Lucky Numbers

It's a well known fact that in some cultures the number **13** brings bad luck.

You are given a number **X** consisting of **N** digits. You are to compute how many numbers smaller than or equal to **X** don't contain **13** as a substring in their base 10 representation. Since the answer can be quite large, you are to print it modulo **1 000 000 007**.

In addition, you are to process **Q** queries of two possible types:
(1) **Query(radixL, radixR)**: you are to compute how many numbers smaller than or equal to **substr(X, radixL, radixR)** don't contain **13** as a substring in their base 10 representation. Since the answer can be quite large, you are to print it modulo **1 000 000 007**. **substr(X, L, R)** stands for the substring of **X** starting from the **L**-th digit and ending with the **R**-th digit counting from left to right;
(2) **Update(radix, newDigit)**: one of **X**'s digits is replaced. In particular, digit numbered **radix** counting from left to right is changed to **newDigit**.

Note that number **X** is 1-indexed.
Note that number **X** and **substr(X, l, r)** may have leading zeros.

Input

The first line of input contains two integers **N** and **Q** (**1 ≤ N ≤ 100 000, 0 ≤ Q ≤ 10 000**) - the number of digits of number **X** and the number of queries that need to be processed.

The second line of input contains the number **X**.

The next **Q** lines describe the queries that need to be processed. Each line starts with an integer **t** (**1 ≤ t ≤ 2**) - the type of query that needs to be processed.

If **t = 1**, then the line describes a **Query** and two integers **radixL** and **radixR** (**1 ≤ radixL ≤ radixR ≤ N**) follow - the left and right ends of the substring that you need to consider as bounds for the query.

Otherwise (t = 2), the line describes an **Update** and two integers **radix** and **newDigit** (**1 ≤ radix ≤ N, 0 ≤ newDigit ≤ 9**) follow - the position of the digit that needs to be changed and the new value of the digit.

Output

The first line of the output contains a single integer - how many numbers smaller than or equal to **X** don't contain **13** as a substring in their base 10 representation. Since the answer can be quite large, you are to print it modulo **1 000 000 007**.

Then, for each query of the first type, print the answer modulo **1 000 000 007** on a separate line.

Subtasks

(1) 1 ≤ N ≤ 6, Q = 0 (14 points)
(2) 1 ≤ N ≤ 18, Q = 0 (14 points)
(3) 1 ≤ N ≤ 10 000, 0 ≤ Q ≤ 10 000, all queries will be of the first type **(9 points)**
(4) 1 ≤ N ≤ 100 000, 0 ≤ Q ≤ 10 000, all queries will be of the first type **(27 points)**
(5) 1 ≤ N ≤ 10 000, 0 ≤ Q ≤ 10 000 (9 points)
(6) 1 ≤ N ≤ 100 000, 0 ≤ Q ≤ 10 000 (27 points)

Example(s)

Standard Input	**Standard Output**

6 10
560484
2 6 4
2 1 4
2 5 6
2 6 1
2 3 6
1 3 6
1 1 3
1 6 6
1 2 6
2 1 7
	528145
	6228
	452
	2
	63454

Explanation:
There are 528145 non-negative integers smaller than or equal to 560484 not containing digits 13 as a substring in their base 10 representation. Note that this includes the number 0.

X is initially 560484.
After update "2 6 4", **X** becomes 56048**4**.

After update "2 1 4", **X** becomes **4**60484.
After update "2 5 6", **X** becomes 4604**6**4.
After update "2 6 1", **X** becomes 46046**1**.
After update "2 3 6", **X** becomes 46**6**461.
Query "1 3 6" asks us how many non-negative integers smaller than or equal to ~~46~~**6461** don't contain substring 13 - there are 6228 such numbers.
Query "1 1 3" asks us how many non-negative integers smaller than or equal to **466**~~461~~ not containing substring 13 - there are 452 such numbers.
Query "1 6 6" asks us how many non-negative integers smaller than or equal to ~~46646~~**1** not containing substring 13 - there are 2 such numbers: 0 and 1.
Query "1 2 6" asks us how many non-negative integers smaller than or equal to ~~4~~**66461** not containing substring 13 - there are 63454 such numbers.
After update "2 1 7", **X** becomes **7**66461.