Fast XORting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given an integer n which is a power of two and a permutation $a_1, a_2, \ldots, a_n$ of $0, 1, \ldots, n-1$. In one operation you can do one of the following:

- Swap two adjacent elements. That is, choose any $1 \le i \le n-1$, and swap a_i, a_{i+1}
- Choose any integer $0 \le x \le n-1$, and replace a_i with a_i `XOR` x for every $1 \le i \le n$ (notice that the array remains a permutation)

What is the minimal number of operations needed to sort the permutation?

Here `XOR` denotes the bitwise XOR operation.

Input

The first line of the input contains a single integer n ($1 \le n \le 2^{18}$, n is a power of two) — the length of the permutation.

The second line contains n integers a_1, a_2, ... , a_n which form a permutation of 0, 1, ..., $n-1$.

Output

Output a single integer — the smallest number of operations needed to sort this permutation.

Examples

standard input	standard output
8 0 1 3 2 5 4 7 6	2
8 2 0 1 3 4 5 6 7	2

Note

In the first sample, we can sort the permutation with two operations as follows:

1. Swap a_1, a_2. The permutation becomes $[1, 0, 3, 2, 5, 4, 7, 6]$.
2. Choose $x = 1$, and `XOR` all elements with 1. It will become $[0, 1, 2, 3, 4, 5, 6, 7]$.

In the second sample, we can sort the permutation with two operations as follows:

1. Swap a_1, a_2. The permutation becomes $[0, 2, 1, 3, 4, 5, 6, 7]$.
2. Swap a_2, a_3. It will become $[0, 1, 2, 3, 4, 5, 6, 7]$.