Problem A. Linearization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Bitwise "and" of two non-negative integers is calculated as follows: write both numbers in binary, then the i-th binary digit of the result is equal to 1 if both arguments have the i-th digit equal to 1. For example, $(14 \text{ and } 7) = (1110_2 \text{ and } 0111_2) = 110_2 = 6$.

"Exclusive or" (xor) of two binary digits equals 1 if they are unequal, and 0 if they are equal. Thus, $0 \text{ xor } 0 = 0$, $0 \text{ xor } 1 = 1$, $1 \text{ xor } 0 = 1$ and $1 \text{ xor } 1 = 0$.

Parity function $P(x)$ for a non-negative integer x equals 1 if the binary notation of x has odd number of ones, and 0 if the binary notation of x has even number of ones. For example, $P(5) = P(101_2) = 0$, $P(7) = P(111_2) = 1$.

Consider a binary string whose length is a power of two: $s = s_0 s_1 \ldots s_{n-1}$, where $n = 2^k$. We will call this string *linear*, if there is an integer x, $0 \le x < n$, and a binary digit b, such that for all i from 0 to $n-1$ holds $s_i = P(i \text{ and } x) \text{ xor } b$.

For example, a string "`1100`" is linear: take $x = 2 = 10_2$ and $b = 1$.

- $s_0 = P(0 \text{ and } 2) \text{ xor } 1 = P(0) \text{ xor } 1 = 0 \text{ xor } 1 = 1$
- $s_1 = P(1 \text{ and } 2) \text{ xor } 1 = P(0) \text{ xor } 1 = 0 \text{ xor } 1 = 1$
- $s_2 = P(2 \text{ and } 2) \text{ xor } 1 = P(2) \text{ xor } 1 = 1 \text{ xor } 1 = 0$
- $s_3 = P(3 \text{ and } 2) \text{ xor } 1 = P(2) \text{ xor } 1 = 1 \text{ xor } 1 = 0$

Meanwhile, "`0001`" is not linear: whatever x we chose, we would have $P(0 \text{ and } x) = P(0) = 0$, therefore $b = 0$. We have $0 = P(1 \text{ and } x)$ and $0 = P(2 \text{ and } x)$, therefore $x = 0$. But $P(3 \text{ and } 0) = 0 \ne s_3 = 1$.

Consider a binary string. In one action you can take a continuous segment of digits and invert them: change all zeros to ones and vice versa. Call *hardness of linearization* of this string the minimal number of actions one needs to make it linear.

For example, the hardness of linearization for the string "`0001`" is 1: you can invert the left three digits to get the string "`1111`" which is linear with $x = 0$, $b = 1$. There are other ways to linearize it in one action.

You are given a string t and q queries (l_i, r_i). For each query, consider a substring of t from l_i-th digit to r_i-th digit, inclusive. Digits of t are numbered from left to right, starting with 0. It is guaranteed that the length of each query is a power of two. Calculate the hardness of linearization for every given substring.

Input

The first line of input contains a single integer m — the length of the string t ($1 \le m \le 200\,000$). The second line contains a binary string t of length m.

The next line contains integer q — the number of queries ($1 \le q \le 200\,000$). Each of the next q lines contains two integers, l_i and r_i ($0 \le l_i \le r_i < m$, $r_i - l_i + 1 \ge 2$, substring length is a power of two).

Output

For each query, print one integer: the hardness of linearization of the corresponding substring of t.

Example

standard input	standard output
8 00000101 3 0 7 2 5 0 3	2 1 0

Note

In the first query we need to linearize the whole string. This can be done, for example, by inverting the segment from 4-th to 6-th digit, getting the string "00001011", and then inverting the 5-th digit, getting "00001111" which is linear with $x = 4$ and $b = 0$.

In the second query, the string "0001'' can be linearized in one action, as described in the problem statement.

In the third query the string "0000" is already linear with $x = 0$, $b = 0$.