{gcd(m,n)}$

Any three consecutive Fibonacci numbers are pairwise coprime, which means that, for every n, $gcd(F_n, F_{n+1}) = gcd(F_n, F_{n+2}), gcd(F_{n+1}, F_{n+2}) = 1$

If the members of the Fibonacci sequence are taken $mod\ n$, the resulting sequence is periodic with period at most $6n$.

. **Derangement:** a permutation of the elements of a set, such that no element appears in its original position. Let $d(n)$ be the number of derangements of the identity permutation fo size n.

$$d(n) = (n-1) \cdot (d(n-1) + d(n-2)) \text{ where } d(0) = 1, d(1) = 0$$

Bipartite Graph

```
ll n;
vll adj[mxx*3];
bool is_bipartite;
pll bip(vll &side,ll st)
{
   ll parity[3]={};    side[st] = 0;
   queue<ll> q;  q.push(st);
   while (!q.empty()) {
      ll v = q.front(); q.pop();
      parity[side[v]]++;
      for (ll u : adj[v]) {
         if (side[u] == -1) {
            side[u] = side[v] ^ 1;
            q.push(u);
         }
         else is_bipartite &= side[u] != side[v];
      }
   }
   return {parity[0],parity[1]};
}
void solve()
{
   vll side(n, -1);
   vector< pll >res;
   is_bipartite = true;
   for(ll st=0;st<n;st++)
   {
      if(side[st]==-1)
      {
         pll p=bip(side,st);
         if(is_bipartite)res.pb({p.ff,p.ss});
      }
   }
   if(!is_bipartite)
       cout<<"Not Bipartite"<<endl;
   else
       for(ll i=0;i<SZ(res);i++)
           cout<<res[i].ff<<" - "<<res[i].ss<<endl;
}
```

Dijkstra Algorithm

```
ll n,edge;
const long long int INF = 1e15;
vector< pair<ll,ll> > adj[100002];
vector<ll> restore_path(ll s, ll t, vector<ll> const& p)
{
   vector<ll> path;
   for (ll v = t; v != s; v = p[v])path.push_back(v);
   path.push_back(s);
   reverse(path.begin(), path.end());
   return path;
}

void dijkstra(ll s, vector<ll> & d, vector<ll> & p)
{
   d.assign(n+1, INF);
   p.assign(n+1, -1);
   d[s] = 0;

   set<pair<ll, ll>> q;
   q.insert({0, s});
   while (!q.empty())
   {
      ll v = q.begin()->second;
      q.erase(q.begin());

      for (auto edge : adj[v])
      {
         ll to = edge.first;
         ll len = edge.second;

         if (d[v] + len < d[to])
         {
            q.erase({d[to], to});
            d[to] = d[v] + len;
            p[to] = v;
            q.insert({d[to], to});
         }
      }
   }
}
```

Mobius inversion theorem: The classic version states that if g and f are arithmetic functions satisfying $g(n) = \sum_{d|n} f(d)$ for every integer $n \geq 1$ then

$$g(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) \text{ for every integer } n \geq 1$$

If $F(n) = \prod_{d|n} f(d)$, then $F(n) = \prod_{d|n} F\left(\frac{n}{d}\right)^{\mu(d)}$

Lucas Theorem

```
ll NCRmodP(ll n, ll r,ll p)
{
   if(n < r)return 0 ;
   ll den = (fact[r]*fact[(n-r)])%p;
   den =bigmod(den, p-2, p);
   return (fact[n]*den)%p;
}
ll Divider_Maker(ll n, ll r, ll p)
{
   if( n==0 && r==0)return 1;
   ll N = n%p, R = r%p;
   ll i =  NCRmodP(N, R, p);
   return (Divider_Maker(n/p, r/p, p) * i)%p;
}
ll Locus_Result(ll n, ll r, ll p)
{
   fact[0]=1;
   for(int i=1; i<p; i++)fact[i]=(i*fact[i-1])%p;
   return Divider_Maker(n, r, p);
}
```

Prime factorization of N!

```
void factFactorize ( ll n )
{
   for ( ll i = 0; i < prime.size() && prime[i] <= n; i++ )
   {
      ll x = n;
      ll freq = 0;

      while ( x / prime[i] )
      {
         freq += x / prime[i];
         x = x / prime[i];
      }

      printf ( "%d^%d\n", prime[i], freq );
   }
}
```

Chinese Reminder theorem

```
pair<ll, ll> CRT( vector<ll> A, vector<ll> M )
{
   if(A.size() != M.size()) return {-1,-1};
   ll n = A.size();
   ll a1 = A[0],m1 = M[0];
   for ( ll i = 1; i < n; i++ )
   {
      ll a2 = A[i],m2 = M[i];
      ll g = __gcd(m1, m2);
      if ( a1 % g != a2 % g ) return {-1,-1};
      ll p, q;
      extended_euclid(m1/g, m2/g, p, q);
      ll mod = m1 / g * m2;
     ll x = ((__int128)a1*(m2/g)*q + (__int128)a2*(m1/g)*p) %
mod;
     a1 = x;
      if (a1 < 0) a1 += mod;
      m1 = mod;
   }
   return {a1, m1};
}
```

Base to Decimal

```
ll baseToDecimal ( string x, ll base )
{
   ll res = 0;
   ll len = x.length();

   ll coef = 1;
   for ( int i = len - 1; i >= 0; i-- )
   {
      res += (x[i]-'0') * coef;
      coef *= base; // increase power of base
   }
   return res;
}
```

. **Combination with repetition:** Let's say we choose k elements from an n-element set, the order doesn't matter and each element can be chosen more than once. In that case, the number of different combinations is: $\binom{n+k-1}{k}$

. Number of ways to divide n persons in $\frac{n}{k}$ equal groups i.e. each having size k is

$$\frac{n!}{k!^{\frac{n}{k}}\left(\frac{n}{k}\right)!} = \prod_{n \geq k}^{n-=k} \binom{n-1}{k-1}$$

. $\sum_{0\le k\le n}\binom{n-k}{k} = Fib_{n+1}$

. $\binom{n}{k} = \binom{n}{n-k}$

. $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$

. $k\binom{n}{k} = n\binom{n-1}{k-1}$

. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$

. $\sum_{i=0}^{n}\binom{n}{i} = 2^n$

. $\sum_{i\ge 0}\binom{n}{2i} = 2^{n-1}$

. $\sum_{i\ge 0}\binom{n}{2i+1} = 2^{n-1}$

. $\sum_{i=0}^{k}(-1)^i\binom{n}{i} = (-1)^k\binom{n-1}{k}$

. $\sum_{i=0}^{k}\binom{n+i}{i} = \sum_{i=0}^{k}\binom{n+i}{n} = \binom{n+k+1}{k}$

. $1\binom{n}{1} + 2\binom{n}{2} + 3\binom{n}{3} + \ldots + n\binom{n}{n} = n2^{n-1}$

. $1^2\binom{n}{1} + 2^2\binom{n}{2} + 3^2\binom{n}{3} + \ldots + n^2\binom{n}{n} = (n+n^2)2^{n-2}$

. Vandermonde's Identify: $\sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k} = \binom{m+n}{r}$

. Hockey-Stick Identify: $n, r \in N, n > r, \sum_{i=r}^{n}\binom{i}{r} = \binom{n+1}{r+1}$

. $\sum_{i=0}^{k}\binom{k}{i}^2 = \binom{2k}{k}$

. $\sum_{k=0}^{n}\binom{n}{k}\binom{n}{n-k} = \binom{2n}{n}$

. $\sum_{k=q}^{n}\binom{n}{k}\binom{k}{q} = 2^{n-q}\binom{n}{q}$

. $\sum_{i=0}^{n}k^i\binom{n}{i} = (k+1)^n$

. $\sum_{i=0}^{n}\binom{2n}{i} = 2^{2n-1} + \frac{1}{2}\binom{2n}{n}$

. $\sum_{i=1}^{n}\binom{n}{i}\binom{n-1}{i-1} = \binom{2n-1}{n-1}$

. $\gcd(a, \operatorname{lcm}(b, c)) = \operatorname{lcm}(\gcd(a, b), \gcd(a, c))$.

. $\operatorname{lcm}(a, \gcd(b, c)) = \gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(a, c))$.

. For non-negative integers a and b, where a and b are not both zero,

$\gcd(n^a - 1, n^b - 1) = n^{\gcd(a,b)} - 1$

. $\gcd(a, b) = \sum_{k|a \text{ and } k|b}\phi(k)$

. $\sum_{i=1}^{n}[\gcd(i, n) = k] = \phi\left(\frac{n}{k}\right)$

. $\sum_{k=1}^{n}\gcd(k, n) = \sum_{d|n} d\cdot\phi\left(\frac{n}{d}\right)$

. $\sum_{k=1}^{n}x^{\gcd(k,n)} = \sum_{d|n}x^d\cdot\phi\left(\frac{n}{d}\right)$

. $\sum_{k=1}^{n}\frac{1}{\gcd(k, n)} = \sum_{d|n}\frac{1}{d}\cdot\phi\left(\frac{n}{d}\right) = \frac{1}{n}\sum_{d|n}d\cdot\phi(d)$

. $\sum_{k=1}^{n}\frac{k}{\gcd(k, n)} = \frac{n}{2}\cdot\sum_{d|n}\frac{1}{d}\cdot\phi\left(\frac{n}{d}\right) = \frac{n}{2}\cdot\frac{1}{n}\cdot\sum_{d|n}d\cdot\phi(d)$

. $\sum_{k=1}^{n}\frac{n}{\gcd(k, n)} = 2 * \sum_{k=1}^{n}\frac{k}{\gcd(k, n)} - 1$, for $n > 1$

. $\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i, j) = 1] = \sum_{d=1}^{n}\mu(d)\lfloor\frac{n}{d}\rfloor^2$

. $\sum_{i=1}^{n}\sum_{j=1}^{n}\gcd(i, j) = \sum_{d=1}^{n}\phi(d)\lfloor\frac{n}{d}\rfloor^2$

. $\sum_{i=1}^{n}\sum_{j=1}^{n}i\cdot j[\gcd(i, j) = 1] = \sum_{i=1}^{n}\phi(i)i^2$

. $F(n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{lcm}(i, j) = \sum_{l=1}^{n}\left(\frac{\left(1 + \lfloor\frac{n}{l}\rfloor\right)\left(\lfloor\frac{n}{l}\rfloor\right)}{2}\right)^2\sum_{d|l}\mu(d)ld$

. $\gcd(\operatorname{lcm}(a, b), \operatorname{lcm}(b, c), \operatorname{lcm}(a, c)) = \operatorname{lcm}(\gcd(a, b), \gcd(b, c), \gcd(a, c))$

. $\gcd(A_L, A_{L+1}, \ldots, A_R) = \gcd(A_L, A_{L+1} - A_L, \ldots, A_R - A_{R-1})$.'

. Given n, If $SUM = LCM(1, n) + LCM(2, n) + \ldots + LCM(n, n)$

Pythagorean triplets

Let the given number be $n > 2$.

For n even, $n^2 + ((n/2)^2 - 1)^2 = ((n/2)^2 + 1)^2$.

For n odd, $n^2 + ((n^2 - 1)/2)^2 = ((n^2 + 1)/2)^2$.

```
vector<int> Z_algo(string s) {
    int i,l=0,r=0,n=s.size();
    vector<int> z(n);
    for(i=1; i<n; i++) {
        if(i<=r)
            z[i]=min(r-i+1,z[i-l]);
        while(i+z[i]<n&&s[z[i]]==s[i+z[i]])
            z[i]++;
        if(i+z[i]-1>r)
            l=i,r=i+z[i]-1;
    }
    z[0]=n;
    return z;
}
-----------------------------------------------
vector<int> fail(N);
void failure(string s){
    int i=1,j=0;
    while(i<s.size()){
        while(j>0&&s[i]!=s[j])
            j=fail[j-1];
        if(s[i]==s[j])j++;
        fail[i++]=j;
    }
}
int kmp(string s,string t){
    failure(t);
    int i=0,j=0;
    while(i<s.size()){
        while(j>0&&s[i]!=t[j])
            j=fail[j-1];
        if(s[i]==t[j])
            j++;
        if(j==t.size())
            return 1;
        i++;
    }
    return 0;
}
-----------------------------------------------
string min_cyclic_string(string s) {
    s+=s;
    int n=s.size(),i=0,ans=0;
    while(i<n/2) {
        ans=i;
        int j=i+1, k=i;
        while(j<n && s[k]<=s[j]) {
            if(s[k]<s[j])
                k=i;
            else
                k++;
            j++;
        }
        while(i<=k)
            i+=j-k;
    }
    return s.substr(ans,n/2);
}
```

```
struct Manacher {
    string s;
    int n;
    vector<int>d1,d2;
    Manacher() {}
    Manacher(string _) {
        s=_;
        n=s.size();
        d1=d2=vector<int>(n);
        Build();
    }
    void Build() {
        for(int i=0, l=0, r=-1; i<n; i++) {
            int k=(i>r)?1:min(d1[l+r-i],r-i);
            while(0 <= i-k && i+k<n && s[i-
k] == s[i+k])
                k++;
            d1[i]=k--;
            if(i+k>r)
                l=i-k,r=i+k;
        }
        for(int i=0, l=0, r=-1; i<n; i++) {
            int k=(i>r)?0:min(d2[l+r-i+1],r-
i+1);
            while(0 <= i-k-1 && i+k<n &&
s[i-k-1] == s[i+k])
                k++;
            d2[i]=k--;
            if(i+k > r)
                l=i-k-1,r=i+k;
        }
    }
};
-----------------------------------------------
struct SA {   /// suffix array
    int n;
    vector<int>p,c,ocur,lcp;
    string s;
    SA() {}
    SA(string _) {
        s=_+"$";
        n=s.size();
        p=c=lcp=vector<int>(n);
        ocur=vector<int>(max(n,256));
        Build();
        Build_lcp();
}
    void Build() {
        for(int i=0; i<n; i++)
            ocur[s[i]]++;
        for(int i=1; i<256; i++)
            ocur[i]+=ocur[i-1];
        for(int i=0; i<n; i++)
            p[--ocur[s[i]]]=i;
        c[p[0]]=0;
        int cls=1;
```

```
        for(int i=1; i<n; i++) {
            if(s[p[i]]!=s[p[i-1]])
                cls++;
            c[p[i]]=cls-1;
        }
        vector<int> pn(n),cn(n);
        for(int h=0; (1<<h)<n; h++) {
            for(int i=0; i<n; i++) {
                pn[i]=p[i]-(1<<h);
                if(pn[i]<0)
                    pn[i]+=n;
            }
            fill(ocur.begin(),ocur.begin()+cl
s,0);
            for(int i=0; i<n; i++)
                ocur[c[pn[i]]]++;
            for(int i=1; i<cls; i++)
                ocur[i]+=ocur[i-1];
            for(int i=n-1; i>=0; i--)
                p[--ocur[c[pn[i]]]]=pn[i];
            cn[pn[0]]=0;
            cls=1;
            for(int i=1; i<n; i++) {
                pair<int,int>
cur={c[p[i]],c[(p[i]+(1<<h))%n]};
                pair<int,int> prev={c[p[i-
1]],c[(p[i-1]+(1<<h))%n]};
                if(cur!=prev)
                    cls++;
                cn[p[i]]=cls-1;
            }
            c.swap(cn);
        }
    }
    void Build_lcp() {
        vector<int> rnk(n,0);
        for(int i=0;i<n;i++)
            rnk[p[i]]=i;
        int k=0;
        for(int i=0;i<n;i++) {
            if(rnk[i]==n-1)
            {
                k=0;
                continue;
            }
            int j=p[rnk[i]+1];
            while(i+k<n&&j+k<n&&s[i+k]==
s[j+k])
                k++;
            lcp[rnk[i]]=k;
            if(k)
                k--;
        }
    }
};
```

```
const int mod1 = 127657753,mod2 =
987654319;
const int b1 = 141, b2 = 277;
pair<int,int>pw[N],inv[N];
void precalc() {    /// call this from main
function
   pw[0] = {1,1};
   for(int i=1; i<N; i++) {
      pw[i].F = 1LL*pw[i-1].F*b1%mod1;
      pw[i].S = 1LL*pw[i-1].S*b2%mod2;
   }
   inv[N-1].F=powmod(pw[N-
1].F,mod1-2,mod1);
   inv[N-1].S=powmod(pw[N-
1].S,mod2-2,mod2);
   for(int i=N-2; i>=0; i--) {
      inv[i].F=1LL*inv[i+1].F*b1%mod1;
      inv[i].S=1LL*inv[i+1].S*b2%mod2;
   }
}
struct HASH    /// 1-indexed
{
   int n;
   vector<pair<int,int>>h,rh;
   HASH() {}
   HASH(string s) {
      n=s.size();
      h.resize(n+1);
      rh.resize(n+1);
      for(int i=1; i<=n; i++) {
         h[i].F=(1LL*b1*h[i-1].F+(s[i-1]-
'a'+1))%mod1;
         h[i].S=(1LL*b2*h[i-1].S+(s[i-1]-
'a'+1))%mod2;

         rh[i].F=(1LL*rh[i-1].F+1LL*pw[i-
1].F*(s[i-1]-'a'+1))%mod1;
         rh[i].S=(1LL*rh[i-1].S+1LL*pw[i-
1].S*(s[i-1]-'a'+1))%mod2;
      }
   }
   pair<int,int> get_hash(int l,int r) {
      int val1=(h[r].F-(1LL*h[l-1].F*pw[r-
l+1].F)%mod1+mod1)%mod1;
      int val2=(h[r].S-(1LL*h[l-1].S*pw[r-
l+1].S)%mod2+mod2)%mod2;
      return {val1,val2};
   }
   pair<int,int> get_revhash(int l,int r) {
      int val1=1LL*((rh[r].F-rh[l-
1].F+mod1)%mod1)*inv[l-1].F%mod1;
      int val2=1LL*((rh[r].S-rh[l-
1].S+mod2)%mod2)*inv[l-1].S%mod2;
      return {val1,val2};
   }
};
/// append -> H*prime + c
/// prepend -> H + (prime^n)*c
```

```
struct eerTree {
   struct node {
      int nxt[26],len,link;
      int tot; /// total number of
palindrome ends here
      int occ; /// frequency of current
palindrome
   };
   string s;
   vector<node>t;
   int idx,tt,n; /// tt -> last processed
node
   eerTree(string s) {
      this->s=s;
      n=s.size();
      t=vector<node>(n+3);
      t[1].len=-1,t[1].link=1;
      t[2].len=0,t[2].link=1;
      tt=idx=2;
   }
   int get_link(int x,int i) {
      while(s[i-t[x].len-1]!=s[i])
         x=t[x].link;
      return x;
   }
   bool extend(int i) {
      tt=get_link(tt,i);
      int cur=t[tt].link,c=s[i]-'a';
      cur=get_link(cur,i);
      if(!t[tt].nxt[c]) { /// new
palindrome
         t[tt].nxt[c]=++idx;
         t[idx].len=t[tt].len+2;
         t[idx].link=t[idx].len==1?2:t[cur
].nxt[c];
         t[idx].tot=1+t[t[idx].link].tot;
         tt=t[tt].nxt[c];
         t[tt].occ=1;
         return true;
      }
      tt=t[tt].nxt[c];
      t[tt].occ++;
      return false;
   }
};
```

```
struct Trie {
   struct node {
      int occ;
      node* next[26];
      node() {
         occ=0;
         for(int i=0; i<26; i++)
            next[i]=NULL;
      }
   }*root;
   Trie() {
      root=new node();
   }
   void Insert(string s) {
      node* cur=root;
      for(int i=0; i<s.size(); i++) {
         int x=s[i]-'a';
         if(cur->next[x]==NULL)
            cur->next[x]=new node();
         cur=cur->next[x];
      }
      cur->occ++;
   }
   int Query(string s) {
      node* cur=root;
      int ans=0;
      for(int i=0; i<s.size(); i++) {
         int x=s[i]-'a';
         if(cur->next[x]==NULL)
            return 0;
         cur=cur->next[x];
      }
      return cur->occ;
   }
   void Delete(node* cur) {
      for(int i=0; i<26; i++)
         if(cur->next[i]!=NULL)
            Delete(cur->next[i]);
      delete cur;
   }
};
```

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100001;
bitset<N>bs[26],oc;
int main() {
   string s,t;   cin>>s;
   for(int i=0; i<s.size(); i++)
      bs[s[i]-'a'][i]=1;
   int q,k;    cin>>q;
   while(q--) {
      cin>>k>>t;
      oc.set();
      for(int i=0; i<t.size(); i++)
         oc&=(bs[t[i]-'a']>>i);
      if(oc.count()<k) {
         cout<<-1<<'\n';
         continue;
      }
      vector<int>v;
      int pos=oc._Find_first();
      int ans=INT_MAX,m=t.size();
      while(pos<N)  {
         v.push_back(pos);
         pos=oc._Find_next(pos);
      }
      for(int i=k-1; i<v.size(); i++)
         ans=min(ans,v[i]-v[i-k+1]+m);
      cout<<ans<<'\n';
   }
}
```

```
struct suffix_automata {
  struct node {
    int len,link;
    int fp; /// first position
    long long dp,occ;
    map<char,int>nxt;
  };
  string s;
  int idx,tt,n;
  vector<node>t;
  suffix_automata(string s) {
    this->s=s;
    n=s.size();
    t=vector<node>(n<<1);
    idx=tt=t[0].len=0;
    t[0].link=-1;
    ++idx;
  }
  void extend(char c) {
    int cur=idx++;
    t[cur].len=t[tt].len+1;
    t[cur].fp=t[cur].len-1;
    t[cur].occ=1;
    int p;
    for(p=tt; ~p&&!t[p].nxt.count(c);
p=t[p].link)
      t[p].nxt[c]=cur;
    if(p==-1)
      t[cur].link=0;
    else {
      int q=t[p].nxt[c];
      if(t[p].len+1==t[q].len)
        t[cur].link=q;
      else {
        int clone=idx++;
        t[clone].len=t[p].len+1;
        t[clone].link=t[q].link;
        t[clone].nxt=t[q].nxt;
        t[clone].fp=t[q].fp;
        t[clone].occ=0;
        while(~p&&t[p].nxt[c]==q) {
          t[p].nxt[c]=clone;
          p=t[p].link;
        }
        t[q].link=t[cur].link=clone;
      }
    }
    tt=cur;
  }
  int first_occurence(string s) {
    int v=0;
    for(int i=0; i<s.size(); i++) {
      if(t[v].nxt.count(s[i]))
        v=t[v].nxt[s[i]];
      else
        return -1;
    }
    return t[v].fp-s.size()+1;// 0 idx
  }
  void update_occurence() { /// need
for total number of substring (not
distinct)
    vector<vector<int>>g(idx+1);
    for(int i=0; i<=idx; i++)
      g[t[i].len].push_back(i);
    for(int i=idx; i>=0; i--)
      for(auto u:g[i])
        if(~t[u].link)
          t[t[u].link].occ+=t[u].occ;
  }
  void no_of_sub(int x) { /// total
substrings
    ///t[x].dp=1; /// distinct
    t[x].dp=t[x].occ;
    for(auto [ch,id]:t[x].nxt) {
      if(!t[id].dp)
        no_of_sub(id);
      t[x].dp+=t[id].dp;
    }
  }
  string kth_substr(long long k) {
    update_occurence(); /// for total
substrings not distinct
    no_of_sub(0);
    string res;
    int v=0;
    while(k>0) {
      for(auto [ch,id]:t[v].nxt)
        if(k>=t[id].dp)
          k-=t[id].dp;
        else {
          res.pb(ch);
          v=id;
          ///k--; ///distinct
          k-=t[v].occ; ///
          break;
        }
    }
    return res;
  }
  string max_substr_with_(string s) { ///
Longest Common String
    int v=0,l=0,best=0,bestpos=0;
    for(int i=0; i<s.size(); i++) {
      while(v && !t[v].nxt.count(s[i])) {
        v=t[v].link;
        l=t[v].len;
      }
      if(t[v].nxt.count(s[i])) {
        v=t[v].nxt[s[i]];
        l++;
      }
      if(l>best)
        best=l,bestpos=i;
    }
    return s.substr(bestpos-
best+1,best);
  }
  bool check_substr(string s) {
    int v=0;
    for(int i=0; i<s.size(); i++) {
      if(t[v].nxt.count(s[i]))
        v=t[v].nxt[s[i]];
      else return false;
    }
    return true;
  }
  void distinct_sub_by_length() {
    vector<int>dist(n<<1,-1);
    vector<long long>ans(n+2);
    queue<int>q;q.push(0);
    dist[0]=0;
    while(!q.empty()) {
      auto x=q.front(); q.pop();
      ans[dist[x]]++;
      ans[t[x].len+1]--;
      for(auto [ch,id]:t[x].nxt)
        if(dist[id]==-1) {
          dist[id]=dist[x]+1;
          q.push(id);
        }
    }
    for(int i=1; i<=n; i++) {
      ans[i]+=ans[i-1];
      cout<<ans[i]<<" \n"[i==n];
    }
  }
};
----------------------------------------------
template<class T>
struct MonotonicQueue {
  struct data {
    int idx; T val; data() {}
    data(int idx,T val) {
      this->idx=idx;this->val=val;
    }
  };
  deque<data>dq;
MonotonicQueue(){}
  void Add(int idx,T val) {

while(!dq.empty()&&dq.back().val>=
val)
      dq.pop_back();
    dq.push_back(data(idx,val));
  }
  void Remove(int idx) {
  while(!dq.empty()&&dq.front().idx
<=idx)
      dq.pop_front();
  }
  T Query() {
    if(!dq.empty())
      return dq.front().val;
    else return INT_MAX;
  }
};
```

```
template<class T>
struct BIT { ///1-indexed;
   int n;
   vector<T> t;
   BIT() {}
   BIT(int _n) {
      n=_n;
      t.assign(_n+1,0);
   }
   void Update(int idx,T val) {
      while(idx<=n) {
         t[idx]+=val;
         idx+=(idx&-idx);
      }
   }
   void Update(int l,int r,T val) {
      Update(l,val);
      Update(r+1,-val);
   }
   T Query(int idx) {
      T s=0;
      while(idx>0) {
         s+=t[idx];
         idx-=(idx&-idx);
      }
      return s;
   }
   T Query(int l,int r){
      return Query(r)-Query(l-1);
   }
};
-----------------------------------------------
struct DSU{
   vector<int>p,sz;
   DSU(){}
   DSU(int n){
      p.assign(n+1,0);
      sz.assign(n+1,1);
      iota(p.begin(),p.end(),0);
   }
   int Find(int u){
      if(p[u]==u)
         return u;
      return p[u]=Find(p[u]);
   }
   bool Unite(int x,int y){
      x=Find(x);  y=Find(y);
      if(x!=y){
         if(sz[x]<sz[y])  swap(x,y);
         p[y]=x;
         sz[x]+=sz[y]; sz[y]=0;
         return true;
      }
      return false;
   }
   int Size(int u){
      return sz[Find(u)];
   }
};
```

```
const int BLOCK=3500; /// 4310 for
2e5
struct MOS {
   struct query {
      int l,r,t,idx;
      query() {}
      query(int  l,int r,int t,int idx){
         this->l=l;
         this->r=r;
         this->t=t;
         this->idx=idx;
      }
      bool operator<(const query
&ot)const {
         if(l/BLOCK==ot.l/BLOCK) {
            if(r/BLOCK==ot.r/BLOCK)
               return t<ot.t;
            else
               return
r/BLOCK<ot.r/BLOCK;
         }
         else
            return (l/BLOCK<ot.l/BLOCK);
      }
   };
   struct update {
      int idx,prv,nxt;
      update() {}
      update(int idx,int prv,int nxt) {
         this->idx=idx;
         this->prv=prv;
         this->nxt=nxt;
      }
   };
   int n;
   long long tot;
   vector<int>ara,last;
   vector<query>p;
   vector<update>up;
   unordered_map<int,int>occ;
   MOS() {}
   MOS(int n,vector<int>ara) {
      this->n=n;
      this->ara=ara;
      this->last=ara;
      tot=0;
   }
   void Add_query(int  l,int r){
      p.emplace_back(query(l,r,(int)up.
size(),(int)p.size()));
   }
   void Add_update(int  idx,int val){
      up.emplace_back(update(idx,last
[idx],val));
      last[idx]=val;
   }
```

```
   void Add(int x){
      if(x%3)return;
      occ[x]++;
      if(occ[x]==1&&x%3==0)
         tot+=x;
   }
   void Remove(int x){
      if(x%3)return;
      occ[x]--;
      if(occ[x]==0&&x%3==0)
         tot-=x;
   }
   void Apply(int  idx,int val,int l,int r){
      if(idx>=l&&idx<=r){
         Remove(ara[idx]);
         ara[idx]=val;
         Add(ara[idx]);
      }
      else
         ara[idx]=val;
   }
   void Solve(){
      sort(p.begin(),p.end());
      vector<long long>ans(p.size());
      int L=0,R=-1,T=0;
      for(auto i:p){
         while(T<i.t)
      Apply(up[T].idx,up[T].nxt,L,R),T++;
         while(T>i.t)
          --
T,Apply(up[T].idx,up[T].prv,L,R);
         while(R<i.r)Add(ara[++R]);
         while(R>i.r)Remove(ara[R--]);
         while(L<i.l)Remove(ara[L++]);
         while(L>i.l)Add(ara[--L]);
         ans[i.idx]=tot;
      }
      for(auto i:ans)
         cout<<i<<'\n';
   }
};
-----------------------------------------------
template<class T>
struct SD   /// 0-indexed
{
   const int BLOCK = 550;  /// change
required
   T n,sz;  /// sz -> no of blocks
   vector<T>ara,sum;
   vector<vector<T>> blocks;
   SD() {}
   SD(int _n,vector<T>vec){
      n=_n;
      ara=vec;
      sz=(n+BLOCK+BLOCK-1)/BLOCK;
      sum=vector<T>(sz);
      blocks=vector<vector<T>>(sz);
      Build();
   }
```

```
  void Build() {
    for(int i=0; i<n; i++) {
      int cur=i/BLOCK;
      sum[cur]+=ara[i];
      blocks[cur].emplace_back(ara[i]);
    }
    for(int i=0; i<sz; i++)
      sort(blocks[i].begin(),blocks[i].end());
  }
  void Update(int pos,T val) {
    int cur=pos/BLOCK;
    sum[cur]+=(val-ara[pos]);
    int val_pos=lower_bound(blocks[cur].begin(),blocks[cur].end(),ara[pos])-blocks[cur].begin();
    ara[pos]=val;
    blocks[cur][val_pos]=val;
    sort(blocks[cur].begin(),blocks[cur].end());
  }
  T Query(int l,int r) {
    T t_sum=0;
    int L=l/BLOCK,R=r/BLOCK;
    if(L==R) {
      for(int i=l; i<=r; i++)
        t_sum+=ara[i];
    }
    else {
      for(int i=l,till=(L+1)*BLOCK-1; i<=till; i++)
        t_sum+=ara[i];
      for(int i=L+1; i<=R-1; i++)
        t_sum+=sum[i];
      for(int i=R*BLOCK; i<=r; i++)
        t_sum+=ara[i];
    }
    return t_sum;
  }
};
------------------------------------------------
#include <bits/stdc++.h>
using namespace std;
const int N=3e5,MAX=1e6;
int a[N];
struct wavelet_tree {
#define vi vector<int>
#define pb push_back
  int lo,hi;
  wavelet_tree *l, *r;
  vi b,c; /// c holds the prefix sum of elements
  /// nos are in range [x,y]
  /// array indices are [from,to)
```

```
  wavelet_tree(int *from,int *to,int x,int y) {
    lo=x,hi=y;
    if(from>=to)
      return;
    if(hi==lo) {
      b.reserve(to-from+1);
      b.pb(0);
      c.reserve(to-from+1);
      c.pb(0);
      for(auto it=from;it!=to;it++){
        b.pb(b.back()+1);
        c.pb(c.back()+*it);
      }
      return ;
    }
    int mid=(lo+hi)/2;
    auto f=[mid](int x) {
      return x<=mid;
    };
    b.reserve(to-from+1);
    b.pb(0);
    c.reserve(to-from+1);
    c.pb(0);
    for(auto it=from; it!=to; it++) {
      b.pb(b.back()+f(*it));
      c.pb(c.back()+*it);
    }
    //see how lambda function is used here
    auto pivot = stable_partition(from,to,f);
    l = new wavelet_tree(from,pivot,lo,mid);
    r = new wavelet_tree(pivot,to,mid+1,hi);
  }
  /// swap a[i] with a[i+1],if a[i]!=a[i+1] call swapadjacent(i)
  void swapadjacent(int i) {
    if(lo==hi)
      return ;
    b[i]=b[i-1]+b[i+1]-b[i];
    c[i] =c[i-1]+c[i+1]-c[i];
    if(b[i+1]-b[i]==b[i]-b[i-1]) {
      if(b[i]-b[i-1])
        return this->l->swapadjacent(b[i]);
      else
        return this->r->swapadjacent(i-b[i]);
    }
    else
      return ;
  }
```

```
  /// kth smallest element in [l,r]
  int kth(int l,int r,int k) {
    if(l>r)
      return 0;
    if(lo==hi)
      return lo;
    int inLeft=b[r]-b[l-1];
    int lb=b[l-1]; /// amt of nos in first (l-1) nos that go in left
    int rb=b[r]; /// amt of nos in first (r) nos that go in left
    if(k<=inLeft)
      return this->l->kth(lb+1,rb,k);
    return this->r->kth(l-lb,r-rb,k-inLeft);
  }
  /// count of nos in [l,r] Less than or equal to k
  int LTE(int l,int r,int k) {
    if(l>r||k<lo)
      return 0;
    if(hi<=k)
      return r-l+1;
    int lb=b[l-1],rb=b[r];
    return this->l->LTE(lb+1,rb,k)+this->r->LTE(l-lb,r-rb,k);
  }
  /// count of nos in [l,r] equal to k
  int Count(int l,int r,int k) {
    if(l>r||k<lo||k>hi)
      return 0;
    if(lo==hi)
      return r-l+1;
    int lb=b[l-1],rb=b[r],mid=(lo+hi)/2;
    if(k<=mid)
      return this->l->Count(lb+1,rb,k);
    return this->r->Count(l-lb,r-rb,k);
  }
  /// sum of nos in [l,r] less than or equal to k
  int sumk(int l,int r,int k) {
    if(l>r||k<lo)
      return 0;
    if(hi<=k)
      return c[r]-c[l-1];
    int lb=b[l-1],rb=b[r];
    return this->l->sumk(lb+1,rb,k)+this->r->sumk(l-lb,r-rb,k);
  }
  ~wavelet_tree(){
    delete l;
    delete r;
  }
};
```

```
int main()
{
   int i,n,k,j,q,l,r,x;
   cin>>n;
   for(i=1; i<=n; i++)cin>>a[i];
   wavelet_tree T(a+1,a+n+1,1,MAX);
   cin>>q;
   while(q--)
   {
      cin>>x>>l>>r>>k;
      if(x==0) /// kth smallest
         cout<<T.kth(l,r,k)<<endl;
      if(x==1) /// less than or equal to K
         cout<<T.LTE(l,r,k)<<endl;
      if(x==2) /// count occurence of K
in [l,r]
         cout<<T.Count(l,r,k)<<endl;
      if(x==3) /// sum of elements less
than or equal to K in [l,r]
         cout<<T.sumk(l,r,k)<<endl;
   }
}
-----------------------------------------------
template<class T>
struct ST {
   int n,m; T sum,mn; vector<T>lg;
   vector<vector<T>>st;
   ST() {}
   ST(vector<int> ara) {
      n=ara.size();
      m=log2(n)+1;
      lg.resize(n+1);
      st=vector<vector<T>>(n,vector<T
>(m+1));
      lg[1]=0;
      for(int i=2;i<=n;i++)
         lg[i]=lg[i/2]+1;
      for(int i=0; i<n; i++)
         st[i][0]=ara[i];
      for(int j=1; j<=m; j++)
         for(int i=0; i+(1<<j)<=n; i++)
            st[i][j]=f(st[i][j-1],st[i+(1<<(j-
1))][j-1]);
   }
   T Range_Sum(int L, int R) {
      sum=0;
      for(int j=m; j>=0; j--)
         if((1<<j)<=R-L+1) {
            sum+=st[L][j];
            L+=(1<<j);
         }
      return sum;
   }
   T RMQ(int L, int R) {
      int j=lg[R-L+1];
      mn=min(st[L][j],st[R-(1<<j)+1][j]);
      return mn;
   }
};
```

```
#include<bits/stdc++.h>
using namespace std;
const int N = 200010;
int node_cnt,n,m;
int
sum[N<<5],rt[N],lc[N<<5],rc[N<<5];
int a[N],b[N],p;
void build(int &t,int l,int r) {
   t=++node_cnt;
   if(l==r) return;
   int mid=(l+r)>>1;
   build(lc[t],l,mid);
   build(rc[t],mid+1,r);
}
int modify(int o,int l,int r) {
   int oo=++node_cnt;
   lc[oo]=lc[o];rc[oo]=rc[o];
   sum[oo]=sum[o]+1;
   if(l==r) return oo;
   int mid=(l+r)>>1;
   if(p<=mid)
      lc[oo]=modify(lc[oo],l,mid);
   else
      rc[oo]=modify(rc[oo],mid+1,r);
   return oo;
}
int query(int u,int v,int l,int r,int k)
{
   int ans,mid=((l+r)>>1),x=sum[lc[v]]-
sum[lc[u]];
   if(l==r) return l;
   if(x>=k)
      ans=query(lc[u],lc[v],l,mid,k);
   else
      ans=query(rc[u],rc[v],mid+1,r,k-
x);
   return ans;
}
int main() {
   int l,r,k,q,ans; cin>>n>>m;
   for(int i=1; i<=n; i++) {
      cin>>a[i];
      b[i]=a[i];
   }
   sort(b+1,b+n+1);
   q=unique(b+1,b+n+1)-b-1;
   build(rt[0],1,q);
   for(int i=1; i<=n; i++) {
      p=lower_bound(b+1,b+q+1,a[i])-
b;
      rt[i]=modify(rt[i-1],1,q);
   }
   while(m--) {
      cin>>l>>r>>k;
      ans=query(rt[l-1],rt[r],1,q,k);
      cout<<b[ans]<<'\n';
   }
}
```

```
vector<int>pf(N);
void smallestpf(){
   for(int i=2; i<N; i+=2)
      pf[i]=2,pf[i-1]=i-1;
   for(int i=3; i*i<N; i+=2)
      if(pf[i]==i)
         for(int j=i*i; j<N; j+=2*i)
            if(pf[j]==j)  pf[j]=i;
}
-----------------------------------------------
vector<bool>mark(N);
vector<int>p;
void seive(){
   mark[0]=mark[1]=true;
   for(int i=4; i<N; i+=2)
      mark[i]=true;
   for(int i=3; i*i<N; i+=2)
      if(!mark[i])
         for(int j=i*i;j<N;j+=2*i)
            mark[j]=true;
   p.push_back(2);
   for(int i=3; i<N; i+=2)
      if(!mark[i])
         p.push_back(i);
}
-----------------------------------------------
vector<long
long>fact(N),inv(N),invfact(N);
void pre() {
   inv[0]=inv[1]=fact[0]=invfact[0]=1;
   ll mod=MOD;
   for(ll i=2; i<N; i++)
      inv[i]=mod-
mod/i*inv[mod%i]%mod;
   for(ll i=1; i<N; i++) {
      fact[i]=fact[i-1]*i%mod;
      invfact[i]=invfact[i-1]*inv[i]%mod;
   }
}
long long ncr(long long n,long long r){
   if(r>n) return 0;
   ll tmp=invfact[n-r]*invfact[r]%MOD;
   return (fact[n]*tmp)%MOD;
}
long long Lucus(long long n,long long
r){
   if(n==0) return 1LL;
   return
(1LL*ncr(n%MOD,r%MOD)*Lucus(n/M
OD,r/MOD))%MOD;
}
-----------------------------------------------
vector<int>mob(N);
void mobius() {
   mob[1]=1;
   for(int i=1;i<N;i++)
      for(int j=i+i;j<N;j+=i)
         mob[j]-=mob[i];
}
```

```
using u64 = uint64_t;
using u128 = __uint128_t;
u64 binpower(u64 base, u64 e, u64
mod){
   u64 result = 1;
   base %= mod;
   while(e) {
      if(e&1)
         result =
(u128)result*base%mod;
      base = (u128)base*base%mod;
      e >>= 1;
   }
   return result;
}
bool check_composite(u64 n, u64 a,
u64 d, int s) {
   u64 x = binpower(a,d,n);
   if(x == 1 || x == n-1)
      return false;
   for(int r = 1; r < s; r++) {
      x = (u128)x*x%n;
      if(x == n-1)
         return false;
   }
   return true;
}
bool MillerRabin(u64 n) {
   if(n<2)
      return false;
   int r = 0;
   u64 d = n-1;
   while((d&1)==0) {
      d>>=1;
      r++;
   }
   vector<int>ara={2,3,5,7,11,13,17,1
9,23,29,31,37};
   for(int a:ara) {
      if(n==a)
         return true;
      if(check_composite(n,a,d,r))
         return false;
   }
   return true;
}
-----------------------------------------------
struct GCD{
   ll x,y,gcd;
};
GCD ex_euclid(ll a,ll b){
   if(b==0)
      return {1,0,a};
   GCD tmp=ex_euclid(b,a%b);
   return {tmp.y,tmp.x-
(a/b)*tmp.y,tmp.gcd};
}
```

```
struct matrix {
   int n;
   vector<vector<int>>mat;
   matrix() {}
   matrix(int n) {
      this->n=n;
      mat=vector<vector<int>>(n,vecto
r<int>(n));
   }
   void make_identity(){
      for(int i=0; i<n; i++) mat[i][i]=1;
   }
   matrix operator +(const matrix
&ot)const{
      matrix res(n);
      for(int i=0; i<n; i++)
         for(int j=0; j<n; j++)
            res.mat[i][j]=(mat[i][j]+ot.ma
t[i][j])%MOD;
      return res;
   }
   matrix operator *(const matrix
&ot)const{
      matrix res(n);
      for(int i=0; i<n; i++)
         for(int j=0; j<n; j++) {
            int s=0;
            for(int k=0; k<n; k++)
               s=(s+1LL*mat[i][k]*ot.mat[
k][j]%MOD)%MOD;
            res.mat[i][j]=s;
         }
      return res;
   }
};
auto multiply(auto X,auto Y) {
   int r1=X.size(),c1=X[0].size();
   int r2=Y.size(),c2=Y[0].size();
   assert(c1==r2);
   vector<vector<int>>
ans(r1,vector<int>(c2));
   for(int i=0; i<r1; i++)
      for(int j=0; j<c2; j++) {
         int res=0;
         for(int k=0; k<c1; k++)
            res=(res+1LL*X[i][k]*Y[k][i]%
MOD)%MOD;
         ans[i][j]=res;
      }
   return ans;
}
matrix binpow(matrix x,int p) {
   matrix res(x.n);res.make_identity();
   while(p){
      if(p&1) res=res*x;
      x=x*x; p/=2;
   }
   return res;
}
```

```
vector<int>phi(N);
void euler_phi(){
   phi[0]=0;
   phi[1]=1;
   for(int i=2;i<N;i++)
      phi[i]=i;
   for(int i=2;i<N;i++)
      if(phi[i]==i)
         for(int j=i; j<N;j+=i)
            phi[j]-=(phi[j]/i);
}
int phi(int x) {
   int ans = x;
   for(int i=2; i*i <= x; i++) {
      if(x%i)
         continue;
      while(x%i==0)
         x/=i;
      ans=(ans/i)*(i-1);
   }
   if(x>1)
      ans=(ans/x)*(x-1);
   return ans;
}
-----------------------------------------------
vpll factor;
vll divi;
void find_divisor(ll pos,ll f){
   if(pos==factor.size()){
      divi.pb(f);
      return;
   }
   find_divisor(pos+1,f);
   for(int i=0;i<factor[pos].second;i++){
      f*=factor[pos].first;
      find_divisor(pos+1,f);
   }
}
-----------------------------------------------
long long ncr[N][N];
void pre() {
   ncr[0][0]=1;
   for(int i=1; i<N; i++) {
      ncr[i][0]=1;
      for(int j=1; j<N; j++) {
         ncr[i][j]=ncr[i-1][j]+ncr[i-1][j-1];
         if(ncr[i][j]>=MOD)
            ncr[i][j]-=MOD;
      }
   }
}
```

```
template<class T>
struct Segtree {
#define segtre int
m=(x+y)>>1,lu=2*u,ru=2*u+1
   struct data {
      T l,v;
      data() {
         this->l=0; this->v=0;
      }
      data(T l,T v) {
         this->l=l; this->v=v;
      }
   };
   vector<T>ara; vector<data>t;
   int n; Segtree() {}
   Segtree(int n) {
      this->n=n; t=vector<data>(4*n);
   }
   void Init(vector<T>vec) {
      this->ara=vec; Init(1,1,n);
   }
   void Update(int l,int r,T val) {
      Update(1,1,n,l,r,val);
   }
   data Query(int l,int r) {
      return Query(1,1,n,l,r);
   }
   void Updatelazy(int u,int x,int y){
      t[u].v+=(t[u].l*(y-x+1));
      if(x!=y) {
         t[2*u].l+=t[u].l;
         t[2*u+1].l+=t[u].l;
      } t[u].l=0;
   }
   data Combine(data a,data b) {
      data temp; temp.v=a.v+b.v;
      return temp;
   }
   void Init(int u,int x,int y) {
      if(x==y) {
         t[u].v=ara[x];
         return;
      } segtre;
      Init(lu,x,m); Init(ru,m+1,y);
      t[u]=Combine(t[lu],t[ru]);
   }
   void Update(int u,int x,int y,int b,int
e,T val) {
      if(t[u].l) Updatelazy(u,x,y);
      if(x>e||y<b) return;
      if(x>=b&&y<=e) {
         t[u].l+=val;
         Updatelazy(u,x,y);
         return;
      } segtre;
      Update(lu,x,m,b,e,val);
      Update(ru,m+1,y,b,e,val);
      t[u]=Combine(t[lu],t[ru]);
   }
   data Query(int u,int x,int y,int b,int
e) {
      if(t[u].l) Updatelazy(u,x,y);
      if(x>e||y<b) return data();
      if(x>=b&&y<=e) return t[u];
      segtre;
      data res1=Query(lu,x,m,b,e);
      data res2=Query(ru,m+1,y,b,e);
      return Combine(res1,res2);
   } };
template<class T>
struct HLD { /// 1 indexed;
  vector<int>depth,heavy,head,pos,sz
,ara,tin,tout;
   vector<vector<int>>par,g;
   int n,m,timer,cur_pos;
   Segtree<int>t; /// RMQ HLD() {}
   HLD(int n) {
      this->n=n; m=log2(n);
      cur_pos=timer=0;
     tin=tout=depth=head=pos=sz=ara
=vector<int>(n+1);
      heavy=vector<int>(n+1,-1);
      g=vector<vector<int>>(n+1);
      par=vector<vector<int>>(n+1,vec
tor<int>(m+1));
      t=Segtree<int>(n);
   }
   void add_edge(int u,int v) {
  g[u].push_back(v);g[v].push_back(u);
   }
   void dfs(int u,int p=0) {
  tin[u]=++timer;depth[u]=depth[p]+1;
      sz[u]=1; par[u][0]=p;
      int mx_sz=0;
      for(int i=1; i<=m; i++)
         par[u][i]=par[par[u][i-1]][i-1];
      for(auto v:g[u])
         if(v!=p) {
            dfs(v,u); sz[u]+=sz[v];
            if(sz[v]>mx_sz)
               mx_sz=sz[v],heavy[u]=v;
         }
      tout[u]=++timer;
   }
   void decompose(int u,int h) {
      head[u]=h; pos[u]=++cur_pos;
      if(~heavy[u])
         decompose(heavy[u],h);
      for(auto v:g[u])
         if(v!=par[u][0]&&v!=heavy[u])
            decompose(v,v);
   }
   void
initialize_weight(vector<int>vec){
      for(int i=1;i<=n;i++)
         ara[pos[i]]=vec[i];
      t.Init(ara);
   }
   int get_lca(int u,int v) {
      if(depth[u]>depth[v])swap(u,v);
      for(int i=m; i>=0; i--)
         if(depth[par[v][i]]>=depth[u])
            v=par[v][i];
      if(u==v)return v;
      for(int i=m; i>=0; i--)
         if(par[u][i]!=par[v][i])
            u=par[u][i],v=par[v][i];
      return par[u][0];
   } ///is u an ancestor of v?
   bool is_ancestor(int u,int v)
{      return (tin[u]<=tin[v] &&
tout[u]>=tout[v]);
   } ///k'th ancestor of u
   int kth_ancestor(int u,int k)
{      for(int i=m; i>=0; i--)
         if((1<<i)&k) u=par[u][i];
      return u;
   }
   int get_dist(int u,int v) {
      return depth[u]+depth[v]-
2*depth[get_lca(u,v)];
   }
   void update_up(int u,int v,T val) {
      while(head[u]!=head[v]) {
       t.Update(pos[head[v]],pos[v],val);
         v=par[head[v]][0];
      }
      t.Update(pos[u],pos[v],val);
   }
   void path_update(int u,int v,T val) {
      int lca=get_lca(u,v);
      update_up(lca,u,val);
      update_up(lca,v,val);
      update_up(lca,lca,-val);
   }
   T query_up(int u,int v) {
      T ans=0;/// careful
      while(head[u]!=head[v]) {
       T cur_ans=t.Query(pos[head[v]],p
os[v]).v;
         ans=ans+cur_ans;/// check +-*/
         v=par[head[v]][0];
      }
      T
cur_ans=t.Query(pos[u],pos[v]).v;
      ans=ans+cur_ans;///check +-*/
      return ans;
   }
   T path_query(int u,int v) {
      int lca=get_lca(u,v);
      return
query_up(lca,u)+query_up(lca,v)-
query_up(lca,lca);/// check operator
and handle overlap
   }
};
```

```
struct SCC
{
   vector<vector<int>>g,rg,comp;
   vector<bool>vis;
   vector<int>comp_no;
   stack<int>st;
   int n,sc;   /// sc -> no. of SCC
   SCC() {}
   SCC(int _n) {
      n=_n;
      sc=0;
      g=rg=vector<vector<int>>(n+1);
      comp_no=vector<int>(n+1);
      vis=vector<bool>(n+1);
      fill(vis.begin(),vis.end(),false);
   }
   void Add_edge(int u,int v)
   {
      g[u].push_back(v);
      rg[v].push_back(u);
   }
   void Forward(int u)
   {
      vis[u]=true;
      for(auto to:g[u])
         if(!vis[to])
            Forward(to);
      st.push(u);
   }
   void Back(int u)
   {
      vis[u]=true;
      comp_no[u]=sc;
      comp.back().push_back(u);
      for(auto v:rg[u])
         if(!vis[v])
            Back(v);
   }
   void Make_scc()
   {
      for(int i=1; i<=n; i++)
         if(!vis[i])
            Forward(i);
      fill(vis.begin(),vis.end(),false);
      while(!st.empty())
      {
         int u=st.top();
         st.pop();
         if(vis[u])
            continue;
         comp.push_back(vector<int>())
;
         Back(u);
         ++sc;
      }
   }
};
```

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100001;
vector<int> sz(N);
set<int>g[N];
char col[N];
void Get_sz(int u,int p=0) {
   sz[u]=1;
   for(auto v:g[u])
      if(v!=p){
         Get_sz(v,u);
         sz[u]+=sz[v];
      }
}
int Get(int u,int p,int n) {
   for(auto v:g[u])
      if(v!=p&&sz[v]>n)
         return Get(v,u,n);
   return u;
}
void Decompose(int u,int p,char rnk){
   Get_sz(u);
   int centroid=Get(u,0,sz[u]/2);
   col[centroid]=rnk;
   for(auto v:g[centroid]){
      g[v].erase(centroid);
      Decompose(v,centroid,rnk+1);
   }
   g[centroid].clear();
}
int main() {
   int n,i,x,y;
   cin>>n;
   for(i=1; i<n; i++){
      cin>>x>>y;
      g[x].insert(y);
      g[y].insert(x);
   }
   Decompose(1,0,'A');
   for(i=1; i<=n; i++)cout<<col[i]<<' ';
   cout<<'\n';
}
--------------------------------------------------
for(int k=1; k<=n; k++)
for(int i=1; i<=n; i++)
for(int j=1; j<=n; j++)
   floyd[i][j]=min(floyd[i][j],floyd[i][k]+
floyd[k][j]);
--------------------------------------------------
void find_cycle(int u,int anc=0) {
   p[u]=anc; col[u]=1;
   for(auto v:g[u])
      if(!col[v]) find_cycle(v,u);
      else if(col[v]==1&&anc!=v) {
         for(int i=u; i!=p[v]; i=p[i])
            cycle.pb(i);
      }
}
```

```
/// DSU on Tree
int ocur[N],sz[N],col[N];
bool big[N];
vvii g(N);
void Setsize(int v, int p) {
   sz[v]=1;
   for(auto u:g[v]) {
      if(u!=p) {
         Setsize(u,v); sz[v]+=sz[u];
      }
   }
}
void Add(int v, int p, int x) {
   ocur[col[v]]+=x;
   for(auto u:g[v])
      if(u!=p && !big[u])
         Add(u,v,x);
      }
void Dfs(int v, int p, bool keep) {
   int mx=-1,bigChild=-1;
   for(auto u:g[v])
      if(u!=p && sz[u]>mx)
         mx=sz[u],bigChild=u;
   for(auto u:g[v])
      if(u!=p && u!=bigChild)
         Dfs(u,v,0);  ///run a dfs on small
childs and clear them from cnt
   if(bigChild != -1)
      Dfs(bigChild,v,1),big[bigChild]=1;
/// bigChild marked as big and not
cleared from cnt
   Add(v,p,1);
   ///now cnt[c] is the number of
vertices in subtree of vertex v that has
color c. You can answer the queries
easily.
   if(bigChild != -1)
      big[bigChild]=0;
   if(keep == 0)
      Add(v,p,-1);
}
/// Duplicate
int col[N],ans[N];
vvii g(N);
set<int> dfs(int u,int p) {
   set<int>now;
   now.insert(col[u]);
   for(auto v:g[u])
      if(v^p) {
         auto child=dfs(v,u);
 if(child.size()>now.size())swap(child,n
ow);
         for(auto i:child)
            now.insert(i);
      }
   ans[u]=now.size();
   return now;
}
```

```
/// depth wise dp in tree
#include<bits/stdc++.h>
using namespace std;
const int N = 101;
int
n,X,dp[N][N+N][2],v[N],child[N],siblin
g[N];
vector<vector<int>>g(N);
void dfs(int u,int p=0) {
   int last=-1;
   for(auto i:g[u]) {
      if(i==p)
         continue;
      dfs(i,u);
      if(~last)
         sibling[last]=i;
      else
         child[u]=i;
      last=i;
   }
}
int Run(int u,int mov,int ok) {
   if(!mov||!u)
      return 0;
   int &ret=dp[u][mov][ok];
   if(~ret)return ret;
   ret=Run(sibling[u],mov,ok);
   if(!ok) {
      int hv=mov-1;
      for(int i=0; i<=hv; i++)
 ret=max(ret,v[u]+Run(child[u],i,0)+Ru
n(sibling[u],hv-i,1));
      hv=mov-2;
      for(int i=0; i<=hv; i++)
        ret=max(ret,v[u]+Run(child[u],i,
1)+Run(sibling[u],hv-i,0));
   }
   else {
      int hv=mov-2;
      for(int i=0; i<=hv; i++)
        ret=max(ret,v[u]+Run(child[u],i,
1)+Run(sibling[u],hv-i,1));
   }
   return ret;
}
int main() {
   cin>>n>>X;
   for(int i=1; i<=n; i++) cin>>v[i];
   for(int i=1; i<n; i++) {
      int x,y; cin>>x>>y;
      g[x].push_back(y);
      g[y].push_back(x);
   }
   dfs(1);
   memset(dp,-1,sizeof dp);
   cout<<Run(1,X+1,0)<<'\n';
}
```

```
/// LIDS in [L,R]
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 11;
string s,t;
int dp[N][N][2][2][2];
ll occ[N][N][2][2][N][2];
int Run(int pos,int prev,int is_small,int
is_large,int zero) {
   if(pos==t.size())
      return 0;
   int
&ret=dp[pos][prev+1][is_small][is_lar
ge][zero];
   if(~ret)return ret;
   ret=0;
   int st=(is_large)?0:s[pos]-'0';
   int en=(is_small)?9:t[pos]-'0';
   for(int i=st; i<=en; i++) {
      if(i>prev&&(zero||i))
         ret=max(ret,1+Run(pos+1,i,(is_
small||(i<en)),(is_large||(i>st)),(zero|
|i)));
      ret=max(ret,Run(pos+1,prev,(is_s
mall||(i<en)),(is_large||(i>st)),(zero||
i)));
   }
   return ret;
}
ll Go(int pos,int prev,int is_small,int
is_large,int len,int zero) {
   if(pos==t.size())
      return (!len);
   ll
&ret=occ[pos][prev+1][is_small][is_la
rge][len][zero];
   if(~ret)return ret;
   ret=0;
   int st=(is_large)?0:s[pos]-'0';
   int en=(is_small)?9:t[pos]-'0';
   for(int i=st; i<=en; i++) {
      if(i>prev&&(zero||i)&&len)
         ret+=Go(pos+1,i,(is_small||(i<e
n)),(is_large||(i>st)),len-1,(zero||i));
      ret+=Go(pos+1,prev,(is_small||(i
<en)),(is_large||(i>st)),len,(zero||i));
   }
   return ret;
}
int main() {
   cin>>s>>t;
   s=string(t.size()-s.size(),'0')+s;
   memset(dp,-1,sizeof dp);
   memset(occ,-1,sizeof occ);
   int len=Run(0,-1,0,0,0);
   int tot=Go(0,-1,0,0,len,0);
   cout<<len<<' '<<tot<<'\n';
}
```

```
/// Digit Less Number
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int dp[11][2];
string s;
int Run(int pos,int small){
   if(pos==s.size())
     return 1;
   int &ret=dp[pos][small];
   if(~ret)return ret;
   int till=small?s[pos]-'0':9;
   ret=0;
   if(!pos)
     for(int i=1; i<=till; i++)
        if(i!=7)
           ret+=Run(pos+1,(small&(i==til
l)));
   else
     for(int i=0; i<=till; i++)
        if(i!=7)
           ret+=Run(pos+1,(small&(i==til
l)));
   if(!pos)
     ret+=Run(pos+1,0);
   return ret;
}
int main()
{
   int x;
   cin>>x;
   s=to_string(x);
   memset(dp,-1,sizeof dp);
   int ans=(Run(0,1)-1);
   cout<<ans<<'\n';
}
-------------------------------------------------
mt19937 rng(chrono::steady_clock::no
w().time_since_epoch().count());  ///
mt19937_64 (long long)

auto my_rand(long long l,long long r)
{
   return
uniform_int_distribution<long
long>(l,r)(rng);
}
-------------------------------------------------
unordered_map<ull,ull>dp;
dp.reserve(1024);
dp.max_load_factor(0.25);
-------------------------------------------------
```

```
#define fast() ios_base::sync_with_stdio(false),cin.tie(NULL)
#define Unique(x) (x).erase(unique(all(x)),(x).end())
#define strtoint(a) atoi(a.c_str())
-----------------------------------------------------------------------------
///.........Bit_Manipulation...........///
#define leastonepos(mask) __builtin_ffs(mask)
#define leadingoff(mask) __builtin_clz(mask)
#define trailingoff(mask) __builtin_ctz(mask)
#define numofone(mask) __builtin_popcount(mask)
#define checkbit(mask,bit) (mask&(1LL<<bit))
#define setbit(mask,bit) (mask|(1LL<<bit))
#define resetbit(mask,bit) (mask&~(1LL<<bit))
#define changebit(mask,bit) (mask^(1LL<<bit))
-----------------------------------------------------------------------------
///................Graph's Move................
///const int dx[] = {+1,-1,+0,+0}; ///Rock's Move
///const int dy[] = {+0,+0,+1,-1}; ///Rock's Move
///const int dx[] = {+0,+0,+1,-1,-1,+1,-1,+1}; ///King's Move
///const int dy[] = {-1,+1,+0,+0,+1,+1,-1,-1}; ///king's Move
///const int dx[] = {-2,-2,-1,-1,+1,+1,+2,+2}; ///knight's Move
///const int dy[] = {-1,+1,-2,+2,-2,+2,-1,+1}; ///knight's Move
///*....................-_-........................*///
-----------------------------------------------------------------------------
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<typename T> using orderset =
tree<T,null_type,less<T>,rb_tree_tag,tree_order_statistics_
node_update>;
template<typename T> using ordermultiset = tree<T,
null_type, less_equal<T>, rb_tree_tag,
tree_order_statistics_node_update>;
///(X).order_of_key(value) /// return lower_bound(value)
///(*X).find_by_order(index) /// return value (0 index)
void myerase(orderset<int>&t, int v)
{
   int id = t.order_of_key(v);
   auto it = t.find_by_order(id);
   t.erase(it);
}
```

```
/// sin rule
     a/sin(A) = b/sin(B) = c/sin(C)

/// cosine rule
       a^2 = b^2 + c^2 - 2*b*c*cos(A)
       b^2 = a^2 + c^2 - 2*a*c*cos(B)
       c^2 = a^2 + b^2 - 2*a*b*cos(C)

/// Equilateral Triangle (সমবাহু ত্রিভুজ)
       area : sqrt(3)*a*a/4
       height : sqrt(3)*a/2

Cube:
    area -> 6*a*a
    volume -> a*a*a
Cylinder:
    area -> 2*pi*r*h+2*pi*r*r
    volume -> pi*r*r*h
Cone:
    area -> pi*r*l
    volume -> (pi*r*r*h)/3
Sphere:
    area -> 4*pi*r*r
    volume -> (4*pi*r*r*r)/3

Arc length -> s=r*theta (angle in radian)
Sector Area -> area=(theta*r*r)/2 (angle in radian)
Chord length:
    d=2*r*sin(theta/2) (angle in radian)
    d=2*sqrt(r*r-x*x) (x=Perpendicular Distance from the
Centre to Chord)
```

- $a + b = a \oplus b + 2(a \& b)$.
- $a + b = a \mid b + a \& b$
- $a \oplus b = a \mid b - a \& b$
- k_{th} bit is set in x iff $x \mod 2^{k-1} \geq 2^k$. It comes handy when you need to look at the bits of the numbers which are pair sums or subset sums etc.
- k_{th} bit is set in x iff $x \mod 2^{k-1} - x \mod 2^k \neq 0$ ($= 2^k$ to be exact). It comes handy when you need to look at the bits of the numbers which are pair sums or subset sums etc.
- $n \mod 2^i = n \& (2^i - 1)$
- $1 \oplus 2 \oplus 3 \oplus \cdots \oplus (4k - 1) = 0$ for any $k \geq 0$

Outside one another	$C_1C_2 > r_1 + r_2$
Touching externally	$C_1C_2 = r_1 + r_2$
Intersecting at 2 points	$\lvert r_1 + r_2 \rvert < C_1C_2 < r_1 + r_2$
Touching internally	$C_1C_2 = \lvert r_1 - r_2 \rvert$
One inside the other	$C_1C_2 < \lvert r_1 - r_2 \rvert$

Circumradius	$r = \frac{\text{abc}}{\sqrt{(a+b+c)(b+c-a)(c+a-b)(a+b-c)}}$ $r = \frac{\text{abc}}{4 \times AreaOfTriangle}$
Incircle Radius	$r = \frac{1}{2} \times ra + \frac{1}{2} \times rb + \frac{1}{2} rc = AreaOfTriangle$
Excircle Radius (If the circle is tangent to side **a** of the triangle)	$r = \text{IncircleRadius} \times \frac{\text{a + b + c}}{(b+c-a)}$ $r = 2 \times \frac{AreaOfTriangle}{b+c-a}$
Heron's Formula	$\sqrt{s(s-a)(s-b)(s-c)}$
Sine Rule	$\frac{a}{sinA} = \frac{b}{sinB} = \frac{c}{sinC} = 2R$
Cosine Rule	$a^2 = b^2 + c^2 - 2bcCosA$

. The function is multiplicative.
This means that if $\gcd(m,n) = 1, \phi(m \cdot n) = \phi(m) \cdot \phi(n)$.

. $\phi(n) = n \prod_{p|n} (1 - \frac{1}{p})$

. If p is prime and (k \geq 1), then, $\phi(p^k) = p^{k-1}(p-1) = p^k(1 - \frac{1}{p})$

. $J_k(n)$, the Jordan totient function, is the number of k-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with n. It is a generalization of Euler's totient, $\phi(n) = J_1(n)$.

$J_k(n) = n^k \prod_{p|n}(1 - \frac{1}{p^k})$

. $\sum_{d|n} J_k(d) = n^k$

. $\sum_{d|n} \phi(d) = n$

. When $x \geq \log_2 m$, then

$$n^x \mod m = n^{\phi(m)+x \mod \phi(m)} \mod m$$

. $\phi(n) = \sum_{d|n} \mu(d) \cdot \frac{n}{d} = n \sum_{d|n} \frac{\mu(d)}{d}$

. $\phi(n) = \sum_{d|n} d \cdot \mu(\frac{n}{d})$

$\sum_{1 \leq k \leq n, \gcd(k,n)=1} \gcd(k-1, n) = \varphi(n)d(n)$ where $d(n)$ is number of divisors. Same equation for $\gcd(a \cdot k - 1, n)$ where a and n are coprime.

. **Highest Power of 2 that divides ${}^{2n}C_n$:** Let x be the number of 1s in the binary representation. Then the number of odd terms will be 2^x .Let it form a sequence. The n-th value in the sequence (starting from n = 0) gives the highest power of 2 that divides ${}^{2n}C_n$.