That's the important part that you have to keep in mind. Using hashing will not be 100% deterministically correct, because two complete different strings might have the same hash (the hashes collide). However, in a wide majority of tasks this can be safely ignored as the probability of the hashes of two different strings colliding is still very small. And we will discuss some techniques in this article how to keep the probability of collisions very low.

Calculation of the hash of a string

The good and widely used way to define the hash of a string s of length n is

$$\begin{aligned} \text{hash}(s) &= s[0] + s[1] \cdot p + s[2] \cdot p^2 + ... + s[n-1] \cdot p^{n-} \\ &= \sum_{i=0}^{n-1} s[i] \cdot p^i \mod m, \end{aligned}$$

where p and m are some chosen, positive numbers. It is called a **polynomial rolling hash function**.

It is reasonable to make p a prime number roughly equal to the number of characters in the input alphabet. For example, if the input is composed of only lowercase letters of English alphabet, $p = 31$ is a good choice. If the input may contain both uppercase and lowercase