串

串的模式匹配算法

BF算法

时间复杂度

最好 $O(m+n)$,最坏 $O(mn)$

原理

暴力匹配，不同则后移一位

代码

```
int Index(SString S,SString T){
        int i=1,j=1;
        while(i<S.length&&j<T.length){
                if(S.ch[i]==T.ch[i])
                {
                        ++i,++j;
                }
                else{
                        i=i-j+2;j=1;
                }
                if(j>T.length) return i-T.length;
                else return 0;
        }
}
```

KMP算法

部分匹配值

前缀：除最后一个字符外，字符串所有的头部子串->q
后缀：除第一个字符外，字符串所有的尾部字串->l
length=最长相等前后缀长度
例子：'ababa'

- 'a': $\{\emptyset\} \cap \{\emptyset\} = \emptyset$
- 'ab': $\{a\} \cap \{b\} = \emptyset$

- 'aba': $\{a, ab\} \cap \{a, ba\} = \{a\}, length = 1$
- 'abab': $\{a, ab, aba\} \cap \{b, ab, bab\} = \{ab\}, length = 2$
- 'ababa': $\{a, ab, aba, abab\} \cap \{a, ba, aba, baba\} = \{a, aba\}, length = 3$

编号	1	2	3	4	5
S	a	b	a	b	a
PM	0	0	1	2	3

移动位数=已匹配字符数-对应的部分匹配值

时间复杂度

$O(m+n)$

原理

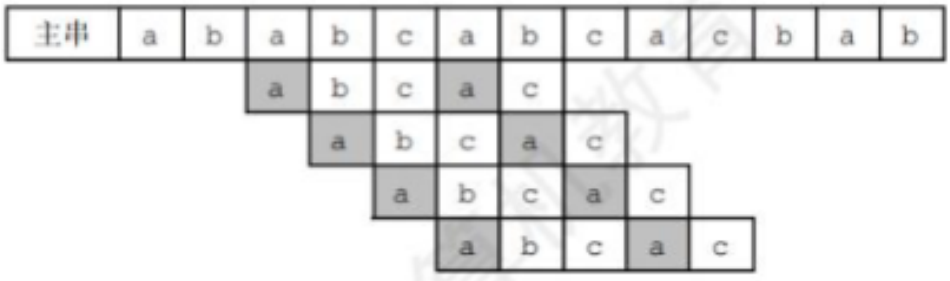

图 4.3　失配后移动情况

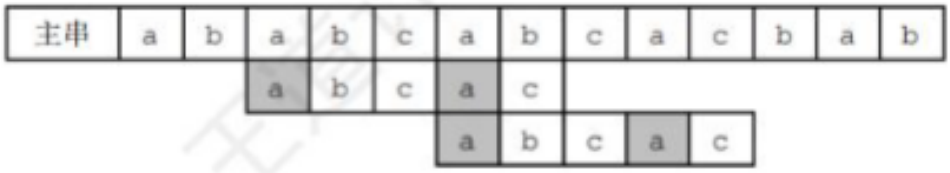

图 4.4　直接移动到合适位置

改进

移动位数=已匹配字符数-对应的部分匹配值
$MOVE = (j-1) - PM[j-1]$
使用部分匹配值时，每当匹配失败，就去找它前一个元素的部分匹配值，这样使用起来有些不方便，所以将PM表右移一位，这样哪个元素匹配失败，直接看它自己的部分匹配值即可。

编号	1	2	3	4	5
S	a	b	a	b	a
next	-1	0	0	1	2

变成了
$MOVE = (j-1) - next[j]$
字串比较指针回退到： $j = j - MOVE = next[j] + 1$
有时为了公式简洁将next整体加一

编号	1	2	3	4	5
S	a	b	a	b	a
next	0	1	1	2	3

进一步优化

主串	a	a	a	b	a	a	a	a	b
模式串	a	a	a	a	b				
j	1	2	3	4	5				
next[j]	0	1	2	3	4				
nextval[j]	0	0	0	0	4				

图 4.7　KMP 算法进一步优化示例

递归直至不再出现 $P_j = P_{next[j]}$

解题方法

1. 做出PM
2. 右移一位
3. 加一
4. 比较