|  |  |
| --- | --- |
| [A](https://codeforces.com/gym/329061/problem/A) | [Шифр Вернама](https://codeforces.com/gym/329061/problem/A) |

Расшифрование происходит точно также как и шифрование.

A хоr К хоr К = А

**Решение C++**

#include <stdio.h>

#include <sstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <list>

#include <iomanip>

#include <map>

#include <set>

#include <cmath>

#include <queue>

#include <cassert>

#include <string.h>

using namespace std;

#pragma comment(linker, "/STACK:20000000")

typedef vector<int> vi;

#define sz(a) int((a).size())

#define all(c) (c).begin(),(c).end()

int mas[111];

int main()

{

int n;

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

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

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

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

int t;

scanf("%d",&t);

printf("%c",t^mas[i]);

}

return 0;

}

**Решение Python**

input**()**

x **=** list**(**map**(**int**,**input**().**split**()))**

y **=** list**(**map**(**int**,**input**().**split**()))**

**for** a**,**b **in** zip**(**x**,**y**):**

**print(**chr**(**a**^**b**),** end **=** ''**)**

|  |  |
| --- | --- |
| [B](https://codeforces.com/gym/329061/problem/B) | [Ксорим](https://codeforces.com/gym/329061/problem/B) |

Подумайте о сложении в системе счисления с основанием 2. Пусть, a = 10101 и b = 1001. Операция изменяет биты в числах, поэтому, если первый бит в a равен 1, а первый бит в b равен 1 (как в случае выше), вы можете сделать оба 0, сделав этот бит 1 в x. На самом деле это единственный способ уменьшить полученную сумму, поэтому x = 1 - это ответ выше.

Теперь выводим x = a & b, где & - это поразрядное И. Так, что (a ⊕ (a & b)) + (b ⊕ (a & b)) работает, но есть еще более короткая формула. Попробуйте доказать, что (a ⊕ (a & b)) + (b ⊕ (a & b)) = a ⊕ b, где ⊕ - побитовое исключающее ИЛИ

**Решение С++**

#include<bits/stdc++.h>

**using** **namespace** std**;**

int main**(){**

int t**;**

cin**>>**t**;**

**while(**t**--){**

int a**,**b**;**

cin**>>**a**>>**b**;**

cout**<<(**a**^**b**)<<**endl**;**

**}**

**}**

**Решение Python**

**for** \_ **in** range**(**int**(**input**())):**

a**,**b**=**map**(**int**,**input**().**split**())**

**print(**a**^**b**)**

|  |  |
| --- | --- |
| [C](https://codeforces.com/gym/329061/problem/C) | [Зерги и Оборотни](https://codeforces.com/gym/329061/problem/C) |

Можно было догадаться, что O подразумевалось, как One, а Z – Zero. Все что оставалось – перевести число из двоичной системы счисления в десятичную.

**Решение С++**

#include <iostream>

#include <algorithm>

#include <string>

#define ll long long

**using** **namespace** std**;**

ll n**,** a**,** b**,** c**,** d**,** k**;**

string s**;**

vector**<**ll**>** v1**,** v2**;**

int main**()** **{**

cin **>>** n**;**

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

cin **>>** s**;**

a **\*=** 2**;**

**if** **(**s **==** "O"**)** **{**

a**++;**

**}**

**}**

cout **<<** a**;**

**return** 0**;**

**}**

**Решение Python**

S **=** int**(**input**())**

res **=** 0

**for** i **in** range**(**S**):**

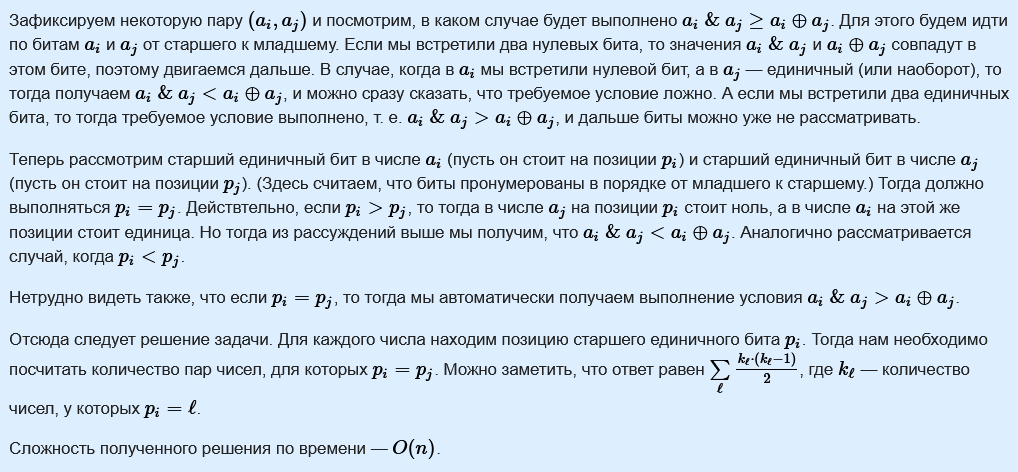
a **=** input**().**strip**()**

res **=** **(**res **<<** 1**)** **|** **(**1 **if** a **==** 'O' **else** 0**)**

**print(**res**)**

|  |  |
| --- | --- |
| [D](https://codeforces.com/gym/329061/problem/D) | [Камень и рычаг](https://codeforces.com/gym/329061/problem/D) |





**Решение С++**

#include<iostream>

#include<vector>

#include<algorithm>

#include<ctime>

#include<random>

**using** **namespace** std**;**

mt19937 rnd**(**time**(NULL));**

int a**[**1000000**+**5**];**

int main**()**

**{**

ios\_base**::**sync\_with\_stdio**(false);**

cin**.**tie**(**0**);**

cout**.**tie**(**0**);**

int t**;**

cin**>>**t**;**

**while** **(**t**--)**

**{**

int n**;**

cin**>>**n**;**

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

**{**

cin**>>**a**[**i**];**

**}**

int64\_t ans**=**0**;**

**for** **(**int j**=**29**;** j**>=**0**;** j**--)**

**{**

int64\_t cnt**=**0**;**

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

**{**

**if** **(**a**[**i**]>=(**1**<<**j**)&&**a**[**i**]<(**1**<<(**j**+**1**)))**

**{**

cnt**++;**

**}**

**}**

ans**+=**cnt**\*(**cnt**-**1**)/**2**;**

**}**

cout**<<**ans**<<**'\n'**;**

**}**

**}**

**Решение Python**

**import** math

**import** sys

input **=** sys**.**stdin**.**readline

**for** nt **in** range**(**int**(**input**())):**

n **=** int**(**input**())**

a **=** list**(**map**(**int**,**input**().**split**()))**

group **=** **[**0**]\***31

**for** i **in** a**:**

x **=** int**(**math**.**log2**(**i**))**

group**[**x**]** **+=** 1

ans **=** 0

**for** i **in** group**:**

ans **+=** **((**i**)\*(**i**-**1**))//**2

**print** **(**ans**)**

|  |  |
| --- | --- |
| [E](https://codeforces.com/gym/329061/problem/E) | [Матрицы конъюнкции](https://codeforces.com/gym/329061/problem/E) |

Во первых заметим, что числа на границе ни от кого не зависят и могут быть любыми (2количество клеток на границе). Также, если на границе (не в углу) есть 0, то он распространяется на все внутри таблицы и задает всю таблицу. Если на границе (угловые не имеют значения) все 1, то внутри может быть как все 0, так и все 1. Таких вариантов 24 (это все варианты угловых клеток)

То есть общее количество 2количество клеток на границе + 24

Частные случаи: когда меньшая сторона <3 и матрица 3\*3

Нужно применять быстрое возведение в степень (также следить за переполнением инт)

#include <stdio.h>

#include <sstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <list>

#include <iomanip>

#include <map>

#include <set>

#include <cmath>

#include <queue>

#include <cassert>

#include <string.h>

using namespace std;

#pragma comment(linker, "/STACK:20000000")

typedef vector<int> vi;

#define sz(a) int((a).size())

#define all(c) (c).begin(),(c).end()

string problem\_name = "a";

void init(){

freopen((problem\_name+".in").c\_str(),"rt",stdin);

// freopen((problem\_name+".out").c\_str(),"wt",stdout);

}

int n,m;

int mod = 1000000007;

long long pow(long long a, int b, int m)

{

long long c=a, r=1;

while (b)

{

if (b&1)

{

r \*= c;

r %= m;

}

b >>= 1;

c \*= c;

c%=m;

}

return r;

}

long long solve(long long n, long long m)

{

if (n>m) swap(n,m);

if (n==1) return pow(2,m,mod);

if (n==2) return pow(2,m\*2,mod);

if (n==3 && m==3) return pow(2,8,mod);

return (pow(2,2\*n+2\*m-4) + 16)% mod;

}

|  |  |
| --- | --- |
| [F](https://codeforces.com/gym/329061/problem/F) | [AND 0, большая сумма](https://codeforces.com/gym/329061/problem/F) |
|  |  |

Начнем с массива, в котором каждый бит в каждом элементе равен 1. Он явно не имеет побитового И равного 0, поэтому для каждого бита нам нужно выключить его (сделать 0) хотя бы в одном из элементов. Однако мы не можем отключить его более чем в одном элементе, так как тогда сумма будет уменьшаться. Поэтому для каждого бита мы должны выбрать ровно один элемент и отключить его там. Поскольку имеется k битов и n элементов, ответ будет просто nk.

**Решение С++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

int n**,**k**;**

const int MOD**=**1e9**+**7**;**

int main**()**

**{**

int t**;**

scanf**(**"%d"**,&**t**);**

**while(**t**--)**

**{**

scanf**(**"%d %d"**,&**n**,&**k**);**

long long ans**=**1**;**

**for(**int i**=**0**;**i**<**k**;**i**++)**

ans**=(**ans**\***n**)%**MOD**;**

printf**(**"%lld\n"**,**ans**);**

**}**

**}**

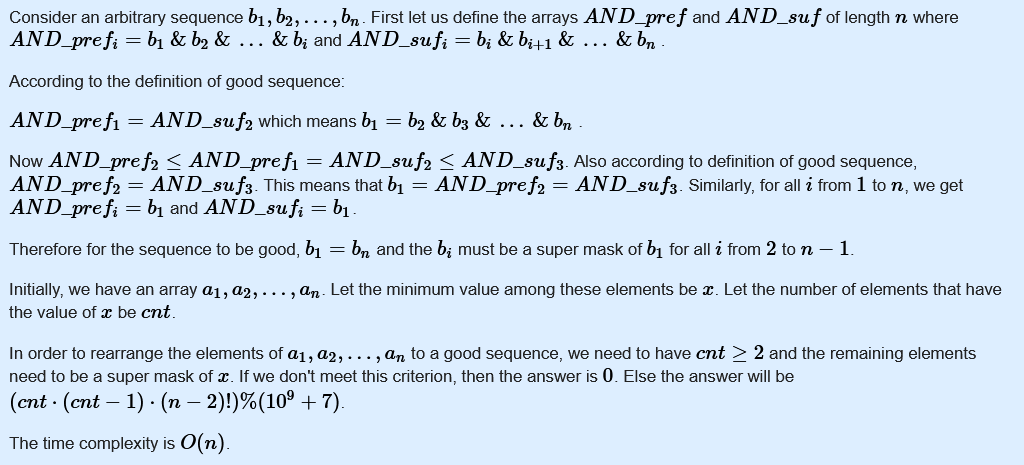
**Решение Python**

**for** f **in** range**(**int**(**input**())):**

n**,**k**=**map**(**int**,**input**().**split**())**

**print(**pow**(**n**,**k**,**10**\*\***9**+**7**))**

|  |  |
| --- | --- |
| [G](https://codeforces.com/gym/329061/problem/G) | [Последовательности И](https://codeforces.com/gym/329061/problem/G) |
|  |  |



**Решение С++**

#include<bits/stdc++.h>

**using** **namespace** std**;**

void solveTestCase**()**

**{**

int MOD**=**1e9**+**7**;**

int n**;**

cin**>>**n**;**

vector**<**int**>** a**(**n**);**

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

int min1**=\***min\_element**(**a**.**begin**(),**a**.**end**());**

int cnt**=**0**;**

**for(**int x**:**a**)**

**{**

**if(**min1**==**x**)**cnt**++;**

**if((**min1**&**x**)!=**min1**)**

**{**

printf**(**"0\n"**);**

**return;**

**}**

**}**

int fact**=**1**;**

**for(**int i**=**1**;**i**<=**n**-**2**;**i**++)**fact**=(**1LL**\***fact**\***i**)%**MOD**;**

int ans**=(**1LL **\*** cnt **\*** **(**cnt**-**1**))%**MOD**;**

ans **=** **(**1LL **\*** ans **\*** fact**)** **%** MOD**;**

printf**(**"%d\n"**,**ans**);**

**}**

int main**()**

**{**

int tests**;**

cin**>>**tests**;**

**while(**tests**--)**

solveTestCase**();**

**return** 0**;**

**}**

**Решение Python**

MOD **=** int**(**1e9**)+**7

**for** \_ **in** range**(**int**(**input**())):**

n **=** int**(**input**())**

a **=** **[\***map**(**int**,**input**().**split**())]**

x **=** a**[**0**]**

**for** i **in** a**[**1**:]:**

x **&=** i

occ **=** a**.**count**(**x**)**

res **=** 1

**for** i **in** range**(**1**,**n**-**1**):**

res**\*=**i

res**%=** MOD

**print((**occ**\*(**occ**-**1**)\***res**)%**MOD**)**

|  |  |
| --- | --- |
| [H](https://codeforces.com/gym/329061/problem/H) | [Псевдопростые числа](https://codeforces.com/gym/329061/problem/H) |

Наверное, первая мысль писать решето эратосфена. Ну или прекальк какой-нибудь.

Но это не проходит по времени и по памяти.

Хотя есть и решето с линейным временем работы

<http://e-maxx.ru/algo/prime_sieve_linear>

А сам массив, кстати, в обычном решете можно ужать в 8 раз, используя биты числа.

Решаем с помощью принципа включений – исключений.

Будем считать количество чисел в интервале, которые делятся хотя бы на одно из данных, а потом вычтем из длины интервала.

Заметим что нам можно рассматривать только простые числа. Так как, если число делится на простое, то и на составное)

Напишем функцию F(X) = числу чисел из интервала от L до R, которые делятся на X

F(X) = R/X – (L-1)/X

Получим все простые числа из отрезка от 0 до N

Решением будет

F(2) + F(3) + F(5) + …

- F(2\*3) - F(2\*5) - F(3\*5) ..

+ F(2\*3\*5) …

Это я думаю понятно считаем сколько чисел делится на 2, потом на 3, но надо вычесть те, которые делятся на 2\*3 их 2 раза посчитали.

То есть надо перебрать все подмножества простых чисел.

Так как их много, то будем использовать перебор с отсечениями, так как можно заметить что F(X) = 0, если X>R

Для 46 простых работает достаточно быстро.

**Вот тут можно почитать про принцип включений и исключений и его применение**

<http://e-maxx.ru/algo/inclusion_exclusion_principle>

#include <stdio.h>

#include <sstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <list>

#include <iomanip>

#include <map>

#include <set>

#include <cmath>

#include <queue>

#include <cassert>

#include <string.h>

using namespace std;

#pragma comment(linker, "/STACK:20000000")

typedef vector<int> vi;

#define sz(a) int((a).size())

#define all(c) (c).begin(),(c).end()

long long calc(long long l, long long r, long long val)

{

return r/val - (l-1)/val;

}

vi pr;

int ispr(int n)

{

if (n<2) return 0;

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

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

return 1;

}

long long res=0,l,r,n;

// номер простого, текущее произведение, сколько чисел в произведении

void go(int pos, long long cur, int col)

{

if (cur>r) return ;

if (pos>=sz(pr))

{

if (cur==1) return;

if (col%2) res+=calc(l,r,cur); else

res-=calc(l,r,cur);

return;

}

go(pos+1,cur,col);

go(pos+1,cur\*pr[pos],col+1);

}

int main()

{

pr.clear();

cin >> n >> l >> r;

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

if (ispr(i))

pr.push\_back(i);

res=0;

go(0,1,0);

cout << r-l+1- res << endl;

return 0;

}

|  |  |
| --- | --- |
| [I](https://codeforces.com/gym/329061/problem/I) | [Кольцевой путь](https://codeforces.com/gym/329061/problem/I) |

Решение в лоб за куб не проходит по времени.

Тут можно вспомнить что возведение матрицы смежности в степень K дает количество путей между каждой парой вершин длины К (с повторными прохождениями по вершинам). Но тут бы норм было так К=3, но по времени мы же не будем писать возведение в степень быстрее чем за куб)

Тут в общем есть читы на битах.

Перебираем пару вершин, соединенных ребром.

Далее нужно быстро сравнить их списки смежности –по считать сколько единиц на общих позициях. Это число прибавить к обеим вершинам.

Сожмем матрицу смежности в 8 раз по памяти. Будем использовать биты инта (проще все же использовать встроенный тип **bitset**).

unsigned int mas[N][N/32 + 1]

Далее достаточно будет для каждой пары вершин, соединенных ребром пробежаться по всем битовым числам, применить операцию И. После чего посчитать количество единиц в числе. При чем это надо сделать быстро. Используем предпросчет. Посчитаем количество единиц во всех числах от 0 до 65535 и за 2 операции будем считать количество бит в числе.

int res = cnt[val & 0xFFFF] + cnt[val >> 16]

**Заметим, что val обязательно должен быть беззнаковым**

Потому что если у нас стоит бит в стершем (31 – начиная с 0) разряде, то это считается отрицательным числом. И при сдвиге его вправа левая часть будет заполняться единицами, а не нулями.

Ну либо писать так

int res = cnt[val & 0xFFFF] + cnt[(val >> 16) & 0xFFFF]

#include <stdio.h>

#include <sstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <list>

#include <iomanip>

#include <map>

#include <set>

#include <cmath>

#include <queue>

#include <cassert>

#include <string.h>

#include <time.h>

#include <fstream>

using namespace std;

#pragma comment(linker, "/STACK:50000000")

typedef vector<int> vi;

#define sz(a) int((a).size())

#define all(c) (c).begin(),(c).end()

const int mx= 3000;

int col;

unsigned int mas[mx+3][mx/32 + 10];

int ocnt[1<<16];

int res[mx+2];

int n;

void fill()

{

char s[mx+10];

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

assert(n>=1 && n<=2000);

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

gets(s);

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

{

if (i==j) assert(s[j]=='0');

assert(s[j]=='0' || s[j]=='1');

if (s[j]=='1')

mas[i][j>>5]|=1LL<<(j&31);

}

}

}

int calc()

{

for (int i=0;i<1<<16;i++)

{

int c=0;

int t=i;

while (t)

{

c++;

t&=t-1;

}

ocnt[i]=c;

}

unsigned int r=0;

col = n/32;

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

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

if (mas[i][j>>5]&(1LL<<(j&31)))

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

r=mas[i][k]&mas[j][k];

r = ocnt[r&0xFFFF] + ocnt[r>>16];

res[i]+=r;

res[j]+=r;

}

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

printf("%d ",res[i]);

return 0;

}

void stupid();

void solve()

{

memset(mas,0,sizeof(mas));

memset(res,0,sizeof(res));

memset(ocnt,0,sizeof(ocnt));

n=0;

col=0;

fill();

calc();

}

**C использованием bitset**

#include <iostream>

#include <algorithm>

#include <vector>

#include <map>

#include <set>

#include <string>

#include <iomanip>

#include <bitset>

#define ll long long

#define mod 1000000007

**using** **namespace** std**;**

ll n**,** a**,** b**,** c**,** d**,** k**,** t**[**2000**];**

string s**;**

vector**<**ll**>** v**;**

char aa**;**

bitset**<**2000**>** m**[**2000**];**

int main**()** **{**

cin **>>** n**;**

aa **=** getchar**();**

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

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

aa **=** getchar**();**

**if** **(**aa **==** '0'**)** **{**

m**[**i**][**j**]** **=** **false;**

**}**

**else** **{**

m**[**i**][**j**]** **=** **true;**

**}**

**}**

aa **=** getchar**();**

**}**

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

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

**if** **(**m**[**i**][**j**])** **{**

bitset**<**2000**>** bt **=** m**[**i**]** **&** m**[**j**];**

t**[**i**]** **+=** bt**.**count**();**

t**[**j**]** **+=** bt**.**count**();**

**}**

**}**

**}**

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

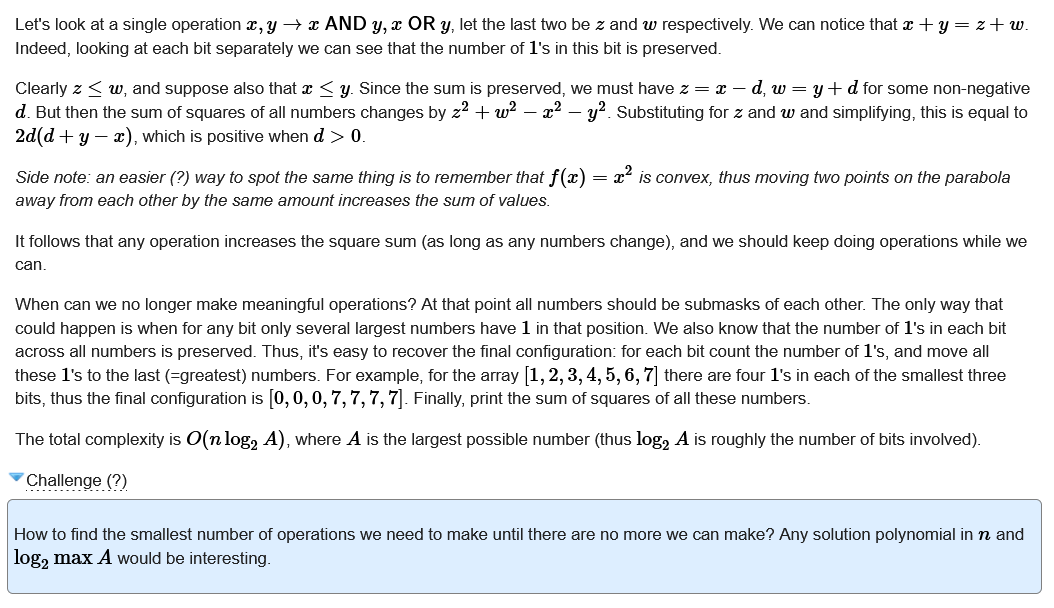
cout **<<** t**[**i**]** **<<** " "**;**

**}**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [J](https://codeforces.com/gym/329061/problem/J) | [AND, OR и сумма квадратов](https://codeforces.com/gym/329061/problem/J) |



**Решение С++**

#include <bits/stdc++.h>

#define inf (ll)(1e16)

**using** **namespace** std**;**

**typedef** long long ll**;**

int sum**[**20**];**

int main**()** **{**

int n**;**

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

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

int x**;**

scanf**(**"%d"**,&**x**);**

**for(**int j**=**0**;**j**<**20**;**j**++)**

**if** **((**x**>>**j**)&**1**)** sum**[**j**]++;**

**}**

ll ans**=**0**;**

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

int x**=**0**;**

**for(**int j**=**19**;**j**>=**0**;**j**--)**

x**=(**x**<<**1**)|(**sum**[**j**]>=**i**);**

ans**+=(**ll**)**x**\***x**;**

**}**

printf**(**"%lld\n"**,**ans**);**

**return** 0**;**

**}**

**Решение Python**

n **=** int**(**input**())**

a **=** list**(**map**(**int**,**input**().**split**()))**

ls **=** **[**0**]\***20

**for** i **in** a**:**

**for** j **in** range**(**20**):**

**if** i**&**1**<<**j**:**

ls**[**j**]** **+=** 1

ans **=** 0

**for** i **in** range**(**n**):**

x **=** 0

**for** j **in** range**(**20**):**

**if** ls**[**j**]:**

x **+=** 1**<<**j

ls**[**j**]** **-=** 1

ans **+=** x**\*\***2

**print(**ans**)**

|  |  |
| --- | --- |
| [K](https://codeforces.com/gym/329061/problem/K) | [XOR-угадайка](https://codeforces.com/gym/329061/problem/K) |

Пусть все числа, которые мы отправили в каком-то запросе, имеют одинаковые значения в *k*-м бите. Тогда вне зависимости от того, какой *i* выберет проверяющая программа, мы всегда можем распознать, чему равен *k*-й бит в *x*.

Это приводит нас к простому решению: поделим 14 бит числа *x* на две группы по 7 бит. В первом запросе отправим 100 чисел с одинаковыми значениями во всех битах из первой группы, и найдем значения этих битов в *x*. Во втором запросе сделаем все то же самое для второй группы. Будьте аккуратными, чтобы не отправить одно и то же число дважды.

**Решение С++**

#include<bits/stdc++.h>

**using** **namespace** std**;**

int main**()**

**{**

cout **<<** "?"**;**

**for(**int i **=** 1**;** i **<=** 100**;** i**++)**

cout **<<** " " **<<** i**;**

cout **<<** endl**;**

cout**.**flush**();**

int res1**;**

cin **>>** res1**;**

cout **<<** "?"**;**

**for(**int i **=** 1**;** i **<=** 100**;** i**++)**

cout **<<** " " **<<** **(**i **<<** 7**);**

cout **<<** endl**;**

cout**.**flush**();**

int res2**;**

cin **>>** res2**;**

int x **=** 0**;**

x **|=** **(**res1 **&** **(((**1 **<<** 7**)** **-** 1**)** **<<** 7**));**

x **|=** **(**res2 **&** **((**1 **<<** 7**)** **-** 1**));**

cout **<<** "! " **<<** x **<<** endl**;**

cout**.**flush**();**

**return** 0**;**

**}**

**Решение Python**

**from** sys **import** stdout

**print(**'?'**,** **\*(**i **for** i **in** range**(**1**,** 101**)))**

stdout**.**flush**()**

x **=** int**(**input**())**

**print(**'?'**,** **\*(**i **<<** 7 **for** i **in** range**(**1**,** 101**)))**

stdout**.**flush**()**

y **=** int**(**input**())**

**print(**'!'**,** **(**x **&** 0x3f80**)** **|** **(**y **&** 0x007f**))**