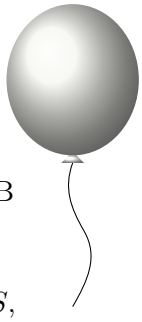
H The Real Folk Blues



Time limit: 1.0s
Memory limit: 1024MB

You are slicing into a Red Dragon Syndicate data terminal. The terminal's archive, denoted as S, initially contains n encrypted signal frequencies, each represented as a binary string of length k. These initial frequencies are indexed from 1 to n in the exact order they are extracted from the terminal's memory. To breach the core, you must synthesize new frequencies and inject them into the archive. Each synthesis operation appends exactly one new binary string to S, automatically assigning it the next available integer index.

You are equipped with two operations:

- **Phase Inversion**: Select an existing frequency s and append its exact *bitwise complement*[1];
- **Signal Triangulation**: Select three existing frequencies u, v, and w (not necessarily distinct) and append their *bitwise majority*, denoted as $\text{maj}(u, v, w)$. For every bit position i, the operation evaluates as:
$$\text{maj}(u, v, w)_i = \text{maj}(u_i, v_i, w_i).$$
For individual bits a, b, c, $\text{maj}(a, b, c)$ is 1 if at least two of them are 1, and 0 otherwise.

Your objective is to figure out if it is possible to forge a specific target bounty code t of length k. If it is possible, you need to provide a sequence of operations (at most 10^5) that successfully builds it.

Input

The first line of the terminal feed contains two integers n and k ($1 \le n, k \le 200$), representing the number of initial codes and the length of each code.

Each of the following n lines contains a binary string (`0`s and `1`s) of length k, showing a code currently in the archive.

The final line contains a single binary string t of length k, representing the target bounty code you need to forge.

Output

If it is impossible to forge the target code t, print a single line containing `NO`.

Otherwise, print `YES`. On the next line, print an integer m ($0 \le m \le 10^5$), representing the total number of operations you will use. Then, print m lines describing your operations in order:

- `1 x`: Pick the existing code at index x and apply Phase Inversion (append bitwise complement of x);
- `2 x y z`: Pick the existing codes at indices x, y, and z and apply Signal Triangulation (append maj of the existing strings with indices x, y, and z).

[1] Let s be a binary string of length k, such that $s = s_1 s_2 \dots s_k$ where each bit $s_i \in \{0, 1\}$. The bitwise complement of s, often denoted mathematically as $\neg s$, is defined as a new binary string of length k where the value of the i-th bit is exactly $1 - s_i$.

Every index you use must already exist in the archive at the moment you use it. After all m operations, at least one code in the archive must perfectly match your target t. If the target code t is already in the starting archive, you can just output $m = 0$.

If there are multiple correct ways to forge the code, you may print any valid sequence of operations.

Samples

Sample input 1	Sample output 1
3 4 1000 0100 0010 1111	YES 4 1 1 1 2 1 3 2 4 5 6

Explanation of sample 1.

- The first three operations append the complements of the initial strings, creating `0111`, `1011`, and `1101`.
- The final operation takes the bitwise majority of these three strings.
- In every position at least two of them have bit 1, so the appended string is `1111`, which is the target.