E. New Year and Old Subsequence

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A string t is called *nice* if a string "2017" occurs in t as a **subsequence** but a string "2016" doesn't occur in t as a **sub sequence**. For example, strings "203434107" and "9220617" are nice, while strings "20016", "1234" and "20167" aren't nice.

The *ugliness* of a string is the minimum possible number of characters to remove, in order to obtain a nice string. If it's impossible to make a string nice by removing characters, its ugliness is -1.

Limak has a string s of length n, with characters indexed 1 through n. He asks you q queries. In the i-th query you should compute and print the ugliness of a **substring** (continuous subsequence) of s starting at the index a_i and ending at the index b_i (inclusive).

Input

The first line of the input contains two integers n and q ($4 \le n \le 200\,000$, $1 \le q \le 200\,000$) — the length of the string s and the number of queries respectively.

The second line contains a string s of length n. Every character is one of digits '0'–'9'.

The i-th of next q lines contains two integers a_i and b_i ($1 \le a_i \le b_i \le n$), describing a substring in the i-th query.

Output

For each query print the ugliness of the given substring.

Examples

input
8 3 20166766 1 8 1 7 2 8

output
4 3 -1

input
15 5 012016662091670 3 4 1 14 4 15 1 13 10 15

output
-1 2 1 -1 -1

input

```
4 2
1234
2 4
1 2
```

output

```
-1
-1
```

Note

In the first sample:

- In the first query, $ugliness(\texttt{"20166766"}) = 4$ because all four sixes must be removed.
- In the second query, $ugliness(\texttt{"2016676"}) = 3$ because all three sixes must be removed.
- In the third query, $ugliness(\texttt{"0166766"}) = -1$ because it's impossible to remove some digits to get a nice string.

In the second sample:

- In the second query, $ugliness(\texttt{"01201666209167"}) = 2$. It's optimal to remove the first digit '2' and the last digit '6', what gives a string "010166620917", which is nice.
- In the third query, $ugliness(\texttt{"016662091670"}) = 1$. It's optimal to remove the last digit '6', what gives a nice string "01666209170".