TỔNG XOR

Cho dãy số nguyên dương $A = (a_1, a_2, \dots, a_n)$, ta gọi tổng XOR của nó là

$$a_1 \text{ XOR } a_2 \text{ XOR } \dots \text{ XOR } a_n$$

Yêu cầu: Chọn một dãy con của A sao cho tổng XOR của dãy con này là lớn nhất.

Dữ liệu: Vào từ file văn bản SXOR.INP

- Dòng 1 chứa số nguyên dương $n \leq 10^5$
- Dòng 2 chứa n số nguyên dương $a_1, a_2, \dots, a_n$ $(\forall i: a_i \leq 10^{18})$

Kết quả: Ghi ra file văn bản SXOR.OUT **chỉ số** những phần tử được chọn trên một dòng cách nhau bởi dấu cách

Ví dụ

SXOR.INP	SXOR.OUT
4 1 2 4 5	4 2
5 14 8 13 6 10	1 3 4 5

Giải thích ví dụ 1: $a[4]$ XOR $a[2] = 7$ là tổng XOR lớn nhất có thể chọn được

Thuật toán

Một trong những tính chất quan trọng của phép XOR đó là

$$x \text{ XOR } x = 0$$

$$y \text{ XOR } x \text{ XOR } x = y$$

Tức là nếu trong tổng XOR có hai số giống nhau, ta có thể bỏ hai số đó đi mà không ảnh hưởng tới kết quả.

Nhận xét 1

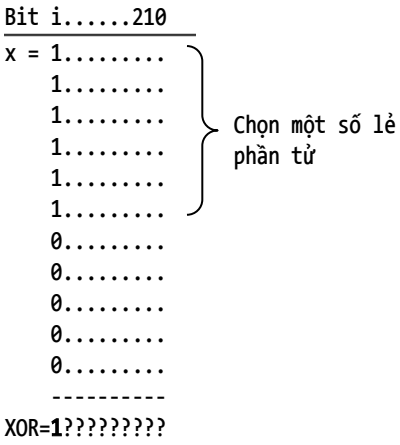
Tạo dãy A' từ dãy A như sau: a_1 giữ nguyên, thay thế một số phần tử a_i khác bởi $a'_i := a_i \text{ XOR } a_1$. Khi đó mọi phương án chọn trên dãy A đều có phương án chọn cho tổng XOR tương đương trên dãy A'

Chứng minh:

Với một phương án chọn trong A, ta chọn nguyên những vị trí đó trong A'. Tổng XOR của hai phương án chọn trên hai dãy sẽ khác nhau một số lần XOR a_1. Nếu số lần này chẵn, đây là cách chọn tương đương. Nếu số lần này lẻ, trong phương án chọn trên A', ta đảo trạng thái chọn/không chọn của phần tử a_1 để được cách chọn cho tổng XOR tương đương trên A. Điều ngược lại chứng minh tương tự.

Nhận xét 2

Vì các số $a_i \leq 10^{18} < 2^{60}$, ta có thể biểu diễn mỗi phần tử $\in A$ bởi dãy bit đánh số từ bit 0 là bit đơn vị đến bit $i \leq 59$. Xét những phần tử của A có bit i là 1. Không giảm tính tổng quát, ta coi như tồn tại ít nhất một phần tử $x \in A$ có bit i là 1 và tổng XOR lớn nhất chắc chắn có bit i là 1 (tệ nhất là ta chọn duy nhất số x)



Từ nhận xét 1, ta có thể giữ nguyên x và đặt tất cả các phần tử y khác có bit i là 1 trong A bởi $y \coloneqq y$ XOR x

Dãy mới giờ đây chỉ có duy nhất phần tử x có bit i là 1 nên nó chắc chắn được chọn. Ta chọn x và vấn đề lặp lại với việc cố gắng bật bit $k-1$ của tổng XOR lên thành 1 (nếu nó đang là 0)...

Cài đặt

Nếu chỉ để tính tổng XOR lớn nhất, ta có thể dùng đoạn chương trình ngắn gọn sau:

```
res := 0;
for i := maxBit downto 0 do //maxBit = 59, cố gắng bật bit i của res lên thành 1
  begin
    «Tìm x có bit i là 1»;
    if «không tìm thấy» then
      Continue;
    if «bit i của res là 0» then
      res := res XOR x; //Chọn x sẽ bật được bit i của kết quả lên thành 1
    for j := 1 to n do
      if «bit i của a[j] là 1» then
        a[j] := a[j] XOR x; //Chú ý là ta đã chọn x, nên phần tử = x sau lệnh XOR này sẽ = 0, ko bao giờ chọn nữa
    end;
Output ← res;
```

Truy vết

Vì đề bài yêu cầu nói rõ chọn phần tử nào nên ta cần có cơ chế đánh dấu chọn và lưu vết hợp lý. Việc tìm x trong mỗi vòng for i cần phải chỉ ra được phần tử $a[k]$ nào bằng x. Ngoài ra khi đặt $a[j] \coloneqq a[j]$ XOR $a[k]$, ta phải lưu lại thông tin để biết rằng $a[j]$ nó được sinh ra từ $a[j]$ và $a[k]$ của bước trước qua phép XOR.

Ta dùng hai bảng

- $selected[0 \dots maxBit, 1 \dots n]$: $selected[i,k] = True$ nếu phần tử $a[k]$ được chọn vào tổng XOR bước lặp xét bit i.
- $trace[0 \dots maxBit, 1 \dots n]$: $trace[i,j] = k$ nếu phần tử $a[j]$ tại bước xét bit i được sinh ra từ phép XOR của chính nó với phần tử $a[k]$ tại bước trước (bước $i+1$).

```
res := 0;
selected[0…maxBit, 1…n] := False; trace[0…maxBit, 1…n] := 0;
for i := maxBit downto 0 do //maxBit = 59, cố gắng bật bit i của res lên thành 1
  begin
    «Tìm a[k] = x có bit i là 1»;
    if «không tìm thấy» then
      Continue;
    if «bit i của res là 0» then
      begin
        res := res XOR x; //Chọn x sẽ bật được bit i của kết quả lên thành 1
        selected[i, k] := True; //Đánh dấu chọn a[k] tại bước i
      end;
    if i = 0 then Break; //Đã xét đến bit đơn vị
    for j := 1 to n do
      if «bit i của a[j] là 1» then
        begin
          a[j] := a[j] XOR x; //Chú ý là ta đã chọn x, nên phần tử = x sau lệnh XOR này sẽ = 0, ko bao giờ chọn nữa
          trace[i - 1, j] := k; //a[j] tại bước sau sinh ra từ a[j] xor a[k] tại bước này
        end;
    end;
```

Vấn đề cuối cùng là chỉ ra những phần tử nguyên thủy nào được chọn. Để ý rằng nếu $k = trace[i, j]$ thì việc chọn $a[j]$ tại bước sau tương đương với chọn $a[j]$ và $a[k]$ tại bước trước. Ngoài ra, nếu một phần tử bị chọn một số chẵn lần, ta coi như không chọn. Lần ngược quá trình tính toán để xác định phần tử nào được chọn ở bước đầu tiên (bước xét $i = maxBit$)

```
for i := 0 to maxBit - 1 do
    for j := 1 to n do
      if selected[i, j] then //a[j] được chọn tại bước i
        begin
          k := trace[i, j]; //Xem nó được sinh ra từ 2 pt a[j] và a[k] nào ở bước trước
          selected[i + 1, j] := not selected[i + 1, j]; //Đảo trạng thái chọn/không chọn ở bước trước
          if k <> 0 then //k <> 0: a[j] không giống như a[j] ở bước trước mà được sinh ra qua phép XOR vơi a[k]
            selected[i + 1, k] := not selected[i + 1, k];
        end;
  for j := 1 to n do
    if selected[maxBit, j] then Output ← Chọn a[j];
```

Source code:

```
{$MODE OBJFPC}
program MaximumXOR;
const
  InputFile  = 'SXOR.INP';
  OutputFile = 'SXOR.OUT';
  maxN = Round(1E5);
  maxV = Round(1E18);
  maxBit = 59;
var
  fi, fo: TextFile;
  a: array[1..maxN] of Int64;
  trace: array[0..maxBit, 1..maxN] of Integer;
  selected: array[0..maxBit, 1..maxN] of Boolean;
  n: Integer;
  res: Int64;

procedure Enter;
var
  i: Integer;
begin
  ReadLn(fi, n);
  for i := 1 to n do Read(fi, a[i]);
end;
```

```
function Bit(const value: Int64; id: Integer): Integer; //Bit id của value là 0 hay 1?
begin
  Result := value shr id and 1;
end;

function Find1(id: Integer): Integer; //Tìm phần tử trong A có bit id là 1
var
  i: Integer;
begin
  for i := 1 to n do
    if Bit(a[i], id) = 1 then
      Exit(i);
  Result := -1;
end;

procedure Solve;
var
  i, j, k: Integer;
  x: Int64;
begin
  res := 0;
  FillChar(trace, SizeOf(trace), 0);
  FillChar(selected, SizeOf(selected), False);
  for i := maxBit downto 0 do
    begin
      k := Find1(i);
      if k = -1 then Continue; //Không có phần tử nào có bit i là 1
      x := a[k];
      if Bit(res, i) = 0 then //bit i của res đang là 0
        begin
          selected[i, k] := True;
          res := res xor x; //chọn x sẽ bật bit i của res lên thành 1
        end;
      if i = 0 then Break; //Đã xét qua bit đơn vị, giải tán
      for j := 1 to n do
        if Bit(a[j], i) = 1 then
          begin
            a[j] := a[j] xor x;
            trace[i - 1, j] := k; //a[j] bước sau sinh ra từ a[j] xor a[k] tại bước này
          end;
    end;
end;

procedure DoTrace;
var
  i, j, k: Integer;
begin
  for i := 0 to maxBit - 1 do
    for j := 1 to n do
      if selected[i, j] then
        begin
          k := trace[i, j];
          selected[i + 1, j] := not selected[i + 1, j];
          if k <> 0 then
            selected[i + 1, k] := not selected[i + 1, k];
        end;
  for j := 1 to n do
    if selected[maxBit, j] then Write(fo, j, ' ');
end;

begin
```

```
  AssignFile(fi, InputFile); Reset(fi);
  AssignFile(fo, OutputFile); Rewrite(fo);
  try
    Enter;
    Solve;
    DoTrace;
  finally
    CloseFile(fi); CloseFile(fo);
  end;
end.
```