#### **Rabin Karp**

有一种最坏时间复杂度也为O(N) 的算法。思路是这样的，先生成窗口内子串的哈希码，然后再跟 needle 字符串的哈希码做比较。

这个思路有一个问题需要解决，如何在常数时间生成子串的哈希码？

滚动哈希：常数时间生成哈希码

生成一个长度为 L 数组的哈希码，需要O(L) 时间。

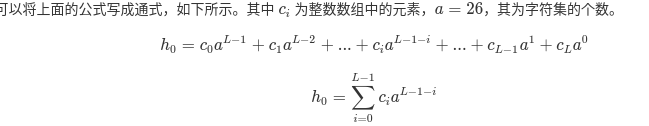
如何在常数时间生成滑动窗口数组的哈希码？利用滑动窗口的特性，每次滑动都有一个元素进，一个出。

由于只会出现小写的英文字母，因此可以将字符串转化成值为 0 到 25 的整数数组： arr[i] = (int)S.charAt(i) - (int)'a'。按照这种规则，abcd 整数数组形式就是 [0, 1, 2, 3]，转换公式如下所示。

h0 = 0\*26^3 + 1 \* 26^2 + 2 \* 26^1 + 3 \* 26^0

可以将上面的公式写成通式，如下所示。其中 ci 为整数数组中的元素，a=26，其为字符集的个数。

h0 = c0 \*a^{L - 1} + c1\* a^{L - 2} + ... + ci \* a^{L - 1 - i} + ... + c{L - 1}\* a^1 + cL\* a^0

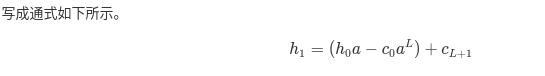


下面来考虑窗口从 abcd 滑动到 bcde 的情况。这时候整数形式数组从 [0, 1, 2, 3] 变成了 [1, 2, 3, 4]，数组最左边的 0 被移除，同时最右边新添了 4。滑动后数组的哈希值可以根据滑动前数组的哈希值来计算，计算公式如下所示。

h1 = (h0 - 0\* 26^3) \* 26 + 4 \* 26^0

写成通式如下所示。

h1 = (h0 \* a - c0\* a^L) + c{L + 1}



Hi = (h{i - 1} \* a - c{i - 1} \* a^L) + c{L + i - 1}

如何避免溢出

a^L 可能是一个很大的数字，因此需要设置数值上限来避免溢出。设置数值上限可以用取模的方式，即用 h % modulus 来代替原本的哈希值。

理论上，modules 应该取一个很大数，但具体应该取多大的数呢? 详见这篇文章，对于这个问题来说 2^{31}2就足够了。

算法

计算子字符串 haystack.substring(0, L) 和 needle.substring(0, L) 的哈希值。

从起始位置开始遍历：从第一个字符遍历到第 N - L 个字符。

根据前一个哈希值计算滚动哈希。

如果子字符串哈希值与 needle 字符串哈希值相等，返回滑动窗口起始位置。

返回 -1，这时候 haystack 字符串中不存在 needle 字符串。