推导&构建 KMP 算法 By Hug

（转载时，请注明出处：By Hug）

曾经自己理解了 KMP 算法以后，跟小伙伴吹牛逼：你信不信，我五分钟给你讲懂 KMP。事实证明，我失败了。KMP 就是这样的一种存在，理解了思想非常简单，可是真想讲出来，还给对方讲懂，让对方忘不了，真的是一件困难的事情。

静下心来，反复思索，形成如下推导过程，站在这个点上，我可以跟曾经的那个小伙伴说一句：我曾经吹的牛逼，自己圆回来，也谢谢你陪我在百度大厦同住了三个月。

经过如下推导过程，没有我曾经说的 5 分钟那么简单，但也绝对没有当初表述的那么复杂。来吧，我们开始吧。

阅读此文之前，请自行阅读如下资料：

http://blog.csdn.net/yutianzuijin/article/details/11954939

有了以上资料的了解后，让我们一起来开始推导&构建 KMP 算法

定义：A 是问题中的模式串（短串），长度为 n
定义：B 是问题中的文本串（长串），长度为 m

A[i] 代表 A 字符串的第 i 位
B[j] 代表 B 字符串的第 j 位
A[i, j] 代表 A 字符串第 i 位到第 j 位，(**注：**下标从 1 开始)
B[i, j] 代表 B 字符串第 i 位到第 j 位，(**注：**下标从 1 开始)

Sign(i, j) 为分段函数，取值{0, 1}，
当 A[i] = B[j] 的时候，sign(i, j) = 1
当 A[i] != B[j] 的时候，sign(i, j) = 0

重点一：原始问题的等价表示

$$\max(\sum_{i=1}^{n} sign(i, j+i-1) \mid j \in [1, m]) ? = n \quad \text{公式①}$$

原始问题转化为如上公式的判定问题，判定最大值公式是否等于 n，公式①与原问题等价

对于公式①做优化，其中 sign 的加和部分没有必要每次都运算 n 次，其实

第一次碰到 sign(i, j + i - 1) =0 的时候，就应该停止了。所以，假设，第一次碰到 sign(i, j + i - 1) = 0 的位置时，i 的值是 k + 1，则公式①，简化为如下公式：

$$\max\left(\sum_{i=1}^{k} sign(i, j+i-1) \;\middle|\; j \in [1, m], k+1 \text{ 位置失配}\right) ?= n \quad \text{公式②}$$

公式①和公式②所求问题等价。

重点二：推导公式②等价问题的性质

公式②中有两个不定量，j 和 k，j 与文本串有关，k 与答案有关，所以设置函数 f(j) = k，完成从文本串到答案的映射。
原问题变成 max(f(j)) ?= n，与寻找 f(j) 函数的最大值等价
设:

f(j) = k　　条件①
f(j + e) = l　　条件②

其中，l > k，e 为满足条件的最小正整数，具体含义是，未来能找到一个后面的位置，其匹配成功的长度比之前匹配成功的长度 k 更大。

条件①等价于：A[1,k] = B[j, j + k - 1]
条件②等价于：A[1, l] = B[j + e, j + e + l - 1]

将条件②与条件①对齐，看看我们能推出什么样的性质（这个性质，与原问题不等价，但是不满足这个性质，原问题肯定不成立）

另：j + e + l – 1 = j + k – 1 得 l = k – e，则有

A[1, k - e] = B[j + e, j + k - 1] = A[e + 1, k]

由此，我们推导得出 f(j) < f(j + e) 的一个重要性质，是通过 A 串来进行表达的，**这个性质就是：A[1, k - e] = A[e + 1, k]** （也就是通常所说的，前半段等于后半段）

由于 e 是满足条件的最小正整数，所以 **A[1, k - e] = A[e + 1, k]** 的含义如下：

1、 描述的物理含义是，前后最长相等的片段。
2、 当匹配 k+1 失败的时候，如果此时 A 串前 k 位匹配成功了，说明从 B 串的 j + e 位开始匹配，最起码能够匹配成功 k – e 位，那么下一次判断，应该是用 A[k-e + 1]位 与 B[j + k] 位进行比较。（对于这个性质的理解很重要，在重点三中会用到）

重点三：推导 k 和 e 的映射关系

设置函数 g(k) = k – e 等价于 A[1, k - e] = A[e + 1, k]，现在讨论 g(k + 1)的值：
操作一： 当 A[k – e + 1] = A[k + 1]时，g(k + 1) = g(k) + 1
操作二： 当 A[k – e + 1] != A[k + 1]时，说明:

A[1, k – e]A[k – e + 1] 与串 A[e + 1, k]A[k + 1] 在最后一位上失配了

将 A[e + 1, k]A[k + 1]看做是文本串（与 B 串的性质类似），A[1, k - e]A[k – e + 1]看做模式串，则根据以上推导，在 k – e 位匹配成功，下一位失配的情况下，我们应该用 A[g(k - e) + 1] (等价与 A[g(g(k)) + 1]) 位与 A[k + 1]进行比较，若相等，则进行操作一的类似操作，否则继续进行操作二的操作。

由于我们的字符串下表是从 1 开始的，所以定义边界条件 g(1) = 0，下标如果是从 0 开始，则 g(1) = -1

对于 g 函数的理解很重要，对于不同的 KMP 算法的实现，其实就是在维护 g 函数中不同的变量值，由于变量值涉及到 k 和 e 两个值，所以相应的，我们可以定义如下三种不同的 g 函数，对于每一种 g 函数的定义，就对应了不同的 KMP 算法的具体实现：

g(k) = k - e

g(k) = -e

g(k) = e

文章写到这里，我相信你已经具备了自己应用的能力了，得到了 g 函数（所谓的 next 数组）所有的对应关系以后，怎样应用，请回顾**重点二**中的内容。