TỔNG VÀ TỔNG XOR

Nhắc lại phép toán **AND,OR,XOR**

AND: trả về True khi và chỉ khi cả hai toán hạng là True.

011100010

&

010101111

= 010100010

OR: trả về True khi và chỉ khi ít nhất một toán hạng là True.

011100010

|

010101111

= 011101111

XOR: trả về True khi và chỉ khi hai toán hạng có giá trị khác nhau.

011100010

^

010101111

= 001001101

Cho hai số nguyên s và x.

Yêu cầu:

Bạn hãy đếm có bao nhiêu cặp số nguyên dương a và b sao cho tổng và tổng XOR của chúng lần lượt bằng s và x.

Mô tả đầu vào

Dòng duy nhất chứa hai số nguyên $s(2 \le s \le 10^{12})$ và $x(0 \le x \le 10^{12})$.

Mô tả đầu ra In ra số nguyên là kết quả của bài toán.

Lý thuyết liên quan đến bài toán

a + b = s

a ⊕ b = x

trong đại số nhị phân ta có đẳng thức:

a + b = (a ⊕ b) + 2(a & b)

đặt c = a & b => c = $\frac{s-x}{2}$

⇨ Điều kiện để tồn tại c:

1. S > = x
2. s – x là chẵn
3. c & x = 0 (Nếu tại bit i ta có ci=1, nghĩa là ai=bi=1. Nhưng khi đó ai⊕bi=0.

   Vậy không thể có cùng lúc ci=1 **và** xi=1.)

Nếu bất kỳ điều kiện nào thất bại, kết quả = 0 cặp.

⇨ **Từ c và x đến đếm số cách chọn bit cho a, b**

Khi thoả điều kiện, ta đã “tách” bài toán ra:

- Mỗi bit i của a,b phải thỏa:
  - Nếu xi=1: bits khác nhau ⇒ hai lựa chọn (ai,bi)=(1,0) hoặc (0,1).
  - Nếu xi=0: bits bằng nhau. Lại chia:

- Nếu ci=1: thì (ai,bi)=(1,1) **chỉ 1 cách**.
- Nếu ci==0: thì (ai,bi)=(0,0) **chỉ 1 cách**.

1001 (9)

0101 (5)

C =0001

X =1100

2 * 2 *1*1 = 4

Vì các bit độc lập lẫn nhau, tổng số cách để chọn toàn bộ chuỗi bit của (a,b) là tích số cách ở mỗi bit.

- Chỉ có những bit i với xi=1 mới cho hệ số nhân là 2;
- Mọi bit khác góp hệ số 1.

Do đó tổng số cặp (a,b) ban đầu là

2 ^(số bit 1 trong x)

Loại bỏ các cặp không hợp lệ (dương)

Khi c=0 tức là s−x=0hay s= x, phương trình:

$$\begin{cases} a+b=s, \\ a \oplus b = x \end{cases} \implies \begin{cases} a+b=x, \\ a \oplus b = x, \end{cases}$$

⇨ x=x+2(a&b) ⟹ a & b=0.

Như vậy mọi bit của a và b không được có cả hai cùng là 1 — tức là chúng “chia” các bit 1 của x ra cho nhau mà không chồng lên.

Giả sử x có k bit 1. Mỗi bit 1 đó có thể thuộc về a hoặc b (nhưng không cả hai). Vậy tổng cộng có

2 mũ k

cách phân chia. Trong đó có **2** trường hợp đặc biệt:

1. Tất cả k bit 1 được gán cho a, thì a= x và b=0.

2. Tất cả k bit 1 được gán cho b, thì a=0 và b =x.

Do a và b là nguyên dương nên 2 th trên bị loại => cách chọn cặp a, b nếu c = 0 là

(2 mũ k) -2

Code:

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

int main(){
  ios::sync_with_stdio(false);
  cin.tie(nullptr);

  ll s, x;
  cin >> s >> x;

//   kiểm tra s phải lớn hơn x và s - x phải chẵn nếu sai thì số cặp = 0
  if (s < x || ((s - x) % 2 !=0)) {
    cout << 0 << "\n";
    return 0;
  }

  ll c = (s - x) >> 1;
//   kiểm tra C and X phải  = 0 nếu sai thì số cặp = 0
  if ((c & x) != 0) {
    cout << 0 << "\n";
    return 0;
  }
```

// có thể dùng hàm đếm bit 1 hoặc tự tạo hàm bằng cách dùng phép toán and số đó với 1 kiểm tra trả về 1 thì temmp +=1, sau đó dùng phép dịch bit số đó đc số mới ...

```
ll ans =pow( __builtin_popcountll(x), 2) ;

//   loại bỏ 2 TH đặc biệt khi c = 0

   if (c == 0) ans -= 2;


   cout << ans << "\n";

   return 0;

}
```

HASH 1

(*Si* − '*a*' + 1) chuẩn hóa các ký tự từ 96 => 1 trăm mấy trong bảng mã ASCII về thành từ 1 đến 26

HASH 2

Thay vì hash từ vị trí 0 đến hết chuỗi, thì ta có thể mã hóa từ L => R, và có thể truy vẫn Q truy vẫn các đoạn từ L => R khác nhau

Ý tưởng:

Đặt

$$G[i] = \sum_{k=1}^{i} v_k \, 31^{\,k-1},$$

trong đó $v_k = S[k] -' a' + 1.$

– Khi đó:

- $G[1] = v_1 \cdot 31^0$
- $G[2] = v_1 \cdot 31^0 + v_2 \cdot 31^1$
- ...
- $G[i] = v_1 \cdot 31^0 + v_2 \cdot 31^1 + \cdots + v_i \cdot 31^{\,i-1}.$

Muốn hash đoạn [L,R] với mũ từ 0 đến R−L

Chúng ta cần:

$$\sum_{k=L}^{R} v_k \times 31^{k-L},$$

tức:

- Khi $k = L$ → exponent $k - L = 0$.
- Khi $k = L + 1$ → exponent 1.
- …
- Khi $k = R$ → exponent $R - L$.

Tách từ G[R]

Ta có

$$G[R] = \sum_{k=1}^{R} v_k\, 31^{k-1} = \Big(\sum_{k=1}^{L-1} v_k\, 31^{k-1}\Big) + \Big(\sum_{k=L}^{R} v_k\, 31^{k-1}\Big).$$

Phần thứ hai là

$$\sum_{k=L}^{R} v_k\, 31^{k-1} = \sum_{k=L}^{R} v_k\, 31^{(k-L)} \times 31^{L-1}.$$

Như vậy nếu ta chỉ làm

$$G[R] - G[L-1] = \sum_{k=L}^{R} v_k\, 31^{k-1},$$

thì kết quả vẫn mang hệ số chung 31^{L-1}.

Bỏ hệ số “thừa”

Muốn về đúng dạng “mũ” từ 0 đến R−L”, ta phải chia cho 31^l-1:

$$\frac{G[R] - G[L-1]}{31^{L-1}} = \sum_{k=L}^{R} v_k \, 31^{k-L}.$$

Đó chính là hash với “mũ tăng dần” từ 0 cho đến R-L