Codeforces Round #197 (Div. 2)

Problema D: *Xenia and Bit Operations*

Prof. Edson Alves – UnB/FCTE

Problema

Xenia the beginner programmer has a sequence a, consisting of 2^n non-negative integers: $a_1, a_2, \ldots, a_{2^n}$. Xenia is currently studying bit operations. To better understand how they work, Xenia decided to calculate some value v for a.

Namely, it takes several iterations to calculate value v. At the first iteration, Xenia writes a new sequence a_1 or a_2, a_3 or $a_4, \ldots, a_{2^n-1}$ or a_{2^n}, consisting of 2^{n-1} elements. In other words, she writes down the bit-wise OR of adjacent elements of sequence a. At the second iteration, Xenia writes the bitwise exclusive OR of adjacent elements of the sequence obtained after the first iteration. At the third iteration Xenia writes the bitwise OR of the adjacent elements of the sequence obtained after the second iteration. And so on; the operations of bitwise exclusive OR and bitwise OR alternate. In the end, she obtains a sequence consisting of one element, and that element is v.

Problema

Let's consider an example. Suppose that sequence $a = (1, 2, 3, 4)$. Then let's write down all the transformations $(1, 2, 3, 4) \rightarrow (1 \text{ or } 2 = 3, 3 \text{ or } 4 = 7) \rightarrow (3 \text{ xor } 7 = 4)$. The result is $v = 4$.

You are given Xenia's initial sequence. But to calculate value v for a given sequence would be too easy, so you are given additional m queries. Each query is a pair of integers p, b. Query p, b means that you need to perform the assignment $a_p = b$. After each query, you need to print the new value v for the new sequence a.

Entrada e saída

Input

The first line contains two integers n and m $(1 \leq n \leq 17, 1 \leq m \leq 10^5)$. The next line contains 2^n integers $a_1, a_2, \ldots, a^{2n} (0 \leq a_i < 2^{30})$. Each of the next m lines contains queries. The i-th line contains integers $p_i, b_i (1 \leq p_i \leq 2^n, 0 \leq b_i < 2^{30})$ – the i-th query.

Output

Print m integers – the i-th integer denotes value v for sequence a after the i-th query.

Exemplo de entradas e saídas

Sample Input

```
2 4
1 6 3 5
1 4
3 4
1 2
1 2
```

Sample Output

```
1
3
3
3
```

Solução com complexidade $O(M \log N + N)$

- O problema consiste em um vetor cujo tamanho é uma potência de 2, cujas operações são a atualização de um elemento em particular ou uma consulta (*query*) em todo intervalo de índices $[1, 2^N]$
- Assim, ele pode ser solucionado de forma eficiente através do uso de uma árvore de segmentos com implementação *bottom-up*
- A alternância entre as duas operações (OR e XOR) pode ser feita por meio de ponteiros para funções e *swaps*
- Esta solução tem complexidade $O(M \log N + N)$

Solução com complexidade $O(M \log N + N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using Op = int (*)(int, int);

class SegmentTree {
    int N;
    std::vector<int> ns;

public:
    SegmentTree(const std::vector<int>& xs) : N(xs.size()), ns(2*N) {
        std::copy(xs.begin(), xs.end(), ns.begin() + N);

        int k = N;
        Op op = [](int x, int y) { return x | y; };
        Op next = [](int x, int y) { return x ^ y; };

        while (k >>= 1) {
            for (int i = k; i < 2*k; ++i)
                ns[i] = op(ns[2*i], ns[2*i + 1]);
```

Solução com complexidade $O(M \log N + N)$

```
                swap(op, next);
            }
        }

        int update(int i, int value)
        {
            int a = i + N - 1;
            ns[a] = value;

            Op op = [](int x, int y) { return x | y; };
            Op next = [](int x, int y) { return x ^ y; };

            while (a >>= 1) {
                ns[a] = op(ns[2*a], ns[2*a + 1]);
                swap(op, next);
            }

            return ns[1];
        }
    };
```

Solução com complexidade $O(M \log N + N)$

```
int main()
{
    ios::sync_with_stdio(false);

    int n, m;
    cin >> n >> m;

    vector<int> xs(1 << n);

    for (int i = 0; i < (1 << n); ++i)
        cin >> xs[i];
```

Solução com complexidade $O(M \log N + N)$

```
    SegmentTree tree(xs);

    while (m--)
    {
        int p, b;
        cin >> p >> b;

        cout << tree.update(p, b) << '\n';
    }

    return 0;
}
```