GONDOR

For every spark K we remember the time index t(K) when the first arrow hits it. Initially we set:

$$t(K) = \begin{cases} 0, & \text{for } K = 1 \\ \infty, & \text{for } K > 1 \end{cases}$$

In each step of the algorithm we pick the next spark that will get lit and mark it as such. Then we use the archer's instructions, simulating the launch of an arrow by refreshing t(K) if the new arrow reaches spark K before any other.

MAJMUN

Note that it suffices to know how many times the monkey jumped in each of the 8 directions to find its final positoin. Let $prefix_K(S)$ be the total number of jumps the monkey would make in direction S if we gave him only the first K commands. Similarly, let $suffix_K(S)$ mean the same for the last K commands.

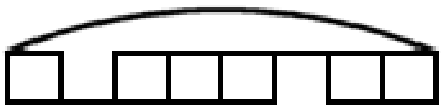
The values of the sequences $prefix_K$ and $suffix_K$ can be easily calculated from the values $prefix_{K-1}$, $suffix_{K-1}$ and the K-th key pressed..

Now changing the K-th key can be quickly processed using the sequences $prefix_{K-1}$, $suffix_{N-K+1}$ and the new key press, taking into account the change of direction in the K-th command. That change of direction manifests in rotating the 8 numbers in the sequence $suffix_{N-K+1}$.

JEZERO

We will describe a combinatoric solution; an approach using dynamic programming is also possible.

Note first that we can remove all fountains whose bridges are not available (these cannot influence the number of walks). After removing, we get a special structure as in the image below (vertical line segments are bridges):



The structure consists of several components, each component of several collections of squares, connected by single paths. The above image illustrates a single component with 3 collections of squares, with 2, 4 and 3 bridges, respectively.

Walks can be enumerated independently for each component. Inside a single component, each walk matches one of the following two patterns:

1. The entire walk is inside a single collection of squares. If the collection has n bridges ($n-1$ squares), then the number of walks is $n \cdot (n-1)/2$.
2. The walk passes through all collections of squares, using all paths that connect adjacent collections (such a walk exists only if the collections form a cycle and in that case there is only one component). For such a walk we know where it enters and exits every collection. If a component consists of K collections of sizes in order $n_1, n_2, \ldots, n_K$, then the number of such walks is $B(n_1) \cdot B(n_2) \cdot \ldots \cdot B(n_K)$. Here $B(n_i) = 2^{n_i - 1}$ is the number of ways such a walk can traverse the i-th collection.

The total number of walks in one component is the sum of walks of type 1 over all components, and the number of walks of type 2. If, after removing fountains in the first step of the algorithm, there is only one collection with exactly 3 available paths incident to each fountain, then the total number of walks must be multiplied by 2.