Given an array $[a_1, a_2, \dots, a_n]$

We are to answer q queries. In each query for given X we should print $\max\limits_{i \in \{1..n\}} (a_i \oplus X)$ and $\min\limits_{i \in \{1..n\}} (a_i \oplus X)$

To solve this problem we'll consider the following data structure — **binary trie**

So the idea is too insert all a_i into trie in binary format. For simplicity and briefness I implemented trie using c++ vector.

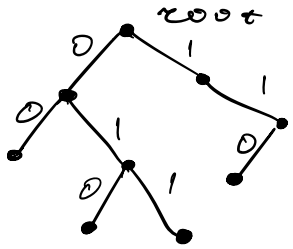
`vector<vector<int>> t = {{-1, -1}}`

↑ root's initialization

Each node contains vector with 2 elements:

$t[i][0]$ - the index of the left son (0)

$t[i][1]$ - the index of the right son (1)



Let LOG be $\max\limits_{i \in \{1..n\}} (\log_2 a_i)$

```
const int LOG = 20;

void insert(int x) {
    int v = 0; - the current vertex (starting from the root)
    for (int i = LOG; i >= 0; i--) {
        int j = (x >> i) & 1; i-th Bit of X
        if (t[v][j] == -1) { if this vertex is not initialized
            t[v][j] = sz(t); set the index of new son
            t.push_back({ -1, -1 }); create new vertex
        }
        v = t[v][j]; one step down
    }
}
```

One important notice. It's necessary to start from LOG and end with 0. In the other case we won't be able to find the answer in "get" functions.

```
void get_max(int x) {
    int v = 0;
    for (int i = LOG; i >= 0; i--) {
        int j = (x >> i) & 1;
        if (t[v][j ^ 1] != -1) {
            j ^= 1;
        }
        x ^= (j << i);
        v = t[v][j];
    }
    return x;
}
```

here the idea is that if for i-th bit of x - $\bar{j}$, exists a_i, which i-th bit is $\bar{j}$ then we'll go to its branch (simple greedy approach)

Consider 2 numbers A and B.

$a_n a_{n-1} \dots a_0$ - A in binary format

$b_m b_{m-1} \dots b_0$ - B in binary format

$x_k x_{k-1} \dots x_0$ - X in binary format

] a_i is the most significant bit that $a_i \neq x_i$ and b_j similarly.

Then $A \oplus x \geq B \oplus x \iff i \geq j$

And that's why we build binary trie from more significant bits, because in the other case we wouldn't have

such condition to choose where to go on i-th step.

get_min function is the same