Substring with Concatenation of All Words

Difficulty: Hard

这道题题目意思很好理解，暴力做也不难，但是复杂度太高。这里借助了Robin-Karp算法，达到了$O(n+mk)$的复杂度，这里n为串s的长度，m为words的数量，k为每个单词的长度。

Robin-Karp

首先简单的介绍一下Robin-Karp算法。这是一种字符串匹配算法，尤其适用于多个模式串的情景。算法的核心是计算每一个模式串的哈希值，构造哈希表。然后计算每一段文本串的哈希值，依次在哈希表中查找这个值，如果成功查找，那么再比较这一段文本与对应模式串，判断是否匹配。

本题中所有words长度相同的条件，对这种算法就非常友好了，省去了很多额外的处理。

对于长度为k的模式串，将其作为一个d进制的数字，d可以取为字母表的大小。例如如果字母表只含有数字，那么d可以取10，或者字母表用到了所有的ASCII字符，那么d可以取128，等等。

考虑到当k很大的时候，可能会造成哈希值的溢出，因此可以在计算的过程中对哈希值模q，q可以取为任意一个较大的质数。

用$O(mk)$的时间即可计算完所有的模式串的哈希值，并构造哈希表。接下来是对文本串s的处理。依次计算$s[0,\cdots,k-1], s[1,\cdots,k],\cdots$的哈希值$t[0], t[1],\cdots$，并在哈希表中查找，即可完成字符串匹配的过程。通常来讲，计算这$n-m+1$个哈希值，需要$O(k(n-m))$的时间，但是利用下面的递推式，可以将时间缩小到$O(n)$.

$$t[i+1] = (d(t[i] - s[i]h) + s[i+m])\%q$$
$$h = d^{m-1}\%q$$

回到本题

那么对本题来讲，可以用一个数组l记录匹配情况，$l[i]$表示$s[i, i+1, \cdots, i+k-1]$与第$l[i]$个单词匹配。

依次对以下序列进行类似于滑动窗口的操作：

$$
\begin{gathered}
l[0], l[k], l[2k], \cdots \\
l[1], l[k+1], l[2k+1], \cdots \\
\vdots \\
l[k-1], l[2k-1], l[3k-1], \cdots
\end{gathered}
$$

窗口大小为m，不断判断窗口中的各个单词匹配的次数是否与要求次数相同即可。这部份的复杂度为$O(n-m)$.