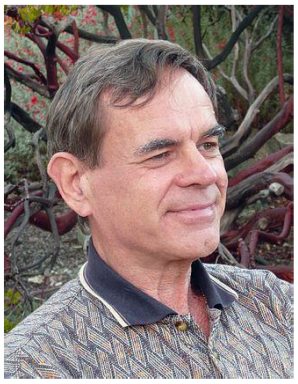
KMP算法

KMP算法

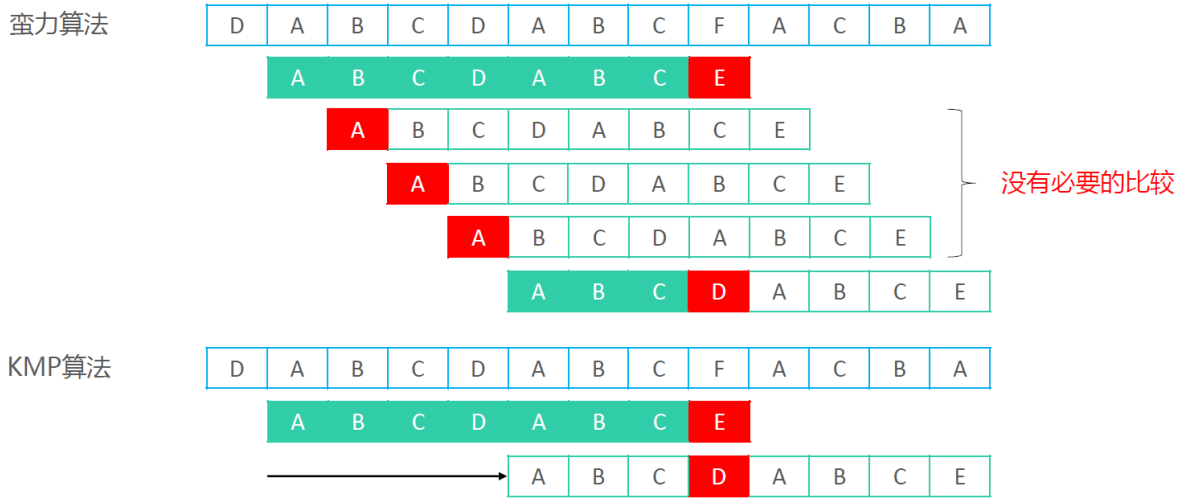
KMP算法，又称模式匹配算法，能够在线性时间内判定字符串p[1~N]是否为字符串s[1~M]的子串，并求出p在s中各次出现的位置。

Donald Knuth

James Hiram Morris



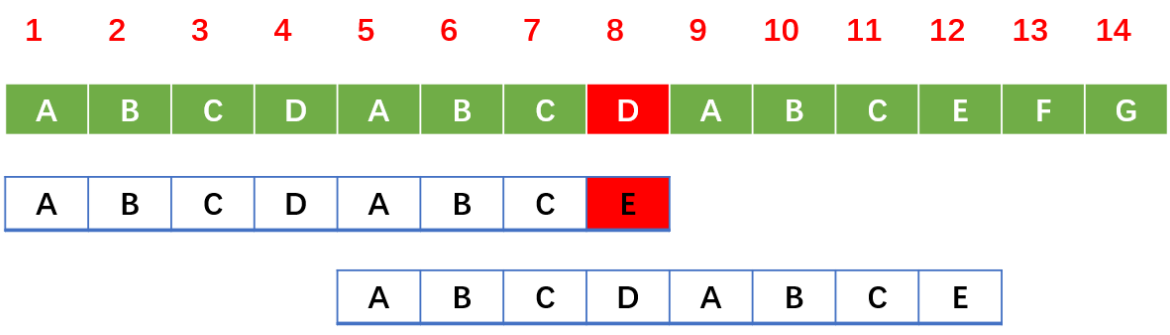
Vaughan Pratt



算法实现

求next数组：以模式串p为基础，以p[i]结尾最长真公共前后缀子串长度。

模式串P	A	B	C	D	A	B	C	E
下标	1	2	3	4	5	6	7	8
next	0	0	0	0	1	2	3	0



1. 初始化：next[1] = 0， j = 0，假定next[1~i-1]已求出，下面求解next[i]；
2. 不断尝试扩展匹配长度j，如果扩展失败，即下一个字符不相等，令 j = next[j]，直至j为0，重新从头匹配；
3. 如果能够扩展成功，匹配长度增加1，next[i]的值就是j。

```
for (int i = 2, j = 0; i <= plen; i++) {
    while (j && p[i] != p[j + 1]) j = ne[j];
    if (p[i] == p[j + 1]) j++;
    ne[i] = j;
}


for (int i = 1, j = 0; i <= slen; i++) {
    while (j && s[i] != p[j + 1]) j = ne[j];
    if (s[i] == p[j + 1]) j++;
    if (j == plen) {
        cout << i - plen + 1 << endl;
        j = ne[j];
    }
}
```

时间复杂度：while循环中，j的值不断减少，j = ne[j] 的次数不会超过每层for循环开始 时j的值与while循环结束时j的值之差，而j的值至多增加1，j的减小幅度不会超过 j增加幅度总和，时间复杂度O(N + M)。

快乐刷题

- P18 KMP模板
- P443 剪花布条
- P453 删减字符串