XOR key

HackerRank

Xorq has invented an encryption algorithm which uses bitwise XOR operations extensively. This encryption algorithm uses a sequence of non-negative integers $x = [x[1], x[2] \cdots x[n]]$ as its key. To implement this algorithm efficiently, *Xorq* needs to find maximum value of $(a \oplus x_j)$ for given integers a, l and r, such that, $l \leqslant j \leqslant r$. Help *Xorq* implement this function.

For example, $x = [3, 5, 9]$, $a = 4$, $l = 1$ and $r = 3$. We test each $x[j]$ for all values of j between l and r inclusive:

```
j   x[j]     x[j]^4
1   3        7
2   5        1
3   9        13
```

Our maximum value is 13.

Function Description

Complete the *xorKey* function in the editor below. It should return an integer array where each value is the response to a query.

xorKey has the following parameters:

- *x*: a list of integers
- *queries*: a two dimensional array where each element is an integer array that consists of $a[i], l[i], r[i]$ for the i^{th} query at indices $0, 1$ and 2 respectively.

Input Format

The first line contains an integer t, the number of test cases.
The first line of each test case contains two space-separated integers n and q, the size of the integer array x and the number of queries against the test case.
The next line contains n space-separated integers $x[j]$.
Each of next q lines describes a query which consists of three integers $a[i],\ l[i]$ and $r[i]$.

Constraints

$1 \leq n \leq 100000$
$1 \leq q \leq 50000$
$0 \leq x[j], a[i] \leq 2^{15}$
$1 \leq l[i], r[i] \leq n$

Output Format

For each query, print the maximum value for $(a[i] \oplus x[j])$, such that, $l[i] \leqslant j \leqslant r[i]$ on a new line.

Sample Input 0

```
1
15 8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
10 6 10
1023 7 7
33 5 8
182 5 10
181 1 13
5 10 15
99 8 9
33 10 14
```

Sample Output 0

```
13
1016
41
191
191
15
107
47
```

Explanation 0

- First Query (10 6 10): $x_6 \oplus 10 = 12, x_7 \oplus 10 = 13, x_8 \oplus 10 = 2, x_9 \oplus 10 = 3, x_{10} \oplus 10 = 0$. The maximum is 13.
- Second Query (1023 7 7): $x_7 \oplus 1023 = 1016$
- Third Query (33 5 8): $x_5 \oplus 33 = 36, x_6 \oplus 33 = 39, x_7 \oplus 33 = 38, x_8 \oplus 33 = 41$
- Fourth Query (182 5 10): $x_5 \oplus 182 = 179, x_6 \oplus 182 = 176, x_7 \oplus 182 = 177, x_8 \oplus 182 = 190, x_9 \oplus 182 = 191, x_{10} \oplus 182 = 188$