

串

模式匹配：问题 & 蛮力算法

13-B

一切事情，一旦失败，就显得很愚蠢

长的是磨难，短的是人生

邓俊辉

deng@tsinghua.edu.cn

## 循模式访问：问题特点 + 测试方法

*Text* : a man with money is no match against a man on a mission

*Pattern* : match

random  $T$  + random  $P$  ?

random  $T$  + substring  $P$  !

$$n = |T| \quad m = |P|$$

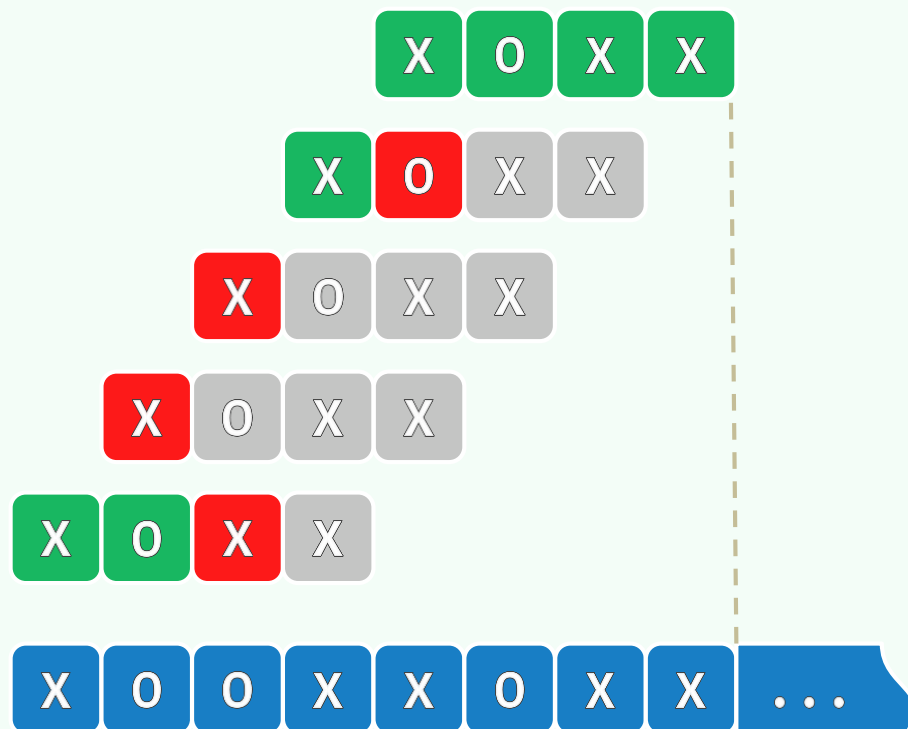
$$2 \ll m \ll n$$

$$s = |\Sigma|$$

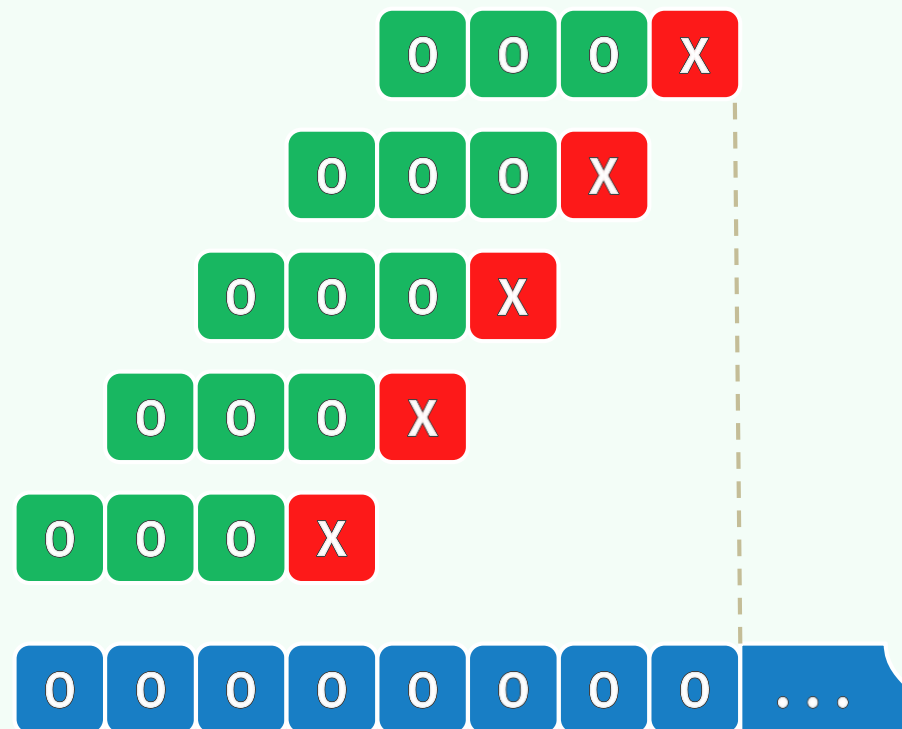
$$Pr_{match} \approx n/s^m \rightarrow 0$$

# 蛮力策略

*Best* :  $\Omega(n)$



*Worst* :  $\mathcal{O}(n \cdot m)$



# 版本1

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

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

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

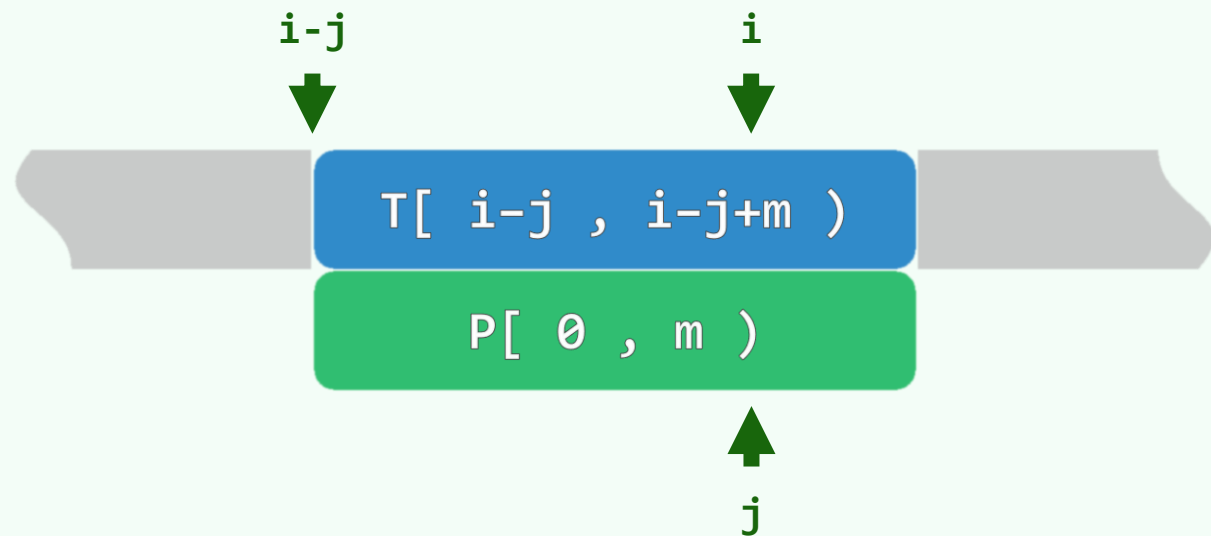
```
    while ( (j < m) && (i < n) ) //自左向右逐次比对 (可优化)
```

```
        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; //最终的对齐位置: 藉此足以判断匹配结果
```

```
}
```



## 版本2

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

```
    int n = strlen(T), i;
```

```
    int m = strlen(P), j;
```

```
    for ( i = 0; i+m <= n; i ++ ) { //T[i]与P[0]对齐后
```

```
        for ( j = 0; j < m; j ++ ) //逐次比对
```

```
            if ( T[i+j] != P[j] ) break; //失配, 转下一对齐位置
```

```
            if ( j >= m ) break; //完全匹配 (Python: 可以写得更简洁、优美)
```

```
    }
```

```
    return i; //最终的对齐位置: 藉此足以判断匹配结果
```

```
}
```

