E. Maze 2D

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The last product of the R2 company in the 2D games' field is a new revolutionary algorithm of searching for the shortest path in a $2 \times n$ maze.

Imagine a maze that looks like a $2 \times n$ rectangle, divided into unit squares. Each unit square is either an empty cell or an obstacle. In one unit of time, a person can move from an empty cell of the maze to any side-adjacent empty cell. The shortest path problem is formulated as follows. Given two free maze cells, you need to determine the minimum time required to go from one cell to the other.

Unfortunately, the developed algorithm works well for only one request for finding the shortest path, in practice such requests occur quite often. You, as the chief R2 programmer, are commissioned to optimize the algorithm to find the shortest path. Write a program that will effectively respond to multiple requests to find the shortest path in a $2 \times n$ maze.

Input

The first line contains two integers, n and m ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 2 \cdot 10^5$) — the width of the maze and the number of queries, correspondingly. Next two lines contain the maze. Each line contains n characters, each character equals either '.' (empty cell), or 'X' (obstacle).

Each of the next m lines contains two integers v_i and u_i ($1 \le v_i, u_i \le 2n$) — the description of the i-th request. Numbers v_i, u_i mean that you need to print the value of the shortest path from the cell of the maze number v_i to the cell number u_i. We assume that the cells of the first line of the maze are numbered from 1 to n, from left to right, and the cells of the second line are numbered from $n + 1$ to $2n$ from left to right. It is guaranteed that both given cells are empty.

Output

Print m lines. In the i-th line print the answer to the i-th request — either the size of the shortest path or -1, if we can't reach the second cell from the first one.

Examples

input
4 7 .X.. ...X 5 1 1 3 7 7 1 4 6 1 4 7 5 7

output
1 4 0 5 2 2 2

input
10 3 X...X..X.. ..X...X..X

```
11 7
7 18
18 10
```

output

```
9
-1
3
```