**CodeBook**

**GUB\_ZeroFactorial**

**Noman Chowdhury Munna**

**Obaydullah khan**

**Mahedi hasan**

**Green University of Bangladesh**

**//I/O**

#ifndef ONLINE\_JUDGE freopen("input.txt", "r", stdin);

freopen("output.txt", "w", stdout);

#endif

**//fastio()**

#define fastio std::ios\_base::sync\_with\_stdio (false); cin.tie(0); cout.tie(0)

**//BigMOD**

ll BigMod(ll a, ll p, ll m){

ll ret=1;

while(p){

if(p&1) ret\*=a;

a\*=a;

p/=2;

}

return ret;

}

**//Addition Multiplication**

ll AddMul(ll a, ll p, ll m){

ll ret=0;

while(p){

if(p&1) ret=((ret%m)+(a%m))%m;

a=((a%m)+(a%m))%m;

}

return ret;

}

**//Normal Sieve**

#define SZ 1000005

int MARK[SZ+1];

vector<int>PRIME;

void sieve(){

MARK[1]=1;

int root=sqrt(SZ);

MARK[0]=MARK[1]=1;

for(int i=4; i<SZ; i+=2) MARK[i]=1;

for(int i=3; i<=root; i+=2){

if(!MARK[i]){

for(int j=i\*i; j<=SZ; j+=(i\*2)){

MARK[j]=1;

}

}

}

PRIME.push\_back(2); for(int i=3; i<=SZ; i+=2){

if(!MARK[i]) PRIME.push\_back(i);

}

return;

}

**//Segmented sieve**

#include<iostream>

#include<stdio.h>

#include<math.h>

using namespace std;

typedef unsigned long long ull;

typedef long long ll;

ll sieve(ll l, ll r){

ll sz = r-l+1;

ll arr[sz]={0};

ll root = sqrt(r);

for(ll i = 2; i <= root; i++){

if(i >= l){

if(arr[i-l] == 0){

for(ll j = 2; i\*j <= r; j++){

arr[(i\*j)-l]=1;

}

}

}

else{

int start;

start = (l % i == 0) ? (l/i) : (l/i+1);

for(ll j = start; i\*j <= r; j++){

arr[(i\*j)-l] = 1;

}

}

}

ll cnt = 0;

for(ll i = 0; i < sz; i++){

if(arr[i]==0){

if(i==0&&l==1)

continue;

cnt++;

}

}

return cnt;

}

int main(){

ll l, r;

int t;

cin>>t;

for(int i = 1; i <= t; i++){

cin>>l>>r;

ll ans = sieve(l, r);

cout<<"Case "<<i<<": "<<ans<<endl;

}

return 0;

}

**//Bitwise Sieve**

#define M 40000000

int marked[M/64 + 2];

vector<int>prime;

#define on(x) (marked[x/64] & (1<<((x%64)/2)))

#define mark(x) marked[x/64] |= (1<<((x%64)/2))

bool isPrime(int num) {

return num > 1 && (num == 2 || ((num & 1) && !on(num)));

}

void sieve(int n) {

for (int i = 3; i \* i < n; i += 2) {

if (!on(i)) {

for (int j = i \* i; j <= n; j += i + i) {

mark(j);

}

}

}

prime.push\_back(2);

for(int i=3; i<M; i+=2) {

if(isPrime(i))

prime.pb(i);

}

}

**//nCr**

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;

}

**//nCr with DP**

ll nCr[Max][Max];

ll rec(ll n, ll r) {

if(n == r) {

return 1;

}

if(r == 1) {

return n;

}

ll &ret = nCr[n][r];

if(~ret) {

return ret;

}

ret = rec(n - 1, r) + rec(n - 1, r - 1);

return ret;

}

**//Sum of Divisors in a range**

ll triangle(ll a, ll b) {

return (a + b + 1) \* (b - a) / 2 ;

}

ll divSum(ll a, ll b) { // Sum of divisors between a to b

ll n = sqrt(b);

ll sum = 0;

for(ll i = 1; i <= n; i++) {

sum += i \* (b / i - a / i) + triangle(max(n, a / i), max(n, b / i));

}

return sum;

}

int main() {

ll a, b;

cin >> a >> b;

ll ans = divSum(a - 1, b);

cout << ans << endl;

return 0;

}

//**Sum of Divisors (hard)**

ll SOD(ll n)

{

ll res = 1;

ll sq = sqrt(n);

for(int i = 0; i < prime.size() && prime[i] <= sq; i++)

{

if(n % prime[i] == 0)

{

ll tm = prime[i];

ll sum = 1;

ll cn = 1;

while(n % tm == 0)

{

n /= tm;

cn \*= tm;

sum += cn;

}

sq = sqrt(n);

res \*= sum;

}

}

if(n != 1) res \*= (n + 1);

return res;

}

**//Divisors of a Factorial**

vector <ull> vec;

bool isPrime[1000101];

void sieve() {

int MAX = 1000100, sq = sqrt(MAX);

memset(isPrime, true, sizeof isPrime);

for(int i = 4; i <= MAX; i += 2) {

isPrime[i] = 0;

}

for(int i = 3; i <= sq; i += 2) {

if(isPrime[i]) {

for(int j = i \* i; j <= MAX; j += i) {

isPrime[j] = 0;

}

}

}

isPrime[1] = 0;

isPrime[0] = 0;

for(int p = 2; p < MAX; p++) {

if(isPrime[p]) {

vec.push\_back(p);

}

}

}

ull factorialDivisors(ull n) {

ull res = 1;

for(ull x : vec) {

ull p = x;

ull exp = 0;

while(p <= n) {

exp = exp + (n / p);

p = p \* x;

}

res = res \* (exp + 1);

}

return res;

}

int main() {

sieve();

cout << factorialDivisors(6) << endl;

return 0;

}

**//Inverse factorial**

#include <iostream>

#include <string>

using namespace std;

const int m = 1e9 + 9;

int main() {

string x; cin >> x;

int y = 0;

for (char i : x)y = (y \* 10LL + i - '0') % m;

for (int i = 1, f = 1;; i++) {

f = 1LL \* f \* i % m;

if (f == y) {

cout << i << '\n';

return 0;

}

}

}

**//Combination**

int number[20];

int n, k;

//call with: permutation(1, 0)

void combination(int at, int last)

{

if(at == k + 1)

{

for(int i = 1; i <= k; i++) printf("%d ", number[i]);

printf("\n");

return;

}

for(int i = last + 1; i <= n - k + at; i++)

{

number[at] = i;

combination(at + 1, i);

}

}

**//Next permutation**

do{

cout << arr[0] << " " << arr[1] << " " << arr[2] << "\n";

}while (next\_permutation(arr, arr + 3));

**//Trailing zero of factorial**

int findTrailingZeros(int n)

{

if (n < 0)

return -1;

int cn = 0;

for (int i = 5; n / i >= 1; i \*= 5)

{

cn += n / i;

}

return cn;

}

**//Returns S % X where S is a numerical string.**

ll stringModx(ll x) {

divide(strlen(s));

return modulo(x);

}

**//Inverse modulo**

// Function for extended Euclidean Algorithm

int gcdExtended(int a, int b, int\* x, int\* y);

// Function to find modulo inverse of a

void modInverse(int a, int m)

{

int x, y;

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

if (g != 1)

cout << "Inverse doesn't exist";

else

{

// m is added to handle negative x

int res = (x % m + m) % m;

cout << "Modular multiplicative inverse is " << res;

}

}

**//Function for extended Euclidean Algorithm**

int gcdExtended(int a, int b, int\* x, int\* y)

{

// Base Case

if (a == 0)

{

\*x = 0, \*y = 1;

return b;

}

// To store results of recursive call

int x1, y1;

int gcd = gcdExtended(b % a, a, &x1, &y1);

// Update x and y using results of recursive

// call

\*x = y1 - (b / a) \* x1;

\*y = x1;

return gcd;

}

**//Phi Function**

ll phi[Max];

void phigen(int n)

{

for(int i = 1; i <= n; i++)

{

phi[i] = i;

}

for(int p = 2; p <= n; p++)

{

if(phi[p] == p)

{

phi[p] = p - 1;

for(int i = 2 \* p; i <= n; i += p)

{

phi[i] = (phi[i] / p) \* (p - 1);

}

}

}

}

**//PHI from 1 to N**

void phi\_1\_to\_n(int n)

{

vector<int> phi(n + 1);

for (int i = 0; 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;

}

}

}

**//Divisor Count**

ll divcnt[Max];

void DivisorCount(ll n)

{

for(int i = 1; i <= n; i++)

{

for(int j = i; j <= n; j += i)

{

divcnt[j]++;

}

}

}

**//Miller Rabin Primality Test**

inline bool miller(ll p, int iter = 20)

{

if(p == 3 || p == 2 || p == 5)

{

return true;

}

if(p % 2 == 0)

{

return false;

}

if(p < 3)

{

return false;

}

mt19937\_64 rnd(chrono::steady\_clock::now().time\_since\_epoch().count());

for(int i = 0; i < iter; i++)

{

ll a = (rnd()) % (p - 4) + 2;

ll s = p - 1;

while(s % 2 == 0)

{

s /= 2;

}

ll mod = bigmod(a, s, p);

if(mod == 1 || mod == p - 1)

{

continue;

}

bool flag = 0;

s \*= 2;

while(s != p - 1)

{

mod = mulmod(mod, mod, p);

if(mod == p - 1)

{

flag = 1;

break;

}

s \*= 2;

}

if(flag == 0)

{

return 0;

}

}

return 1;

}

**//Legendre’s formula.. returns largest power of P that divides n!**

ll largestPower(ll n, ll p)

{

ll x = 0;

// Calculate x = n/p + n/(p^2) + n/(p^3) + ....

while(n)

{

n /= p;

x += n;

}

return x;

}

**//Sum of NOD**

int SNOD(int n)

{

int res = 0;

int u = sqrt(n);

for(int i = 1; i <= u; i++)

{

res += (n / i) - i;

}

res \*= 2;

res += u;

return res;

}

**//Geometry formulas**

**Perimeter**

Perimeter of a square: s + s + s + s

s:length of one side

Perimeter of a rectangle: l + w + l + w

l: length

w: width

Perimeter of a triangle: a + b + c

a, b, and c: lengths of the 3 sides

**//Area**

Area of a square: s × s

s: length of one side

Area of a rectangle: l × w

l: length

w: width

Area of a triangle: (b × h)/2

b: length of base

h: length of height

Area of a trapezoid: (b1 + b2

) × h/2

b1 and b2

: parallel sides or the bases

h: length of height

**//Volume**

Volume of a cube: s × s × s

s: length of one side

Volume of a box: l × w × h

l: length

w: width

h: height

Volume of a sphere: (4/3) × pi × r

3

pi: 3.14

r: radius of sphere

Volume of a triangular prism: area of triangle × Height = (1/2 base × height) ×

Height

base: length of the base of the triangle

height: height of the triangle

Height: height of the triangular prism

Volume of a cylinder: pi × r

2 × Height

pi: 3.14

r: radius of the circle of the base

Height: height of the cylinder

**//Kruskal**

const int Max = 15e3 + 10;

struct Node

{

int u, v, w;

} g[Max];

bool less(Node a, Node b)

{

return a.w < b.w;

}

bool more(Node a, Node b)

{

return a.w > b.w;

}

int node, edge, parent[Max];

int Find\_parent(int n)

{

//cout << ": " << n << " " << parent[n] << endl;

if(parent[n] == n)

{

return n;

}

return Find\_parent(parent[n]);

}

void graph()

{

for(int i = 1; ; i++)

{

cin >> g[i].u >> g[i].v >> g[i].w;

edge++;

}

}

int kruskal()

{

int sum = 0;

for(int i = 0; i <= node; i++)

{

parent[i] = i;

}

for(int i = 1; i <= edge; i++)

{

int u = Find\_parent(g[i].u), v = Find\_parent(g[i].v);

//cout << u << " " << v << endl;

if(u != v)

{

//printf("The cost from %d to %d is : %d\n", s1[i], s2[i], w[i]);

parent[u] = v;

sum += g[i].w;

}

}

return sum;

}

int main()

{

int t;

cin >> t;

for(int tc = 1; tc <= t; tc++)

{

cin >> node;

edge = 0;

graph();

// Minimus

sort(g + 1, g + edge + 1, less);

cout << kruskal() << endl;

// Maximum

sort(g + 1, g + edge + 1, more);

cout << kruskal() << endl;

}

return 0;

}

**//Dijkstra**

struct Node

{

int at, cost;

Node(int \_at, int \_cost)

{

at = \_at;

cost = \_cost;

}

};

bool operator<(Node a, Node b)

{

return a.cost > b.cost;

}

struct Edge

{

int v, w;

Edge(int \_v, int \_w)

{

v = \_v;

w = \_w;

}

};

vector <Edge> G[10001];

priority\_queue <Node> pq;

int dist[10001];

int n, m, s;

void dijsktra(int src)

{

for(int i = 1; i <= n; i++)

{

dist[i] = 1e9;

}

dist[src] = 0;

pq.push(Node(src, 0));

while(!pq.empty())

{

Node u = pq.top();

pq.pop();

if(u.cost != dist[u.at])

{

continue;

}

for(int i = 0; i < G[u.at].size(); i++)

{

Edge e = G[u.at][i];

if(dist[e.v] > u.cost + e.w)

{

dist[e.v] = u.cost + e.w;

pq.push(Node(e.v, dist[e.v]));

}

}

}

}

**//Segment Tree - Lazy Propagation**

const int mx = 1e5 + 10;

ll a[mx];

struct Node

{

ll sm, prop;

} seg[4\*mx];

void build(ll nod, ll lo, ll hi)

{

if(lo==hi)

{

seg[nod].sm=a[lo];

return;

}

ll mid = (lo+hi)>>1;

build(nod\*2, lo, mid);

build(nod\*2+1, mid+1, hi);

seg[nod].sm = seg[nod\*2].sm + seg[nod\*2+1].sm;

seg[nod].prop = 0;

}

ll query(ll nod, ll lo, ll hi, ll l, ll r, ll cary)

{

if(lo>r || hi<l) return 0;

if(lo>=l && hi<=r)

{

return seg[nod].sm + cary\*(r-l+1);

}

ll mid=(lo+hi)>>1;

ll x = query(nod\*2, lo, mid, l, r, cary + seg[nod].prop);

ll y = query(nod\*2+1, mid+1, hi, l, r, cary + seg[nod].prop);

return x+y;

}

void update(ll nod, ll lo, ll hi, ll l, ll r, ll val)

{

if(lo>r || hi<l) return;

if(lo>=l && hi<=r)

{

seg[nod].sm += ((r-l+1) \* val);

seg[nod].prop += val;

return;

}

ll mid=(lo+hi)>>1;

update(nod\*2, lo, mid, l, r, val);

update(nod\*2+1, mid+1, hi, l, r, val);

seg[nod].sm = seg[nod\*2].sm + seg[nod\*2+1].sm + (r-l+1) \* seg[nod].prop;

}

int main()

{

ll n, q;

cin>>n>>q;

for(ll i=0; i<n; i++) cin>>a[i];

build(1, 0, n-1);

for(ll i=0; i<q; i++)

{

int x;

cin>>x;

if(x==2)

{

ll l, r;

cin>>l>>r;

cout<<query(1, 0, n-1, l, r, 0)<<endl;

}

else

{

ll val, pos;

cin>>pos>>val;

update(1, 0, n-1, pos-1, pos-1, val);

}

}

}

**//White-Black (two-color) - Balanced Subtrees(dfs)**

void dfs(int s)

{

for(int i=0; i<nod[s].size(); i++)

{

dfs(nod[s][i]);

b[s-1] += b[nod[s][i]-1];

w[s-1] += w[nod[s][i]-1];

}

}

int main()

{

int t;

cin>>t;

while(t--)

{

int n, q;

cin>>n;

w[0]=b[0]=0;

for(int i=0; i<=n; i++) nod[i].clear();

for(int i=1; i<n; i++)

{

int x;

cin>>x;

nod[x].push\_back(i+1);

w[i]=b[i]=0;

}

cin>>st;

for(int i=0; i<n; i++)

{

if(st[i]=='W')

{

w[i]=1;

b[i]=0;

}

else

{

w[i]=0;

b[i]=1;

}

}

dfs(1);

int cnt=0;

for(int i=0; i<n; i++)

{

if(b[i]==w[i]) cnt++;

}

cout<<cnt<<endl;

}

return 0;}

**//Cycle Detection (DFS)**

vector<int>nod[1000];

int visited[1000] = {0};

int cycle=0;

void dfs(int s)

{

visited[s]=1;

for(int i=0; i<nod[s].size(); i++)

{

int v=nod[s][i];

if(visited[v]==0)

{

dfs(v);

}

else if(visited[v]==1)

{

cycle++;

}

}

visited[s]=2;

}

int main()

{

int n, e;

cin>>n>>e;

for(int i=0; i<e; i++)

{

int a, b;

cin>>a>>b;

nod[a].push\_back(b);

}

for(int i=1; i<=n; i++)

{

if(visited[i]) continue;

dfs(i);

}

if(cycle==0) cout<<"Cycle is not found"<<endl;

else cout<<"Cycle is found"<<endl;

}

**//BFS on GRID**

#include <bits/stdc++.h>

using namespace std;

#define ROW 4

#define COL 4

int dRow[] = { -1, 0, 1, 0 };

int dCol[] = { 0, 1, 0, -1 };

bool isValid(bool vis[][COL],

int row, int col)

{

if (row < 0 || col < 0

|| row >= ROW || col >= COL)

return false;

if (vis[row][col])

return false;

return true;

}

void BFS(int grid[][COL], bool vis[][COL],

int row, int col)

{

queue<pair<int, int> > q;

q.push({ row, col });

vis[row][col] = true;

while (!q.empty()) {

pair<int, int> cell = q.front();

int x = cell.first;

int y = cell.second;

cout << grid[x][y] << " ";

q.pop();

for (int i = 0; i < 4; i++) {

int adjx = x + dRow[i];

int adjy = y + dCol[i];

if (isValid(vis, adjx, adjy)) {

q.push({ adjx, adjy });

vis[adjx][adjy] = true;

}

}

}

}

int main()

{

int grid[ROW][COL] = { { 1, 2, 3, 4 },

{ 5, 6, 7, 8 },

{ 9, 10, 11, 12 },

{ 13, 14, 15, 16 } };

bool vis[ROW][COL];

memset(vis, false, sizeof vis);

BFS(grid, vis, 0, 0);

return 0;

}

**//Path print by (bfs)**

#include<bits/stdc++.h>

using namespace std;

#define ll long long

#define pil pair<int,ll>

#define inf 1000000009

int ara[100005];

ll dis[100005];

ll prnt[100005];

vector<pil>edge[100005];

void BFS(int s,int d)

{

for(int i=0; i<=d; i++) dis[i]=10000000000009;

priority\_queue<pil>Q;

Q.push(pil(s,0));

dis[s]=0;

while(!Q.empty())

{

int u = Q.top().first;

Q.pop();

// cout<<u<<road[u].size()<<endl;

for(int i=0; i<edge[u].size(); i++)

{

int v = edge[u][i].first;

ll vcost = edge[u][i].second + dis[u];

//cout<<bus[v]<<' '<<bus[u]<<endl;

if(dis[v]>vcost)

{

dis[v]=vcost;

prnt[v]=u;

// cout<<u<<' '<<v<<endl;

Q.push(pil(v,-dis[v]));

}

}

}

int x=d;

int i=0;

if(dis[d]==10000000000009)

{

cout<<-1<<endl;

return;

}

while(true)

{

ara[i++]=x;

if(x==s) break;

x=prnt[x];

}

for(int j=i-1; j>=0; j--) cout<<ara[j]<<' ';

cout<<endl;

}

int main()

{

string str;

int n,m,k;

while(cin>>n>>m)

{

for(int i=0; i<=n; i++)

{

edge[i].clear();

}

int u,v,w;

for(int i=0; i<m; i++)

{

cin>>u>>v>>w;

edge[u].push\_back(pil(v,w));

edge[v].push\_back(pil(u,w));

}

BFS(1,n);

}

return 0;

}

nod[b].push\_back(a);

cost[a].push\_back(c);

cost[b].push\_back(c);

}

bfs(1, n);

}

**//KMP**

vector<int> constructLPSarray(string pattern)

{

vector<int> lps(pattern.size()+5);

int len = 0;

lps[0] = 0;

for(int i=1; i<pattern.size();)

{

if(pattern[i] == pattern[len])

{

lps[i] = len+1;

len++;

i++;

}

else

{

if(len!=0) len = lps[len-1];

else

{

lps[i] = len; /// len = 0;

i++;

}

}

}

return lps;

}

void KMPpatternMatching(string text, string pattern)

{

vector<int> lps = constructLPSarray(pattern);

int textLength = text.size();

int patternLength = pattern.size();

int i=0, j=0; // i for text and j for pattern

while(i < textLength)

{

if(text[i] == pattern[j])

{

i++;

j++;

}

else

{

if(j!=0) j = lps[j-1];

else i++;

}

if(j == patternLength)

{

cout<< "found match at: "<< (i - patternLength)<<endl;

j = lps[j-1];

}

}

}

**//Prim’s Algorithm for finding MST**

const int Max = 1e5 + 10;

bool vist[Max];

vector <pll> G[Max];

ll prim(int src)

{

priority\_queue <pll, vector<pll>, greater<pll>> q;

ll mn = 0;

q.push(make\_pair(0, src));

while(!q.empty())

{

pll p = q.top();

q.pop();

int u = p.second;

if(vist[u] == true)

{

continue;

}

mn += p.first;

vist[u] = true;

for(pll v : G[u])

{

if(vist[v.second] == false)

{

q.push(v);

}

}

}

return mn;

}

int main()

{

int n, m, u, v;

ll w, mn;

cin >> n >> m;

for(int i = 1; i <= m; i++)

{

cin >> u >> v >> w;

G[u].push\_back(make\_pair(w, v));

G[v].push\_back(make\_pair(w, u));

}

mn = prim(1);

cout << mn << endl;

return 0;

}

**//Longest increasing subsequence**

void solve(){

ll n;

cin>>n;

ll arr[n];

for(int i=0; i<n; i++)

cin>>arr[i];

vector<ll>v;

v.push\_back(arr[0]);

for(int i=1; i<n; i++){

if(arr[i]>v.back()) v.push\_back(arr[i]);

else{

int ind=lower\_bound(v.begin(),v.end(),arr[i])-v.begin();

v[ind]=arr[i];

}

}

cout<<v.size()<<endl;

}

**//Longest common subsequence**

const int Max = 1000 + 5;

int dp[Max][Max];

bool vist[Max][Max];

string s, t;

int n, m;

int lcs(int i, int j)

{

if(i >= n or j >= m)

{

return 0;

}

int &ret = dp[i][j];

bool &vis = vist[i][j];

if(vis)

{

return ret;

}

vis = 1;

int res = 0;

if(s[i] == t[j])

{

res = 1 + lcs(i + 1, j + 1);

}

else

{

res = max(lcs(i + 1, j), lcs(i, j + 1));

}

return ret = res;

}

string ans;

void solution(int i, int j)

{

if(i >= n or j >= m)

{

return;

}

if(s[i] == t[j])

{

ans += s[i];

solution(i + 1, j + 1);

}

else

{

if(lcs(i + 1, j) > lcs(i, j + 1))

{

solution(i + 1, j);

}

else

{

solution(i, j + 1);

}

}

}

int main()

{

int T;

cin >> T;

for(int tc = 1; tc <= T; tc++)

{

cin >> s >> t;

n = s.size();

m = t.size();

memset(vist, 0, sizeof vist);

int maxlen = lcs(0, 0);

ans = "";

solution(0, 0);

cout << ans << '\n';

}

}

**//Coin Change**

*In a strange shop there are n types of coins ofvalue A1*

*, A2*

*... An*

*. You have to find the number of waysyou can make K using*

*the coins. You can use anycoin at most K times.*

*For example, suppose there are three coins 1, 2, 5. Then if K = 5 the possible ways are:*

*11111*

*1112*

*122*

*5*

*So, 5 can be made in 4 ways.*

*Solution:*

ll ar[Max];

int main()

{

ll t, n, k;

while(scanf("%lld", &t) == 1)

{

int tc = 1;

while(t--)

{

scanf("%lld %lld", &n, &k);

ll dp[k + 1];

memset(dp, 0, sizeof dp);

dp[0] = 1;

for(int i = 1; i <= n; i++)

{

scanf("%lld", &ar[i]);

}

for(int i = 1; i <= n; i++)

{

for(int j = 1; j <= k; j++)

{

if(ar[i] <= j)

{

dp[j] = dp[j] % Mod + dp[j - ar[i]] % Mod;

dp[j] %= Mod;

}

}

}

printf("Case %d: %lld\n", tc, dp[k]);

tc++;

}

}

return 0;

}

**//SOS DP**

*/\**

*Problem: Given a fixed array A of 2N integers, we need to calculate ∀ x*

*function F(x) = Sum of all A[i] such that x&i = i, i.e., i is a subset of x.*

*\*/*

for(int i = 0; i < (1 << N); ++i)

{

F[i] = A[i];

}

for(int i = 0; i < N; ++i)

{

for(int mask = 0; mask < (1 << N); ++mask)

{

if(mask & (1 << i))

{

F[mask] += F[mask ^ (1 << i)];

}

}

}

**//Longest Unique Subarray**

cin >> n;

vector<int > v;

map<int,bool > mp;

int mx=0;

int currans=0;

int j=0;

for(int i=0; i<n; i++)

{

int k;

cin>>k;

v.push\_back(k);

if(mp.find(k)==mp.end()|| mp[k]==false)

{

mp[k]=true;

currans++;

mx=max(currans,mx);

}

else

{

while(v[j]!=k)

{

mp[v[j]]=false;

j++;

}

currans=(i-j);

j++;

}

}

cout<<mx<<endl;

**//Remove k digit build lowest number**

string num;

int k;

cin>>num>>k;

stack<char>stk;

int sz=num.size();

for(int i=0; i<sz; i++)

{

while(k>0 && !stk.empty() && stk.top()>num[i])

{

stk.pop();

k--;

}

if(num[i]!='0') stk.push(num[i]);

else if(!stk.empty()) stk.push(num[i]);

}

while(!stk.empty() && k--)

{

stk.pop();

}

vector<char> ans;

while(!stk.empty())

{

ans.push\_back(stk.top());

stk.pop();

}

reverse(ans.begin(), ans.end());

if(ans.empty()) cout<<0;

for(auto i:ans)

{

cout<<i;

}

cout<<endl;

**//Subarray Sum count number of subarrays have same sum**

cin>>n>>x;

map<ll, ll>mp;

ll A[n+3], cnt=0, sum=0;

for(int i=0; i<n; i++)

{

cin>>k;

sum+=k;

A[i]=sum;

}

for(int i=0; i<n; i++)

{

if(A[i]==x) cnt++;

if(mp[A[i]-x]>=1) cnt+=mp[A[i]-x];

mp[A[i]]++;

}

cout<<cnt<<endl;

**//Minimum Lexicographical Rotation**

int minimumExpression(string s)

{

s = s + s;

int i = 0, j = 1, k = 0, len = s.size();

while(i + k < len && j + k < len)

{

if(s[i + k] == s[j + k]) k++;

else if(s[i + k] < s[j + k])

{

j = max(j + k + 1, i + 1);

k = 0;

}

else

{

i = max(i + k + 1, j + 1);

k = 0;

}

}

return min(i, j);

}

**//Compare for sort()**

typedef pair<int,int> ii;

#define first f

#define s second

bool cmp(ii a,ii b)

{

int as=a.f+a.s,bs=b.f+b.s;

if(as!=bs)

return as<bs;

return a.f<b.f;

}

**//Counting Sort**

void counting\_sort(int A[], int Aux[], int sortedA[], int N)

{

// First, find the maximum value in A[]

int K = 0;

for(int i=0; i<N; i++)

{

K = max(K, A[i]);

}

// Initialize the elements of Aux[] with 0

for(int i=0 ; i<=K; i++)

{

Aux[i] = 0;

}

// Store the frequencies of each distinct element of A[],

// by mapping its value as the index of Aux[] array

for(int i=0; i<N; i++)

{

Aux[A[i]]++;

}

int j = 0;

for(int i=0; i<=K; i++)

{

int tmp = Aux[i];

// Aux stores which element occurs how many times,

// Add i in sortedA[] according to the number of times i occured in A[]

while(tmp--)

{

//cout << Aux[i] << endl;

sortedA[j] = i;

j++;

}

}

}

**//Merge Sort**

void merge(int A[ ], int start, int mid, int end)

{

//stores the starting position of both parts in temporary variables.

int p = start,q = mid+1;

int Arr[end-start+1], k=0;

for(int i = start ; i <= end ; i++)

{

if(p > mid) //checks if first part comes to an end or not .

Arr[ k++ ] = A[ q++] ;

else if ( q > end) //checks if second part comes to an end or not

Arr[ k++ ] = A[ p++ ];

else if( A[ p ] < A[ q ]) //checks which part has smaller element.

Arr[ k++ ] = A[ p++ ];

else

Arr[ k++ ] = A[ q++];

}

for (int p=0 ; p< k ; p ++)

{

/\* Now the real array has elements in sorted manner including both

parts.\*/

A[ start++ ] = Arr[ p ] ;

}

}

void merge\_sort (int A[ ], int start, int end )

{

if( start < end )

{

int mid = (start + end ) / 2 ; // defines the current array in 2 parts .

merge\_sort (A, start, mid ) ; // sort the 1st part of array .

merge\_sort (A,mid+1, end ) ; // sort the 2nd part of array.

// merge the both parts by comparing elements of both the parts.

merge(A,start, mid, end );

}

}

**//Next Greater Element // O(n)**

vector<int> nextGreaterElement(vector<int> &arr)

{

int n = arr.size();

stack<int> s;

vector<int> ret(n + 1, n);

for(int i = n - 1; i >= 0; i--)

{

while(!s.empty() && arr[s.top()] <= arr[i])

{

s.pop();

}

if(!s.empty())

{

ret[i] = s.top();

}

s.push(i);

}

return ret;

}

**//STL**

**String Letter transformation:**

transform(S.begin(), S.end(), S.begin(), :: toupper);

transform(S.begin(), S.end(), S.begin(), :: tolower);

**//Upper bound & lower bound:**

For vector:

lower = lower\_bound(v.begin(), v.end(), num) - v.begin();

upper = upper\_bound(v.begin(), v.end(), num) - v.begin();

**//For array:**

lower = lower\_bound(A, A+n, num)-A;

upper = upper \_bound(A, A+n, num)-A;

**//Priority Queue:**

In descending order:

priority\_queue<int> pq;

In ascending order:

priority\_queue< int, vector <int>, greater <int> > apq;

**//Pair Map:**

map< pair<int, int>, int> prmp;

prmp[ {a, b}] = value;

**Set:**

set<int> st;

st.insert(19);

st.erase(19);

\*st.begin();

[for minimum number in set]

\*st.rbegin();

[for maximum number in set]

**//Mod of a string:**

int mod = 0;

mod = (mod\*10 + s[i]-'0') % num;

**//String Multiplication:**

string multiply(string num1, string num2)

{

int len1 = num1.size();

int len2 = num2.size();

if (len1 == 0 || len2 == 0)

return "0";

vector<int> result(len1 + len2, 0);

int i\_n1 = 0;

int i\_n2 = 0;

for (int i=len1-1; i>=0; i--)

{

int carry = 0;

int n1 = num1[i] - '0';

i\_n2 = 0;

for (int j=len2-1; j>=0; j--)

{

int n2 = num2[j] - '0';

int sum = n1\*n2 + result[i\_n1 + i\_n2] + carry;

carry = sum/10;

result[i\_n1 + i\_n2] = sum % 10;

i\_n2++;

}

if (carry > 0)

result[i\_n1 + i\_n2] += carry;

i\_n1++;

}

int i = result.size() - 1;

while (i>=0 && result[i] == 0)

i--;

if (i == -1)

return "0";

string s = "";

while (i >= 0)

s += std::to\_string(result[i--]);

return s;

}

int main()

{

string str1;

string str2;

if((str1.at(0) == '-' || str2.at(0) == '-') &&

(str1.at(0) != '-' || str2.at(0) != '-' ))

cout<<"-";

if(str1.at(0) == '-')

str1 = str1.substr(1);

if(str2.at(0) == '-')

str2 = str2.substr(1);

cout << multiply(str1, str2);

return 0;

}

**//String Addition:**

string findSum(string str1, string str2)

{

if (str1.length() > str2.length())

swap(str1, str2);

string str = "";

int n1 = str1.length(), n2 = str2.length();

reverse(str1.begin(), str1.end());

reverse(str2.begin(), str2.end());

int carry = 0;

for (int i=0; i<n1; i++)

{

int sum = ((str1[i]-'0')+(str2[i]-'0')+carry);

str.push\_back(sum%10 + '0');

carry = sum/10;

}

for (int i=n1; i<n2; i++)

{

int sum = ((str2[i]-'0')+carry);

str.push\_back(sum%10 + '0');

carry = sum/10;

}

if (carry)

str.push\_back(carry+'0');

reverse(str.begin(), str.end());

return str;

}

**//String:**

sort(s.rbegin(), s.rend()) // sort in non-increasing order

int n = unique(s.begin(),s.end()) – sbegin(); // getting unique char in string

for (int i – 0; i < n; i++) cout << s[i] << endl;

cout << \*max\_element(s.begin(), s.end()) << endl;

// **max char from string**

cout << \* min\_element(s.begin(), s.end()) << endl;

**//when we want to take input with space**:

char c;

cin >> c;

getline(cin, s)

s = c + s;

// **deleting substring from string**

s.erase(s.begin() + 3, s.end()+7) // helloWorld remove loWo from helloworld

// **push any substring to string**

Tmp = hello alam hello

S = tasdid

copy(tmp.begin() + 6, tmp.begin()+10, back\_inserter(s)) ;

cout << s << endl; // tasdid alam

// **specific char remove from string**

s.erase( remove ( s.begin(), s.end(), ‘a’ ), s.end() ) ; // remove all ‘a’

// **check given string is substring or not ( n \* m ) complexity**

if( s.find(“hello”) != -1 ) cout << “item found “<<endl;

else cout << “not found”<< endl;

**//Map:**

map<int, int> mp1;

map<int, pair<int, int>> mp2;

//**if value for a key is not assigned, default value of valueType will be assigned**

//for integers it is 0, for pair<int, int> it is {0, 0}

cout << mp1[0] << "\n"; //0

cout << mp2[0].first << " " << mp2[0].second << "\n"; //0 0

map<int, int> mp3 = {{1, 2}, {3, 4}, {5, 6}};

cout << mp3[1] << " " << mp3[3] << " " << mp3[5] << "\n"; //2 4 6

map<pair<int, int>, int> mp4;

mp4[ {4, 5}] = -1;

mp4[ {1, 3}] = -2;

cout << mp4[ {1, 3}] << " " << mp4[ {4, 5}] << "\n"; //-1 -2

//each element of map is a {key, value} pair

//and they are sorted

for(pair<pair<int, int>, int> p: mp4)

{

cout << p.first.first << " " << p.first.second << " " << p.second << "\n";

}

/\*

1 3 -2

4 5 -1

\*/

map<int, int> mp5;

cout << mp1.size() << " " << mp2.size() << " " << mp3.size() << " " << mp4.size()

<< mp5.size() << "\n"; // 1 1 3 2 0

// map key finding:

map<int, int> mp;

if(mp.find(0) == mp.end()) cout << "not found\n";

else cout << "found\n";

//not found

mp[0] = 1;

if(mp.find(0) == mp.end()) cout << "not found\n";

else cout << "found\n";

set:

**//checking element present or not**

cout << s.count(2) << endl // if present print 1 else 0

cout << \*s.begin() << endl // first element

cout << \*s.rbegin() << endl; // ;last elemet

s.erase(2) // delete 2 from set

s.erase(s.begin()) // first element delete

s.erase(--s.end()) // delete last

set < int, greater<int>>s = { 3,2,1,4}

// output : 4 3 2 1

**//MATH**:

\* if x and m coprime, x ^ phi(m) = 1 (mod m)

\* if x and m coprime, x ^ n = x ^ (n mod phi(m)) (mod m)

**//Matrix expo**

#include<bits/stdc++.h>

using namespace std;

#define ull unsigned long long

#define ll long long

ll a,b,n,x;

void matmul(ll a[2][2], ll b[2][2]){

ll mul[2][2];

for(int i=0; i<2; i++){

for(int j=0; j<2; j++){

mul[i][j]=0;

for(int k=0; k<2; k++){

mul[i][j]=(mul[i][j]+(a[i][k]\*b[k][j])%x)%x;

}

}

}

for(int i=0; i<2; i++){

for(int j=0; j<2; j++){

b[i][j]=mul[i][j];

//cout<<a[i][j]<<" ";

}

//cout<<endl;

}

return;

}

void find\_ans(ll m[2][2],ll b[2][2], ll p){

while(p){

if(p&1) matmul(m,b);

matmul(m,m);

p/=2;

}

return;

}

void solve(){

cin>>a>>b>>n>>x;

x=pow(10,x);

ll base[2][2]={{b,0},{a,0}};

ll mat[2][2]={{1,1},{1,0}};

ll ans;

if(n==1) ans=b;

else if(n==0) ans=a;

else{

find\_ans(mat,base,n-1);

ans=base[0][0]+base[0][1];

}

cout<<ans<<endl;

return;

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(0); cout.tie(0);

#ifndef ONLINE\_JUDGE

freopen("input.txt", "r", stdin);

freopen("output.txt", "w", stdout);

#endif

int t, tc=1;

cin>>t;

while(t--){

cout<<"Case "<<tc++<<": ";

solve();

}

//solve();

return 0;

}

**//Mo Algorithm**

problem link- https://www.hackerrank.com/contests/gub-idpc-2022/challenges/frequency-xor/copy-from/1347288030

#include<bits/stdc++.h>

using namespace std;

#define ull unsigned long long

#define ll long long

#define pii pair<int,int>

#define MAX 1000005

int arr[MAX],cnt[MAX], ans[MAX];

int n,q,block\_size,ansr=0;

pair<pii,int>qry[MAX];

bool cmp(pair<pii,int>x,pair<pii,int>y){

int xx=x.first.first/block\_size;

int yy=y.first.first/block\_size;

if(xx!=yy) return xx<yy;

return x.first.second<y.first.second;

}

void add(int x){

if(cnt[x]==0){

ansr^=1;

cnt[x]++;

}

else{

ansr^=cnt[x];

cnt[x]++;

ansr^=cnt[x];

}

}

void Remove(int x){

ansr^=cnt[x];

cnt[x]--;

ansr^=cnt[x];

}

void solve2(){

int l=0, r=-1;

set<int>s;

for(int i=0; i<q; i++){

int lp=qry[i].first.first;

int rp=qry[i].first.second;

while(r<rp){

r++;

add(arr[r]);

}

while(r>rp){

Remove(arr[r]);

r--;

}

while(l<lp){

Remove(arr[l]);

l++;

}

while(l>lp){

l--;

add(arr[l]);

}

ans[qry[i].second]=ansr;

}

for(int i=0; i<q; i++)

cout<<ans[i]<<endl;

}

void solve(){

cin>>n;

for(int i=1; i<=n; i++) cin>>arr[i];

cin>>q;

for(int i=0; i<q; i++){

int x,y; cin>>x>>y;

qry[i].first.first=x;

qry[i].first.second=y;

qry[i].second=i;

}

block\_size=sqrt(n);

sort(qry,qry+q,cmp);

solve2();

return;

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(0); cout.tie(0);

int t=1,tc=1;

//cin>>t;

while(t--){

solve();

}

return 0;

}

**//Pollard rho**

#include<bits/stdc++.h>

using namespace std;

#define ull unsigned long long

#define ll long long

#define SZ 1000005

int MARK[SZ+1];

vector<int>PRIME;

void sieve(){

MARK[1]=1;

int root=sqrt(SZ);

for(int i=3; i<=root; i+=2){

if(!MARK[i]){

for(int j=i\*i; j<=SZ; j+=(i\*2)){

MARK[j]=1;

}

}

}

PRIME.push\_back(2);

for(int i=3; i<=SZ; i+=2){

if(!MARK[i]) PRIME.push\_back(i);

}

return;

}

//ll Mul(ll a, ll p, ll m){//If we use recursive function for this part then CPU limit will be exit

// if(p==0) return 0;

// ll ret=Mul(a,p/2,m);

// ret=((ret%m)+(ret%m))%m;

// if(p&1) ret=((ret%m)+(a%m))%m;

//

// return ret;

//}

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 p, ll m){

// if(p==0) return 1;

// ll ret=bigmod(a,p/2,m);

// ret\*=ret;

// if(p&1) ret\*=a;

//

// 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(ll n){

if(n==2) return 1;

if(n%2==0) return 0;

ll 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++){

ll x=test[i]%(n-2), temp=d;

if(x<2) x+=2;

ll 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;

}

ll pollard\_rho(ll n, ll c){

ll 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;

}

void llfactorize(ll n, vector<ll> &f){

if(n==1) return;

if(n < 1e9){

for(int i=0; PRIME[i]\*PRIME[i] <= n; i++){

if(n%PRIME[i]==0){

while(n%PRIME[i]==0){

f.push\_back(PRIME[i]);

n/=PRIME[i];

}

}

}

if(n!=1) f.push\_back(n);

return;

}

if(isprime(n)){

f.push\_back(n);

return;

}

ll d=n;

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

d=pollard\_rho(n,i);

}

llfactorize(d,f);

llfactorize(n/d,f);

}

void factorize(ll n, vector<pair<ll,ll>> &ans){

vector<ll>v;

llfactorize(n,v);

if(v.size()==0) return;

sort(v.begin(), v.end());

ll 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<ll,ll>> &ans){

ll ph=n;

for(auto i:ans){

ph/=i.first;

ph\*=(i.first-1);

}

return ph;

}

void solve(){

ll n,ans;

cin>>n;

vector<pair<ll,ll>>v;

factorize(n,v);

ll phi\_n=phi(n,v);

ll b=n+1;

while(1){

vector<pair<ll,ll>>vv;

factorize(b,vv);

ll phi\_b=phi(b,vv);

if(phi\_b>phi\_n){

cout<<b<<endl;

break;

}

b++;

}

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(0); cout.tie(0);

#ifndef ONLINE\_JUDGE

//freopen("input.txt", "r", stdin);

//freopen("output.txt", "w", stdout);

#endif

sieve();

int t=1,tc=1;

cin>>t;

while(t--){

solve();

}

return 0;

}

**//Large Division**

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main(){

int test, cs = 1;

cin>>test;

while(test--){

string s;

ll div;

cin>>s>>div;

ll temp = 0;

if(div < 0)

div = -1\*div;

for(int i = 0; i < s.size(); i++){

if(s[i] == '-')

continue;

temp = temp\*10+(s[i]-'0');

if(temp >= div){

temp %= div;

}

}

cout<<"Case "<<cs++<<": ";

if(temp == 0)

cout<<"divisible"<<endl;

else

cout<<"not divisible"<<endl;

}

}

**//Segment tree**

#include <bits/stdc++.h>

using namespace std;

void build(int node,int start,int end,int tree[],int a[])

{

if(start==end)

{

tree[node]=a[start];

}

else

{

int mid=(start+end)/2;

build(2\*node+1,start,mid,tree,a);

build(2\*node+2,mid+1,end,tree,a);

tree[node]=min(tree[2\*node+1],tree[2\*node+2]);

}

}

void update(int node,int start,int end,int idx,int val,int tree[],int a[])

{

if(start==end)

{

a[idx]=val;

tree[node]=val;

}

else

{

int mid=(start+end)/2;

if(idx>=start&&idx<=mid)

{

update(2\*node+1,start,mid,idx,val,tree,a);

}

else

{

update(2\*node+2,mid+1,end,idx,val,tree,a);

}

tree[node]=min(tree[2\*node+1],tree[2\*node+2]);

}

}

int query(int node,int start,int end,int l,int r,int tree[])

{

if(l>end||start>r)

{

return INT\_MAX;

}

if(l<=start&&r>=end)

{

return tree[node];

}

int q1,q2;

int mid=(start+end)/2;

q1=query(2\*node+1,start,mid,l,r,tree);

q2=query(2\*node+2,mid+1,end,l,r,tree);

return(min(q1,q2));

}

void solve()

{

int n, q;

cin >> n >> q;

int a[n];

for(int i = 0; i < n; i++) cin >> a[i];

int tree[n \* 4];

build(0,0,n-1,tree,a);

while(q--)

{

int l, r;

cin >> l >> r;

cout << query(0, 0, n-1, l - 1, r - 1, tree) << endl;

}

}

int main()

{

// freopen("in.txt", "r", stdin);

// freopen("out.txt", "w", stdout);

int T;

cin >> T;

for(int t = 1; t <= T; t++)

{

cout << "Case " << t << ":\n";

solve();

}

return 0;

}

**//Segmented tree Tri**

#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++)

int tri[1000005][26]; //Total char in input file,Number of distinct char

bool flag[1000005]; //Indicate where string finishes

int id=1;

int main()

{

string str;

cin >> str;

int r=1;

REP(i,str.size())

{

int x=str[i]-'a'; // It maybe '0'/'A'/both

if(!tri[r][x])

{

tri[r][x]=++id;

}

r=tri[r][x];

}

flag[r]=true;

return 0;

}

**//Catalan Number**

ll catalon\_number(ll n){

vll catalon(n+5);

catalon[0]=1;

FOR(1, n+1){

catalon[i] = (catalon[i-1]\*(4\*i-2))/(i+1);

}

return catalon[n];

}

**The Catalan numbers:** 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845

**Distance:**

1. 2 point :
2. point - line: (ax1+by1+c)/sqrt(a^2+b^2)
3. Line - line: (c1-c2)/sqrt(a^2+b^2)

**intersection (m1:m2 ratio):**

X = (m1x2+m2x1)/(m1+m2)

y = (myx2+m2y1)/(m1+m2)

Given a straight line, find the parallel and perpendicular line:

Parallel: change C to K

Perpendicular: swap a and b with sign (bx-ay+k=0)

**# Equation to info:**

ax+by+c=0 dewa thakle x/p+y/q=c banate hobe

i. x,y theke kete niche p, q

ii. Cut by axis = sqrt(p\*p+q\*q)

iii. Axis theke utponno triangle er area = ½\*pq

**Combinatorics:**

1. N ta point theke straight line= nC2
2. ……… triange = nC3
3. ……… ractacangle = nc4
4. ……… diagonal = nC2-n
5. N ta frnd ke dawat = 2^n -1
6. N ta frnd ke gul table a bosanu = (n-1)!
7. N ta frnd ke sarite dar koranu = n!
8. N ta muktar mala = (n-1)!/2

**Ordered 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 Set = tree<T, null\_type,

less<T>, rb\_tree\_tag,

tree\_order\_statistics\_node\_update>;

Set <int> st;

int main()

{

st.insert(5); //Insert

st.erase(5); //Delete

st.insert(1);

st.insert(2);

st.insert(9);

cout << \*st.find\_by\_order(0) << endl; //Find value by rank

cout << st.order\_of\_key(9) << endl; //Find value's rank

/\* For multiple same element, use pair, store index in second of pair \*/

return 0;

}

**# Bitwise operation & Functions**

- bool Check(int N,int pos){return (bool)(N & (1<<pos));}

- int Set(int N,int pos){ return N=N | (1<<pos);}

- #define getbit(n, i) (((n) & (1LL << (i))) != 0)

- #define setbit0(n, i) ((n) & (~(1LL << (i))))

- #define setbit1(n, i) ((n) | (1LL << (i)))

- #define togglebit(n, i) ((n) ^ (1LL << (i)))

- #define lastone(n) ((n) & (-(n)))

**//Hashing**

const int base = 331;

const int Max = 2e6 + 10;

const int Mod = 1e9 + 7;

const ll Inf = 1LL << 62;

ll pw[Max];

ll Hash[Max];

void pre\_power()

{

pw[0] = 1;

for(int i = 1; i < Max; i++)

{

pw[i] = (pw[i - 1] \* base) % Mod;

}

}

void Hashing(string str, int len)

{

ll hash\_val = 0;

for(int i = 0; i < len; i++)

{

hash\_val = (hash\_val \* base + str[i]) % Mod;

Hash[i + 1] = hash\_val;

}

}

ll SubstringHash(int l, int r)

{

return (Hash[r] - (Hash[l - 1] \* pw[r - l + 1]) % Mod + Mod) % Mod;

}

**Mobius:**

int N = 15;

int mu[N+1];

memset(mu,0,sizeof(mu));

mu[1] = 1;

for(int i = 1; i<=N; ++i)

for(int j = 2\*i; j<=N; j += i)

mu[j] -= mu[i];

for(int i = 1; i<=N; ++i)

printf("%d : %d\n",i,mu[i]);

**//Binary Indexed Tree**

const int Max = 1e5 + 10;

int ar[Max], n;

ll BIT[Max];

void update(int idx, int val)

{

while(idx <= n)

{

BIT[idx] += val;

idx += idx & -idx;

}

}

ll query(int idx)

{

ll ret = 0;

while(idx > 0)

{

ret += BIT[idx];

idx -= idx & -idx;

}

return ret;

}

ll query(int l, int r)

{

return query(r) - query(l - 1);

}

void build()

{

for(int i = 1; i <= n; i++)

{

update(i, ar[i]);

}

}

int main()

{

int q, l, r;

scanf("%d %d", &n, &q);

for(int i = 1; i <= n; i++)

{

scanf("%d", &ar[i]);

}

build();

while(q--)

{

scanf("%d %d", &l, &r);

printf("%lld\n", query(l, r));

}

return 0;

}