

13-C2

串

KMP算法：查询表

好记性不如烂笔头

你能看见多远的过去，就能看见多远的未来。

邓俊辉

deng@tsinghua.edu.cn

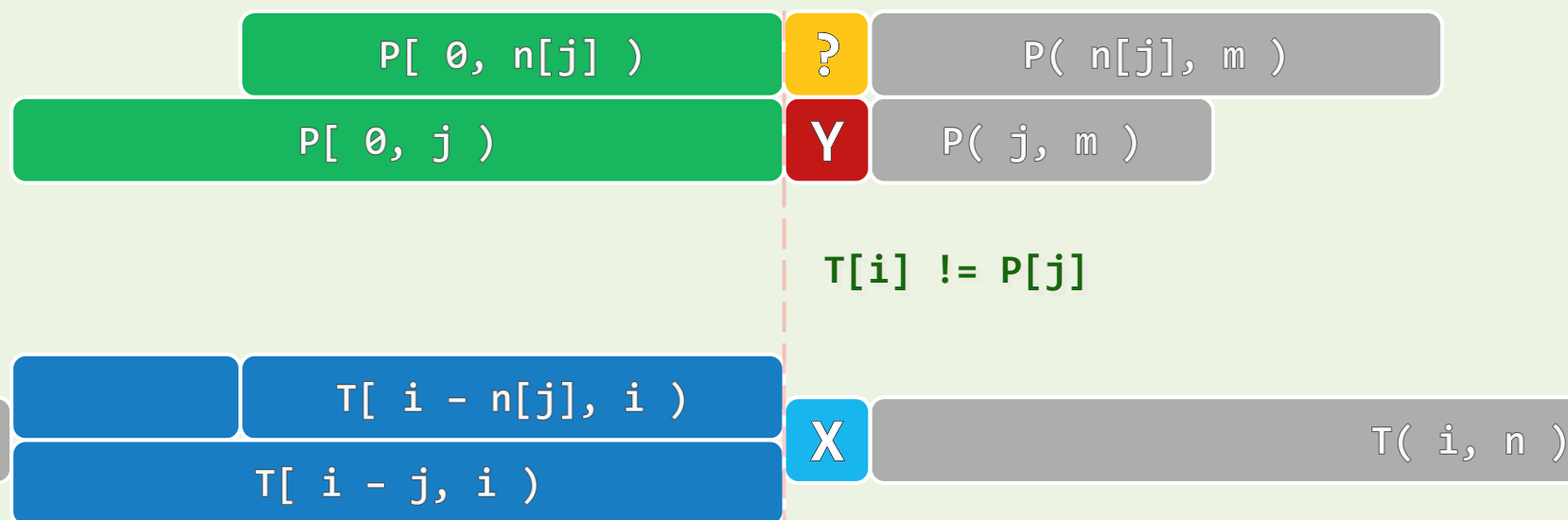
t: 不仅可以事先确定, 而且仅根据 $P[0, j) = T[i-j, i)$ 即可确定

❖ 视失败的位置j, 无非m种情况

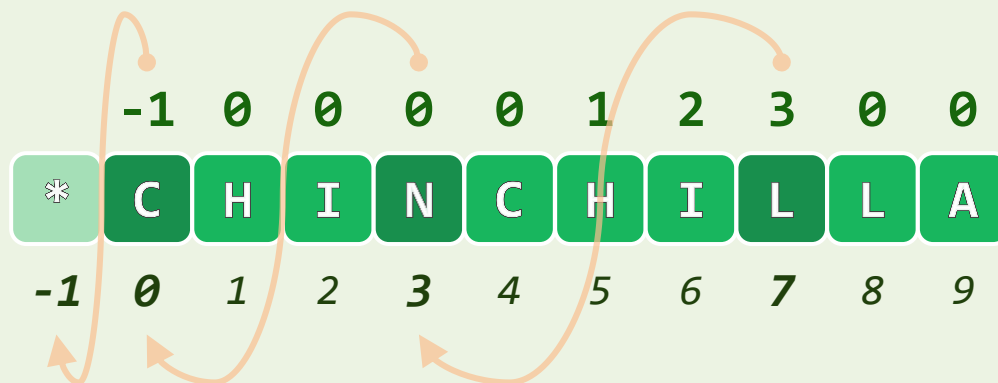
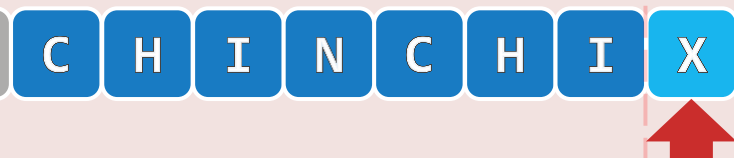
❖ 构造查询表 $next[0, m)$, 做好预案

❖ 一旦在 $P[j]$ 处失配, 只需

将j替换为 $next[j]$, 继续与 $T[i]$ 比对



实例



KMP算法

```
int match( char * P, char * T ) {  
    int * next = buildNext(P);  
    int n = (int) strlen(T), i = 0;  
    int m = (int) strlen(P), j = 0;  
    while ( j < m && i < n )  
        if ( 0 > j || T[i] == P[j] ) {  
            i ++; j ++;  
        } else  
            j = next[j];  
    delete [] next;  
    return i - j;  
}
```



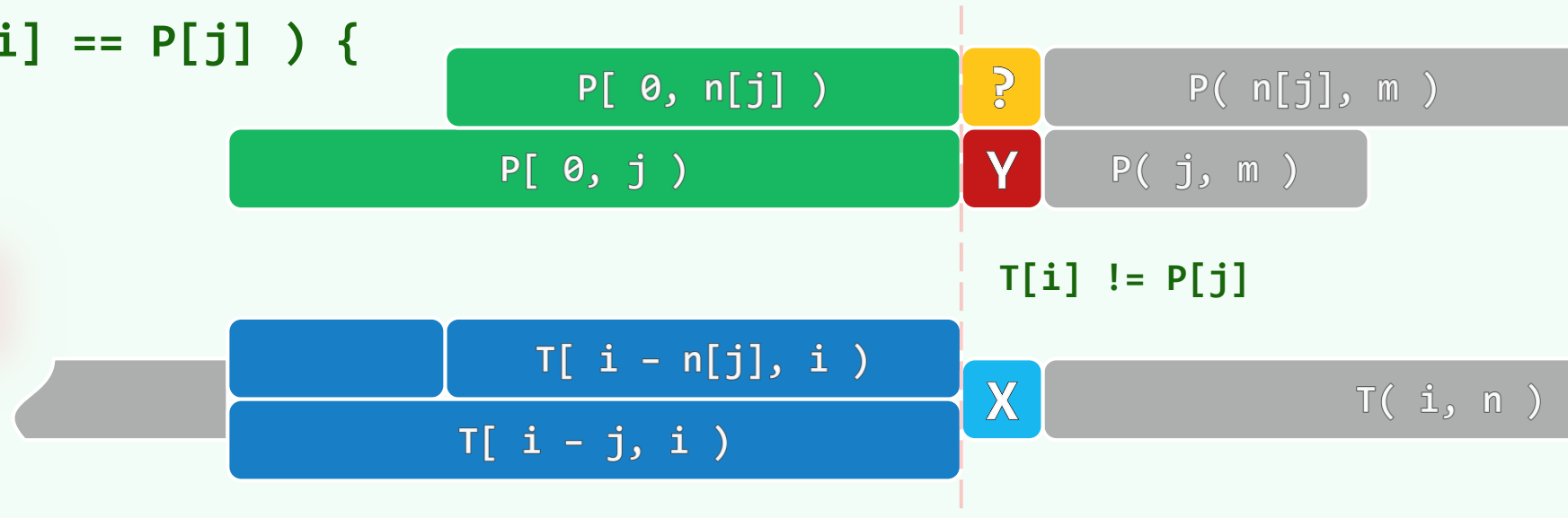
D. E. **K**nuth



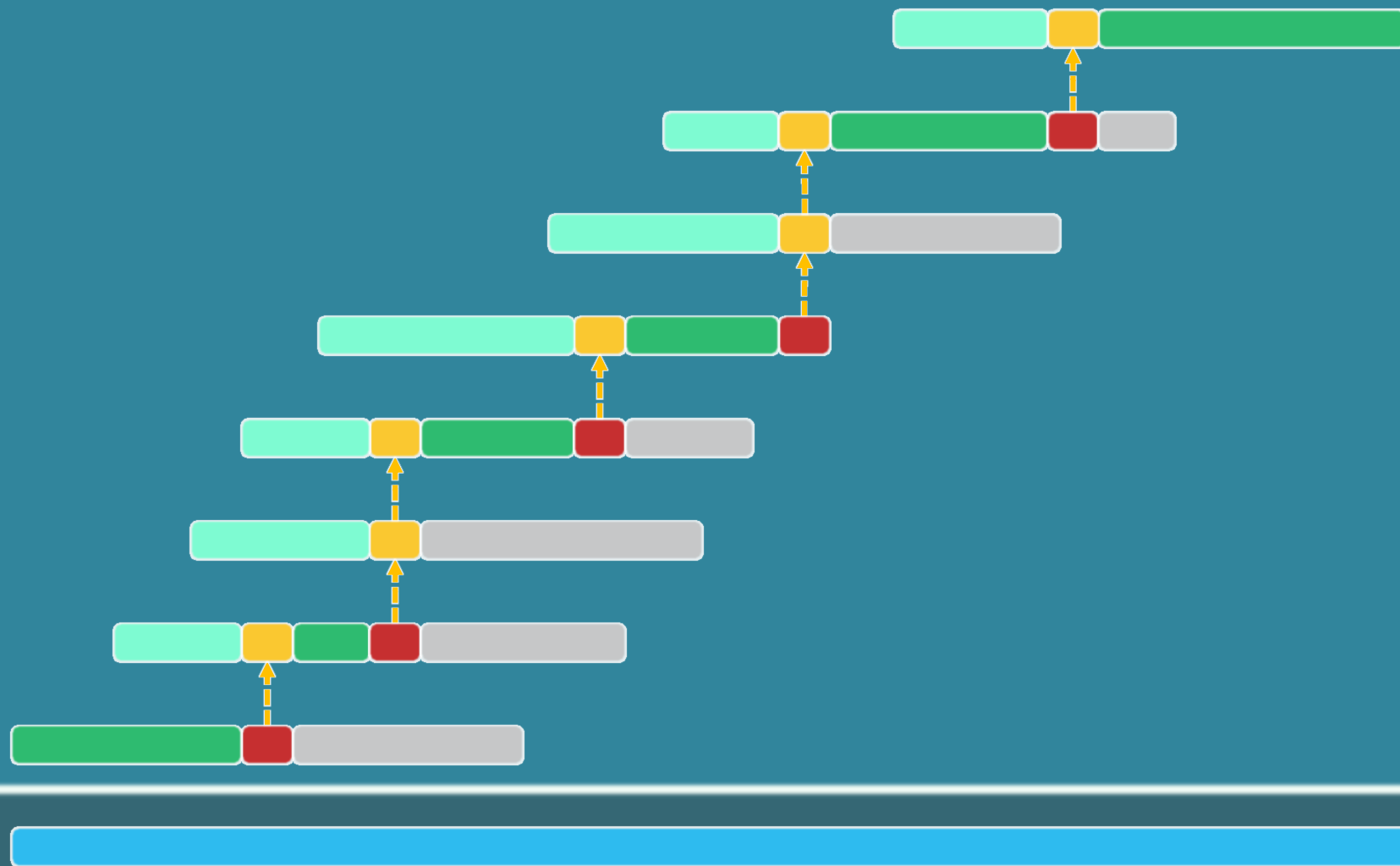
J. H. **M**orris



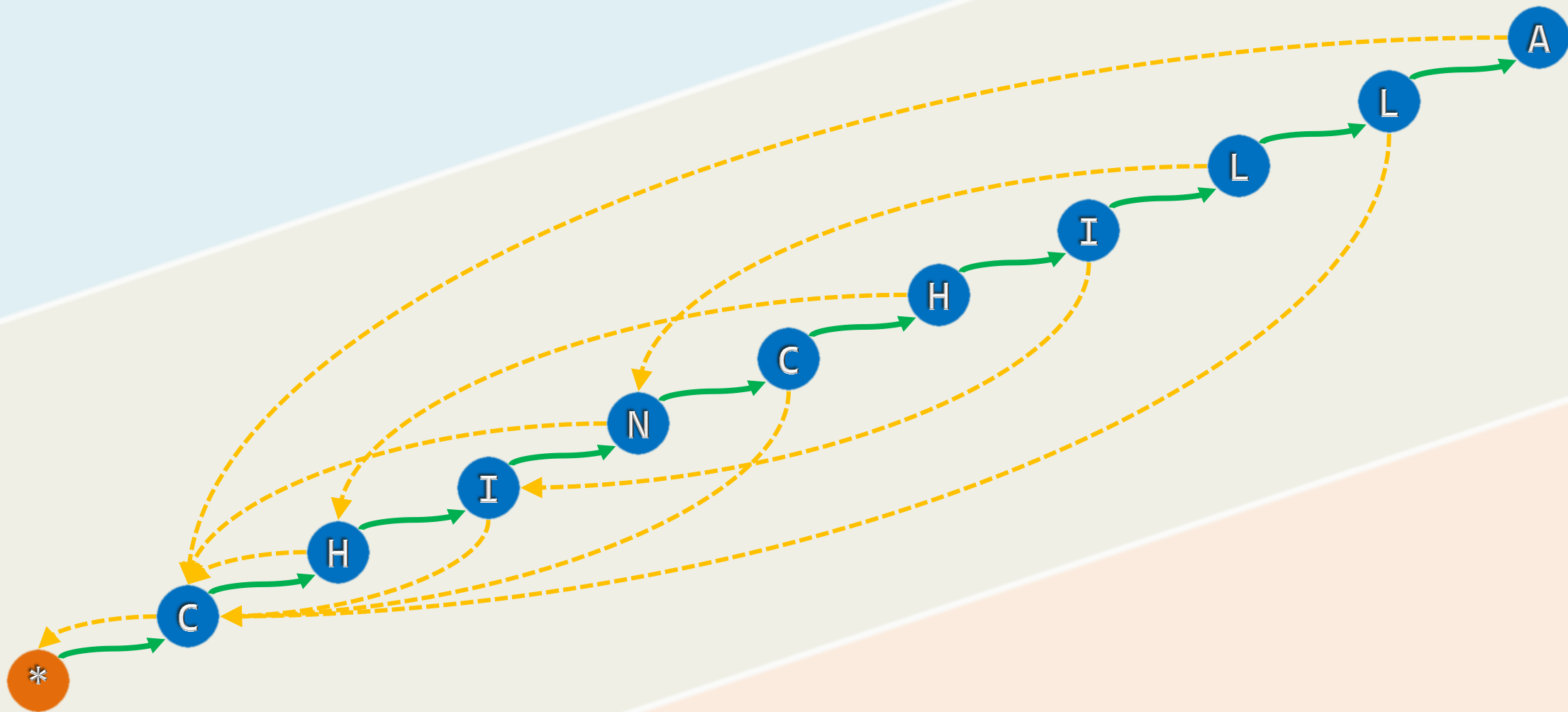
V. R. **P**ratt



快速右移 + 绝不回退



每一个P串，都是一台自动机



模式串 ~ 匹配算法

```
int match( char * T ) { //对任一模式串 (比如P = chinchilla) , 可自动生成如下代码
    int n = strlen(T); int i = -1; //文本串对齐位置
```

```
s_: ++i; // ↑
s0: (T[i] != 'C') ? goto s_ : if (n <= ++i) return -1; // [*] ~ ↑
s1: (T[i] != 'H') ? goto s0 : if (n <= ++i) return -1; // [*C] ~ [*]
s2: (T[i] != 'I') ? goto s0 : if (n <= ++i) return -1; // [*CH] ~ [*]
s3: (T[i] != 'N') ? goto s0 : if (n <= ++i) return -1; // [*CHI] ~ [*]
s4: (T[i] != 'C') ? goto s0 : if (n <= ++i) return -1; // [*CHIN] ~ [*]
s5: (T[i] != 'H') ? goto s1 : if (n <= ++i) return -1; // [*CHINC] ~ [*C]
s6: (T[i] != 'I') ? goto s2 : if (n <= ++i) return -1; // [*CHINCH] ~ [*CH]
s7: (T[i] != 'L') ? goto s3 : if (n <= ++i) return -1; // [*CHINCHI] ~ [*CHI]
s8: (T[i] != 'L') ? goto s0 : if (n <= ++i) return -1; // [*CHINCHIL] ~ [*]
s9: (T[i] != 'A') ? goto s0 : if (n <= ++i) return -1; // [*CHINCHILL] ~ [*]
    return i - 10; // [*CHINCHILLA]
}
```