

一开始看了很多篇文章，感觉都非常复杂，没有搞懂，理解起来有点困难，不过花一周，差不多理解了。

感觉阮一峰老师的这一篇是最容易懂的。

http://www.ruanyifeng.com/blog/2013/05/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm.html

以一个算法小白来说，要理解这个分这几个步骤。

1. 朴素算法 和 KMP 的对比
2. KMP 算法为什么可以回溯指针
3. PMT
4. 怎么在程序上实践

1. 朴素算法

朴素算法，就是穷举，这个很容易理解，还是以阮一峰 的文章举例

上面全句的循环设为：N

下面要搜索的词循环设为：M

先循环 N，直到找到 A。

BBC ABCDAB ABCDABCDABDE
ABCDABD

找到 A，M 开始进入循环，一一对比，

BBC ABCDAB ABCDABCDABDE
ABCDABD

匹配到 空格 和 D，不一致，M 退出当前循环

BBC ABCDAB ABCDABCDABDE
ABCDABD

N continue，继续下一个，直到全部找到为止。

BBC ABCDAB ABCDABCDABDE
ABCDABD

所以，最坏的情况是：O(N * M)

2. KMP 概念理解

上面的朴素算法，假设是一个记忆力非常差的人，一旦发现不对，只能从头开始。

但是现在假设有一个记忆力非常好的人，只要匹配过一次，他就可以跳过之前相同的

我们现在来看要搜索的这个关键字

ABCDABD

细心的你会发现，高亮的部分是一样的。假设这个人记忆力很好，他可以跳过相同部分，直接从后面不一样的开始。

如果用回刚刚的循环，如果匹配到 D 不一样，这个时候不会再退回 从头开始了。

BBC ABCDAB ABCDABCDABDE
ABCDABD

直接从 C 开始匹配，因为 AB 已经匹配过了。

BBC ABCDAB ABCDABCDABDE
ABCDABD

因为 C 和 空格，也不匹配，所以 M 继续回退到 A，为什么要回退到 A，因为『AB』（里面有个空格），AB 已经匹配过，所以，退回到 M 开头。

BBC ABCDAB ABCDABCDABDE
ABCDABD

这个时候 A 和空格 还是不匹配，N 推进，A 相等，继续推进，直到，又是 D 不匹配

BBC ABCDAB ABCDABCDABDE
ABCDABD

跟前面一样，直接跳过已经匹配过的 AB，后移 4 位

BBC ABCDAB ABCDABCDABDE
ABCDABD

这一次终于找到 D 了，如果只搜索一次的话，匹配就结束了

3. PMT

KMP 的核心在于，PMT。但是找了很多文章，其实没有说明，PMT 到底为什么要设计成这样，不过核心思想就是，获取 pattern 每个字符匹配的最大长度。

阮一峰老师的文章只简单说了原理，怎么生成 PMT 没说，下面的图是我自己画的

迭代 Pattern，从 第二字符开始，进行匹配，如果不想等则为0，相等则为1，连续相等自增1.

首先 PMT 第一个字符 恒定为0，从第二个开始，不想等，则为0. 因此 PMT [0,0]

ABCDABD
ABCDABD

C 和 A 还是不等 PMT [0,0,0]

ABCDABD
ABCDABD

D 和 A 不想等 PMT[0,0,0,0]

ABCDABD
ABCDABD

这一次 AB 同时相等，因此 PMT: [0,0,0,0,1,2]

ABCDABD
ABCDABD

最后 D 不想等，则 PMT 最终生成 [0,0,0,0,1,2,0]

ABCDABD
ABCDABD

生成 PMT 的函数(javascript):

```
function getPMT(str) {  
  const pmt = [];  
  pmt[0] = 0;  
  let i = 0;  
  let j = 1;  
  while (j < str.length) {  
    if (str[i] === str[j]) {  
      i++;  
      pmt[j] = i;  
    } else {  
      pmt[j] = 0;  
      i = 0;  
    }  
    j++;  
  }  
  return pmt;  
}
```