

11. 串

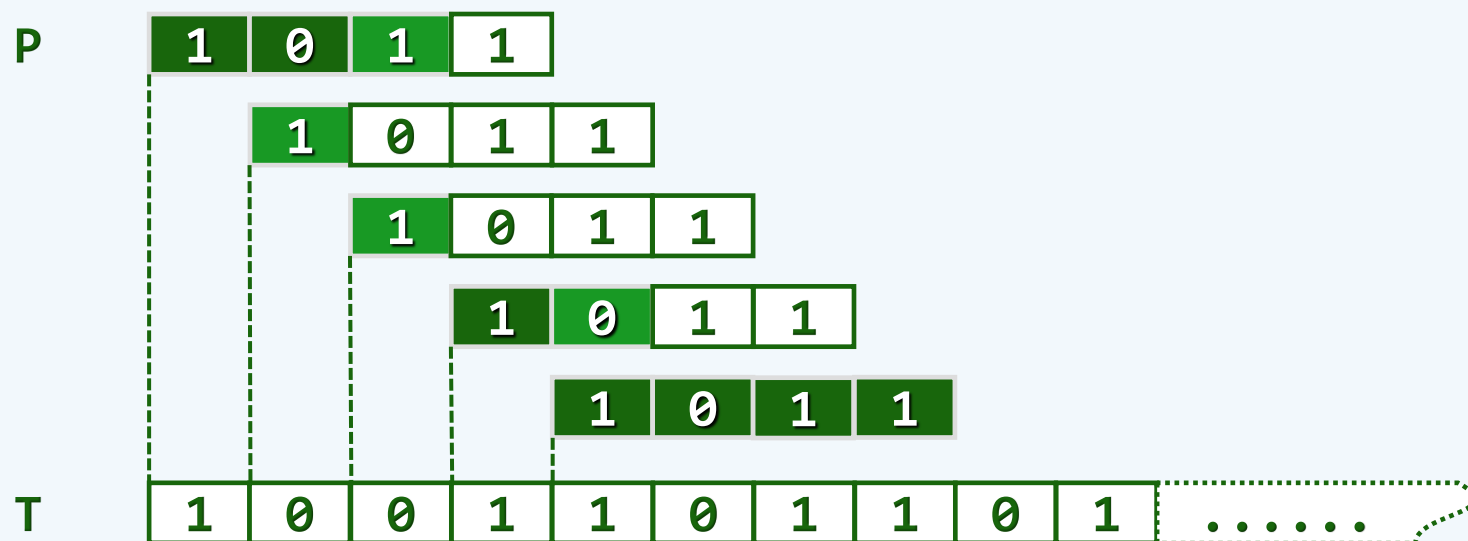
(b2) 蛮力匹配

邓俊辉

deng@tsinghua.edu.cn

构思

- ❖ 自左向右，以字符为单位，依次移动模式串直到在某个位置，发现匹配



- ❖ 如何：确定串T和P每次做比对的字符位置？并在发现某对字符失配后，调整位置以继续比对？

实现

❖ 实现1 : i 和 j 分别指向 $T[]$ 和 $P[]$ 中待比对的字符

```
if ( T[i] == P[j] ) { i++; j++; }
```

//若匹配，则 i 和 j 同时右移

```
else { i -= j - 1; j = 0; }
```

//若失配，则 T 回退、 P 复位

❖ 实现2 : $i + j$ 和 j 分别指向 $T[]$ 和 $P[]$ 中待比对的字符

```
if ( T[i + j] == P[j] ) { j++; }
```

//若匹配，则 $i + j$ 和 j 同时右移

```
else { i++; j = 0; }
```

//若失配，则 i 右移， j 回溯

❖ 这两种实现方法，各有什么优缺点？

版本1

```
❖ int match( char * P, char * T ) {
```

```
    size_t n = strlen(T), i = 0;
```

```
    size_t m = strlen(P), j = 0;
```

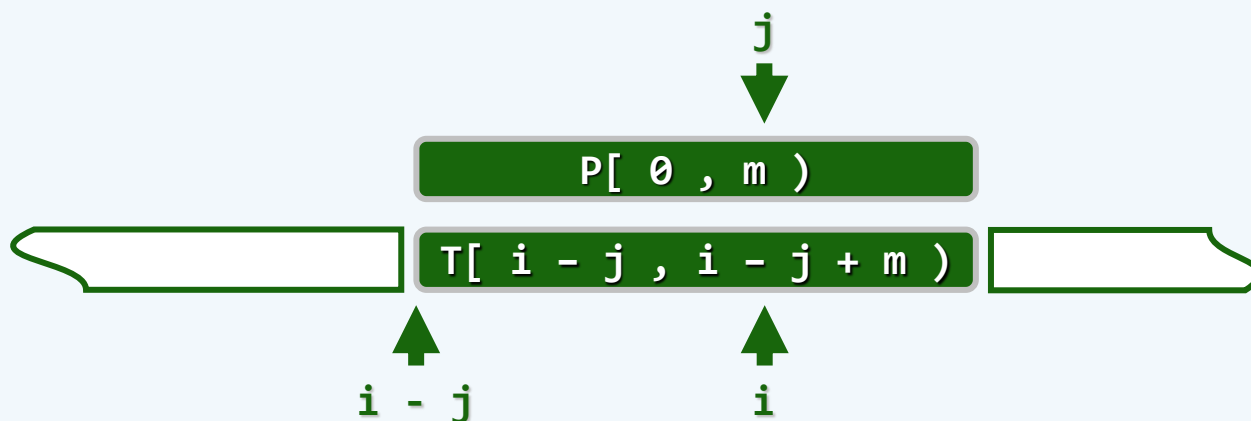
```
    while ( j < m && i < n ) //自左向右逐个比对字符
```

```
        if ( T[i] == P[j] ) { i++; j++; } //若匹配，则转到下一对字符
```

```
        else { i -= j - 1; j = 0; } //否则，T回退、P复位
```

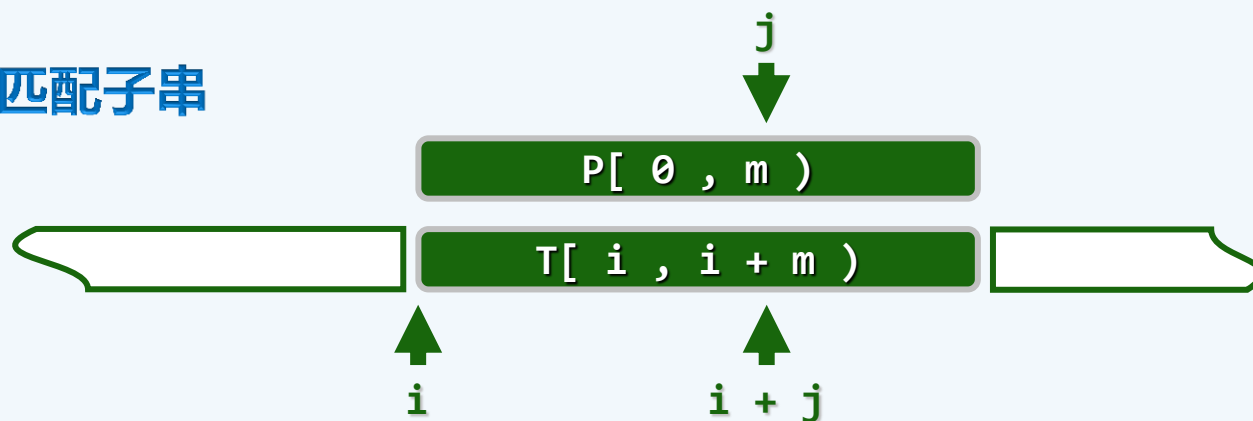
```
    return i - j;
```

```
} //如何通过返回值，判断匹配结果？
```



版本2

```
❖ int match( char * P, char * T ) {  
    size_t n = strlen(T), i = 0; //T[i]与P[0]对齐  
    size_t m = strlen(P), j; //T[i + j]与P[j]对齐  
    for ( i = 0; i < n - m + 1; i ++ ) { //T从第i个字符起，与  
        for ( j = 0; j < m; j ++ ) //P中对应的字符逐个比对  
            if ( T[i + j] != P[j] ) break; //若失配，P整体右移一个字符，重新比对  
        if ( m <= j ) break; //找到匹配子串  
    }  
    return i;  
} //如何通过返回值，判断匹配结果？
```



复杂度

❖ 最好情况（只经过一轮比对，即可确定匹配）： $\# \text{比对} = m = O(m)$

❖ 最坏情况（每轮都比对至P的末字符，且反复如此）

每轮循环： $\# \text{比对} = m - 1(\text{成功}) + 1(\text{失败}) = m$

循环次数 $= n - m + 1$

一般地有 $m \ll n$

故总体地， $\# \text{比对} = m \times (n - m + 1) = O(n \times m)$

❖ 最坏情况，真会出现？

0 0 0 0 0 0 0 0 0 0 0 ... 0 0 1

是的！

0 0 0 0 1

❖ $|\Sigma|$ 越小，最坏情况出现的概率越高； m 越大，最坏情况的后果更加严重