**DP ON TREES (VERTEX COVEX)**

#include<bits/stdc++.h>

using namespace std;

const int N = (1e5);

vector<int> G[N+2], tree[N+2];

bool vis[N+2];

void root(int x){

vis[x] = 1;

for(int i=0;i<G[x].size();i++){

if(vis[G[x][i]]) continue;

tree[x].push\_back(G[x][i]); root(G[x][i]);

}

}

int memo[N+2][2];

int dp(int x,bool tome){

if(memo[x][tome] != -1) return memo[x][tome];

int &ans = memo[x][tome] = 0;

if(tome){

for(int i=0;i<tree[x].size();i++){

ans += min(dp(tree[x][i],0) , 1 + dp(tree[x][i],1));

}

}else{

ans += tree[x].size();

for(int i=0;i<tree[x].size();i++){

ans += dp(tree[x][i],1);

}

}

return ans;

}

int main(){

int n;cin>>n;

int a,b;

memset(memo,-1,sizeof memo);

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

cin>>a>>b;

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

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

}

root(1);

cout<<min(dp(1,0),1+dp(1,1))<<'\n';

return 0;

}

**BIT + BINARY SEARCH**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int M = (1e6);

int N=1;

struct BIT{

ll tree[M+1];

BIT(){

for(int i=0;i<=M;i++) tree[i] = 0;

}

void Clear(){

for(int i=0;i<=4\*N;i++) tree[i] = 0;

}

ll Query(int i){

ll sum = 0;

while(i > 0){

sum += tree[i];

i -= ( i & -i );

}

return sum;

}

void Update(int i,ll val){

while(i <= N){

tree[i] += val;

i += (i & -i);

}

}

} FT;

int T[262144];

void update(int l,int r){

l += N;

r += N;

while(l<r){

if(l&1){

T[l++]++;

}

if(r&1){

T[--r]++;

}

l >>= 1;

r >>= 1;

}

}

int query(int x){

x += N;

int ans = 0;

while(x){

ans += T[x];

x >>= 1;

}

return ans;

}

void clear(int n){

for(int i=1;i<N+n;i++) T[i] = 0;

}

int main(){

int t;cin>>t;

int n;

while(t--){

cin>>n;

N = 1;

while(N < n+1) N<<=1;

FT.Clear();

clear(n+1);

ll num;

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

cin>>num;

FT.Update(i,num);

}

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

//DERECHA

int lo=i,hi=n+1;

ll val = FT.Query(i)-FT.Query(i-1);

while((hi-lo)>1){

int mi = (hi+lo)/2;

ll suma = FT.Query(mi-1)-FT.Query(i);

if( suma > val) hi=mi;

else lo=mi;

}

update(i+1,hi);

//IZQUIERDA

lo=0,hi=i;

while((hi-lo)>1){

int mi = (hi+lo)/2;

if(FT.Query(i)-FT.Query(mi)>2\*val) lo=mi;

else hi=mi;

}

update(hi,i);

}

for(int i=1;i<=n;i++)cout<<query(i)<<(char)(i==n?10:32);

}

return 0;

}

**MO’S ALGORITHM WITH DEQUE**

#include <bits/stdc++.h>

using namespace std;

const int N = 111111;

const int BLOCK = 333; // ~sqrt(N)

int a[N], ans[N];

int answer[N];

deque<pair<int,int> > D;

struct node {

int L, R, i, k;

}q[N];

bool cmp(node x, node y) {

if(x.L/BLOCK != y.L/BLOCK) {

// different blocks, so sort by block.

return x.L/BLOCK < y.L/BLOCK;

}

// same block, so sort by R value

return x.R < y.R;

}

void add1(int position) {

if(D.size()){

if(D[0].second != a[position] ) D.push\_front(make\_pair(1,a[position]));

else D[0].first++;

}

else D.push\_front(make\_pair(1,a[position]));

answer[D[0].first-1]++;

}

void remove1(int position) {

if(D[0].first==1){

D.pop\_front();

answer[0]--;

}

else{

answer[D[0].first-1]--;

D[0].first--;

}

}

void add2(int position) {

if(D.size()){

if(D[D.size()-1].second != a[position] ) D.push\_back(make\_pair(1,a[position]));

else D[D.size()-1].first++;

}else{

D.push\_back(make\_pair(1,a[position]));

}

answer[D[D.size()-1].first-1]++;

}

void remove2(int position) {

if(D[D.size()-1].first==1){

D.pop\_back();

answer[0]--;

}

else{

answer[D[D.size()-1].first-1]--;

D[D.size()-1].first--;

}

}

int main() {

int t;

int n,m;

scanf("%d",&t);

while(t--){

D.clear();

memset(answer,0,sizeof(answer));

memset(a,0,sizeof(a));

scanf("%d", &n);

scanf("%d", &m);

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

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

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

scanf("%d%d%d", &q[i].L, &q[i].R , &q[i].k);

q[i].L--; q[i].R--;q[i].k--;

q[i].i = i;

}

sort(q, q + m, cmp);

int currentL = 0, currentR = 0;

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

int L = q[i].L, R = q[i].R;

while(currentR <= R) {

add2(currentR);

currentR++;

}

while(currentR > R+1) {

remove2(currentR-1);

currentR--;

}

while(currentL < L) {

remove1(currentL);

currentL++;

}

while(currentL > L) {

add1(currentL-1);

currentL--;

}

ans[q[i].i] = answer[q[i].k];

}

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

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

}

}

**BINARY TREE**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e6);

int n,m;

vector<ll> C[N+2];

struct hijos{

vector<ll> der,izq,acumDer,acumIzq;

};

hijos memo[N+2];

void pre(){

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

memo[i].der.push\_back(0);

memo[i].izq.push\_back(0);

if(2\*i <= n){

for(int j=0;j<memo[2\*i].izq.size();j++) memo[i].izq.push\_back(memo[2\*i].izq[j]+C[i][0]);

for(int j=1;j<memo[2\*i].der.size();j++) memo[i].izq.push\_back(memo[2\*i].der[j]+C[i][0]);

}

if(2\*i+1 <= n){

for(int j=0;j<memo[2\*i+1].der.size();j++) memo[i].der.push\_back(memo[2\*i+1].der[j]+C[i][1]);

for(int j=1;j<memo[2\*i+1].izq.size();j++) memo[i].der.push\_back(memo[2\*i+1].izq[j]+C[i][1]);

}

sort(memo[i].der.begin(),memo[i].der.end());

sort(memo[i].izq.begin(),memo[i].izq.end());

}

}

void acumular(){

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

int tam = memo[i].der.size();

memo[i].acumDer.resize(tam);

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

if(j==0) memo[i].acumDer[j] = memo[i].der[j];

else memo[i].acumDer[j] = memo[i].der[j] + memo[i].acumDer[j-1];

}

tam = memo[i].izq.size();

memo[i].acumIzq.resize(tam);

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

if(j==0) memo[i].acumIzq[j] = memo[i].izq[j];

else memo[i].acumIzq[j] = memo[i].izq[j] + memo[i].acumIzq[j-1];

}

}

}

ll f(int nodo,bool ok,ll x){

if(ok){

int lo=0,hi=memo[nodo].izq.size();

while((hi-lo)>1){

int mi = (lo+hi)/2;

if(memo[nodo].izq[mi] <= x) lo=mi;

else hi=mi;

}

return (x\*(hi) - memo[nodo].acumIzq[lo]);

}else{

int lo=0,hi=memo[nodo].der.size();

while((hi-lo)>1){

int mi = (lo+hi)/2;

if(memo[nodo].der[mi] <= x) lo=mi;

else hi=mi;

}

return (x\*(hi) - memo[nodo].acumDer[lo]);

}

}

int main(){

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

cin.tie(NULL);

cin>>n>>m;

int num;

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

cin>>num;

C[(i+1)/2].push\_back(num);

}

pre();

acumular();

int nodo;

ll L;

while(m--){

cin>>nodo>>L;

ll ans = f(nodo,0,L) + f(nodo,1,L)-L;

while(nodo>1){

int dir = (nodo%2);

nodo /= 2;

L -= C[nodo][dir];

if(L < 0) break;

ans += f(nodo,dir,L);

}

cout<<ans<<'\n';

}

return 0;

}

**BINARY SEARCH TREE (BST)**

#include<bits/stdc++.h>

using namespace std;

struct Node{

int val;

int der,izq;

Node(){

der = -1;

izq = -1;

}

Node(int \_val,int \_der,int \_izq){

val = \_val;

der = \_der;

izq = \_izq;

}

};

int nodo = 1;

Node T[2005];

void add(int x,int y){

int last = 0;

while(1){

if(x > T[last].val){

if(T[last].der==-1){

T[last].der = y;

T[y] = Node(x,-1,-1);

return;

}else{

last = T[last].der;

}

}else{

if(T[last].izq==-1){

T[last].izq = y;

T[y] = Node(x,-1,-1);

return;

}else{

last = T[last].izq;

}

}

}

}

int peso[2005];

bool vis[2005];

int dp(int x){

if(peso[x]!=-1) return peso[x];

int &ans = peso[x] = 0;

if(T[x].izq!=-1) ans = max(ans,dp(T[x].izq)+1);

if(T[x].der!=-1) ans = max(ans,dp(T[x].der)+1);

return ans;

}

int main(){

memset(peso,-1,sizeof peso);

int n;cin>>n;

int num;cin>>num;

vis[num]=1;

T[0] = Node(num,-1,-1);

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

cin>>num;

if(vis[num]) continue;

vis[num] = 1;

add(num,nodo++);

}

int maxi = dp(0);

int ans = 0;

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

ans += peso[i];

}

cout<<maxi<<endl<<ans<<endl;

return 0;

}

**HASHING WITH SAME SET**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll MOD1 = (1e9+7);

const ll MOD2 = (1e9+9);

const int B = 29;

ll hpref[200005];

ll pot[200005];

ll hpref2[200005];

ll pot2[200005];

ll sumpref[200005];

ll multpref[200005];

ll multpref2[200005];

void init(){

pot[0] = 1;

for(int i=1;i<=200000;i++) pot[i]=(pot[i-1]\*B)%MOD1;

pot2[0] = 1;

for(int i=1;i<=200000;i++) pot2[i]=(pot2[i-1]\*B)%MOD2;

}

void getSumMult(string s){

memset(sumpref,0,sizeof sumpref);

memset(multpref,0,sizeof multpref);

memset(multpref2,0,sizeof multpref2);

sumpref[0] = (s[0] - 'a' + 1);

multpref[0] = (s[0] - 'a' + 1);

multpref2[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

sumpref[i] = (sumpref[i-1] + (s[i]-'a'+1))%MOD1;

multpref[i] = (multpref[i-1] \* (s[i]-'a'+1))%MOD1;

multpref2[i] = (multpref2[i-1] \* (s[i]-'a'+1))%MOD2;

}

}

ll POT(ll x,ll y,ll mod){

if(y==0) return 1;

if(y==1) return x;

ll ans = 1;

if(y&1) ans = x;

ll val = POT(x,y/2,mod);

ans \*= val;

ans %= mod;

ans \*= val;

ans %= mod;

return ans;

}

ll inv(ll x,ll mod){

return POT(x,mod-2,mod);

}

ll subsum(int i,int j){

if(i==0) return sumpref[j];

return ((sumpref[j] - sumpref[i-1])%MOD1 + MOD1)%MOD1;

}

ll submult(int i,int j){

if(i==0) return multpref[j];

return (multpref[j] \* inv(multpref[i-1],MOD1))%MOD1;

}

ll submult2(int i,int j){

if(i==0) return multpref2[j];

return (multpref2[j] \* inv(multpref2[i-1],MOD2))%MOD2;

}

void getpref(string s){

memset(hpref,0,sizeof hpref);

hpref[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

hpref[i] = (hpref[i-1]\*B + (s[i]-'a'+1))%MOD1;

}

memset(hpref2,0,sizeof hpref2);

hpref2[0] = (s[0] - 'a' + 1);

for(int i=1;i<s.size();i++){

hpref2[i] = (hpref2[i-1]\*B + (s[i]-'a'+1))%MOD2;

}

}

ll hsub(int i,int j){

if(i==0) return hpref[j];

return ((hpref[j] - hpref[i-1]\*pot[j-i+1])%MOD1 + MOD1)%MOD1;

}

ll hsub2(int i,int j){

if(i==0) return hpref2[j];

return ((hpref2[j] - hpref2[i-1]\*pot2[j-i+1])%MOD2 + MOD2)%MOD2;

}

map<pair<ll,ll>,int> M;

int ans[200005];

int main(){

init();

int t;cin>>t;

string a,b;

while(t--){

cin>>a>>b;

M.clear();

memset(ans,0,sizeof ans);

getpref(b);

getSumMult(b);

ll sum=0,mult=1,mult2=1;

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

sum += (a[i]-'a'+1);

mult \*= (a[i]-'a'+1);

mult %= MOD1;

mult2 \*= (a[i]-'a'+1);

mult2 %= MOD2;

}

for(int i=0;i<=b.size()-a.size();i++){

ll val1 = hsub(i,i+a.size()-1);

ll val2 = hsub2(i,i+a.size()-1);

if(subsum(i,i+a.size()-1)==sum && submult(i,i+a.size()-1)==mult && submult2(i,i+a.size()-1)==mult2){//poseen el mismo conjunto de caracteres

pair<ll,ll> p = make\_pair(val1,val2);

if(M.count(p)){

ans[M[p]]++;

}else{

M[p] = i;

ans[i]++;

}

}

}

int maxi = 0;

vector<int> v;

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

if(ans[i] > maxi){

v.clear();

v.push\_back(i);

maxi = ans[i];

}else if(ans[i] == maxi){

v.push\_back(i);

}

}

if(maxi == 0) cout<<"-1\n";

else{

string res = b.substr(v[0],a.size());

for(int i=1;i<v.size();i++){

res = min(res,b.substr(v[i],a.size()));

}

cout<<res<<'\n';

}

}

return 0;

}

**MEET IN THE MIDLE**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll m,x;

ll v[35];

ll X[1000005];

typedef pair<ll,pair<ll,ll> > tup;

tup extGcd(ll a,ll b){

if(b==0) return make\_pair(a,make\_pair(1,0));

tup ret = extGcd(b,a%b);

return make\_pair(ret.first , make\_pair(ret.second.second, ret.second.first - (a/b)\*ret.second.second));

}

ll inv(ll a,ll n){

tup t= extGcd(a,n);

ll inver=((t.second.first%n) + n)%n;

return inver;

}

ll get(ll t){

ll ans = 0;

ll gcd = \_\_gcd(m,t);

if(x%gcd!=0) return ans;

ll auxm = m;

ll auxx = x;

t/=gcd;

auxm/=gcd;

auxx/=gcd;

ll inversa = (inv(t,auxm)\*auxx)%auxm;

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

ll newval = auxm\*i+inversa;

if(newval >= m) return ans;

ans += X[newval];

}

return ans;

}

int main(){

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

cin.tie(NULL);

int n;

cin>>n>>m>>x;

if(m==1){

cout<<(1<<n)-1<<endl;

return 0;

}

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

if(n==1){

cout<<( v[0]%m==x?”1\n”:”0\n”)

return 0;

}

int val = (n/2);

ll ans = 0;

for(ll mask = 1;mask < (1<<val);mask++){

ll maskarita = 1;

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

if(mask & (1<<j)){

maskarita \*= v[j];

maskarita %= m;

}

}

maskarita %= m;

if(maskarita == x) ans++;

X[maskarita]++;

}

for(ll mask=1;mask<(1<<n-val);mask++){

ll maskarita = 1;

for(int j=0;j<n-val;j++){

if(mask&(1<<j)){

maskarita \*= v[j+val];

maskarita %= m;

}

}

maskarita %= m;

if(maskarita == x) ans++;

ans += get(maskarita);

}

cout<<ans<<'\n';

return 0;

}

**DP – PYTHON 3.x**

import sys

memo = []

def dp(x):

if(memo[x] != -1):

return memo[x]

if x==0:

return 1

if x==1:

return 1

memo[x] = dp(x-1)+2\*dp(x-2)

return memo[x]

for i in range(300):

memo.append(-1)

for line in sys.stdin:

n = int(line)

print(dp(n))

**DP WITH MASK, OPTIMIZATION**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 20;

ll memo[(1<<N)];

ll d[N];

int n;

ll dp(int mask){

if(mask+1 == (1<<n)) return 1;

if(memo[mask] != -1) return memo[mask];

int pos = \_\_builtin\_popcount(mask);//sabemos cuantos 1’s o tareas asignadas

ll &ans = memo[mask] = 0;

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

if(mask & (1<<i)) continue;

if(d[pos] & (1<<i)){

ans += dp(mask + ( 1<<i ));

}

}

return ans;

}

void solve(){

cin>>n;

memset(memo,-1,sizeof memo);

memset(d,0,sizeof d);

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

int num;

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

cin>>num;

if(num==1) d[i] += (1<<j);

}

}

cout<<dp(0)<<'\n';

}

int main(){

int t;cin>>t;

while(t--){

solve();

}

return 0;

}

**LCA WITH SPARSE TABLE AND EULER TOUR**

#include<bits/stdc++.h>

using namespace std;

const int N = (1e5);

int n;

vector<int> G[N+2];

int A[4\*N+2];

bool vis[N+2];

int ST[4\*N+2][25];

int cnt = 0;

int pa[N+2];

void bfs(int x){

int lvl = 0;

pa[x] = lvl++;

queue<int> Q;

Q.push(x);

while(!Q.empty()){

int p = Q.front();

Q.pop();

for(int i=0;i<G[p].size();i++){

int pp = G[p][i];

Q.push(pp);

pa[pp] = lvl++;

}

}

}

void dfs(int x){

vis[x] = 1;

A[cnt++] = pa[x];

for(int i=0;i<G[x].size();i++){

if(vis[G[x][i]]) continue;

dfs(G[x][i]);

A[cnt++] = pa[x];

}

}

int fi[N+2];

int de[N+2];

void build(){

for(int i=0;i<cnt;i++) ST[i][0] = A[i];

for(int j=1;(1<<j)<=cnt;j++){

for(int i=0;i+(1<<j)<=cnt;i++) ST[i][j] = min(ST[i][j-1] , ST[i + (1<<(j-1))][j-1]);

}

}

int query(int l,int r){//[l,r>

int d = r-l;

int lg = 31 - (\_\_builtin\_clz(d));

return min(ST[l][lg],ST[r-(1<<lg)][lg]);

}

int main(){

memset(pa,-1,sizeof pa);

cin>>n;

int num,len;

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

cin>>len;

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

cin>>num;

G[i].push\_back(num);

}

}

bfs(0);

dfs(0);

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

//if(fi[A[i]]!=-1) continue;

fi[A[i]] = i;

}

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

de[fi[i]] = i;

}

build();

int q;cin>>q;

int a,b;

while(q--){

cin>>a>>b;

int ans = query(fi[pa[a]],fi[pa[b]]+1);

cout<<de[fi[ans]]<<'\n';

}

return 0;

}

**LCA CLASSIC**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e5);

ll LCA[N+2][25], D[N+2][25];

int lvl[N+2];//profundidad del nodo

ll G[N+2];//padres

vector<int> GREV[N+2];//hijos

ll C[N+2];

int n;

void dfs(int x,int level){

lvl[x] = level;

for(int i=0;i<GREV[x].size();i++) dfs(GREV[x][i],level+1);

}

void preprocess(){

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

for(int j=0;(1<<j)<n;j++){

LCA[i][j] = -1;

D[i][j] = 0;

}

}

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

LCA[i][0] = G[i];

D[i][0] = C[i];

}

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

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

if(LCA[i][j-1] != -1){

LCA[i][j] = LCA[LCA[i][j-1]][j-1];

D[i][j] = D[i][j-1] + D[LCA[i][j-1]][j-1];

}

}

}

dfs(0,1);

}

void clear(){

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

GREV[i].clear();

G[i] = 0,C[i]=0,lvl[i]=0;

}

}

int lca(int u,int v){

if(lvl[u] < lvl[v]) swap(u,v);

int lg = 31 - (\_\_builtin\_clz(lvl[u]));

for(int i=lg;i>=0;i--){

if(lvl[u] - (1<<i) >= lvl[v]){

u = LCA[u][i];

}

}

if(u==v) return u;

for(int i=lg;i>=0;i--){

if(LCA[u][i] != -1 && LCA[u][i] != LCA[v][i]){

u = LCA[u][i];

v = LCA[v][i];

}

}

return G[u];

}

ll dist(int pa,int hi){

if(pa==hi) return 0;

int sube = lvl[hi] - lvl[pa];

ll ans = 0;

for(int i=0;i<25;i++){

if(sube & (1<<i)){

ans += D[hi][i];

hi = LCA[hi][i];

}

}

return ans;

}

int main(){

while(cin>>n){

if(n==0) break;

clear();

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

cin>>G[i]>>C[i];

GREV[G[i]].push\_back(i);

}

preprocess();

int q;cin>>q;

vector<ll> ans;

while(q--){

int a,b;

cin>>a>>b;

int ancestro = lca(a,b);

ans.push\_back(dist(ancestro,a) + dist(ancestro,b));

}

for(int i=0;i<ans.size();i++) cout<<ans[i]<<(char)(i+1==ans.size()?10:32);

}

return 0;

}

**TRIE**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

int trie[100005][26];

int cnt[100005][26];

int nodo;

void limpiar(){

memset(trie,0,sizeof trie);

memset(cnt,0,sizeof cnt);

nodo = 1;

}

void addWord(string s){

int n = s.size();

int numNodo = 0;

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

cnt[numNodo][s[i]-'A']++;

if(trie[numNodo][s[i]-'A']){

numNodo = trie[numNodo][s[i]-'A'];

}else{

trie[numNodo][s[i]-'A'] = nodo;

numNodo = nodo++;

}

}

}

int cntPref(string s){

int n = s.size();

int numNodo = 0;

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

if(trie[numNodo][s[i]-'A'] && cnt[numNodo][s[i]-'A']){

//cout<<"entre\n";

//cnt[numNodo][s[i]-'A']--;

numNodo = trie[numNodo][s[i]-'A'];

}else return i;

}

return n;

}

int main(){

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

cin.tie(NULL);

int n;

while(cin>>n){

if(n==-1) break;

limpiar();

string s;

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

cin>>s;

addWord(s);

}

int ans = 0;

vector<string> v(n);

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

sort(v.rbegin(),v.rend());

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

ans += cntPref(v[i]);

}

cout<<ans<<'\n';

}

return 0;

}

**BIT – UPDATE IN RANGE**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e5);

ll bit1[N+2],bit2[N+2];

void update(ll bit[], int idx, ll val){

while(idx <= N+1){

bit[idx] += val;

idx += (idx & -idx);

}

}

ll query(ll bit[], int idx){

ll ret = 0;

while(idx){

ret += bit[idx];

idx -= (idx & -idx);

}

return ret;

}

int main(){

int T,N,Q;

cin>>T;

while(T--){

cin>>N>>Q;

memset(bit1,0,sizeof bit1);

memset(bit2,0,sizeof bit2);

int op,l,r;ll v;

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

cin>>op>>l>>r;

if(op == 0){

cin>>v;

update(bit1,l,v); update(bit1,r + 1,-v);

update(bit2,l,-v \* (l - 1)); update(bit2,r + 1,v \* r);

}else{

ll ans = query(bit1,r) \* r + query(bit2,r) - query(bit1,l - 1) \* (l - 1) - query(bit2,l - 1);

cout<<ans<<'\n';

}

}

}

return 0;

}

**SPARSE TABLE**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = (1e5);

const int M = (20);

ll ST[N+2][M+1];

ll A[N+2];

int n;

ll f(ll x,ll y){

    return x+y;

}

void build(){

    for(int i=0;i<n;i++) ST[i][0] = A[i];

    for(int j=1;(1<<j)<=n;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]);

    }

}

ll query(int l,int r){//[l,r>

    //O(1)

    if(l==r) return 0;

    int d = r-l;

    int lg = 31 - (\_\_builtin\_clz(d));

    return f(ST[l][lg],ST[r-(1<<lg)][lg]);

    //O(log(N))

    //int d = r-l;

    ll ans = 0;

    for(int i=0;i<20;i++){

        if((1<<i) & d){

            ans = f(ST[l][i],ans);

            l+=(1<<i);

        }

    }

    return ans;

}

int main(){

    cin>>n;

    for(int i=0;i<n;i++) cin>>A[i];

    build();

    int q;

    cin>>q;

    int l,r;

    while(q--){

        cin>>l>>r;

        l--;

        cout<<query(l,r)<<'\n';

    }

    return 0;

}