Problem H. Rebellious Sequences

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

A bracket sequence is a sequence of opening and closing brackets. A bracket sequence is called *balanced* if and only if two conditions hold:

- it contains equal number of opening and closing brackets;
- for each prefix, the number of closing brackets is not greater than the number of opening brackets.

A bracket sequence is called *rebellious* if it contains equal number of opening and closing brackets and is **not** balanced.

You are given a balanced bracket sequence of n brackets and q queries to it. Each query asks you to swap two brackets. It is guaranteed that after each query the sequence stays balanced.

After each query you should find whether it is possible to split the sequence into several disjoint rebellious subsequences.

Formally, let the original sequence be $s_1 s_2 \dots s_n$. You have to check if there exists a number $k > 1$ (the number of subsequences) and k non-empty sequences of indices $(i_{1,1} < \dots < i_{1,l_1}), \dots, (i_{k,1} < \dots < i_{k,l_k})$ such that for each $1 \le j \le k$ the bracket sequence $s_{i_{j,1}} s_{i_{j,2}} \dots s_{i_{j,l_j}}$ is rebellious and every integer from 1 to n appears among those sequences exactly once.

Input

The first line of the input contains two numbers n and q ($1 \le n, q \le 300\,000$), the length of the sequence and the number of queries respectively. In the second line, there is a string of length n consisting of characters '`(`' and '`)`'.

Then follow q lines that describe the queries. Each of them contains two integers i and j ($1 \le i < j \le n$), denoting that you should swap the brackets on i-th and j-th position.

It is guaranteed that the sequence is balanced initially and remains balanced after each query.

Output

After each query print "`Yes`", if it is possible to split the sequence into several disjoint rebellious subsequences, and "`No`" otherwise.

Example

standard input	standard output
8 4 (()()()) 3 4 5 6 2 7 6 7	No No Yes No