11.串

（c2）KMP算法：查询表

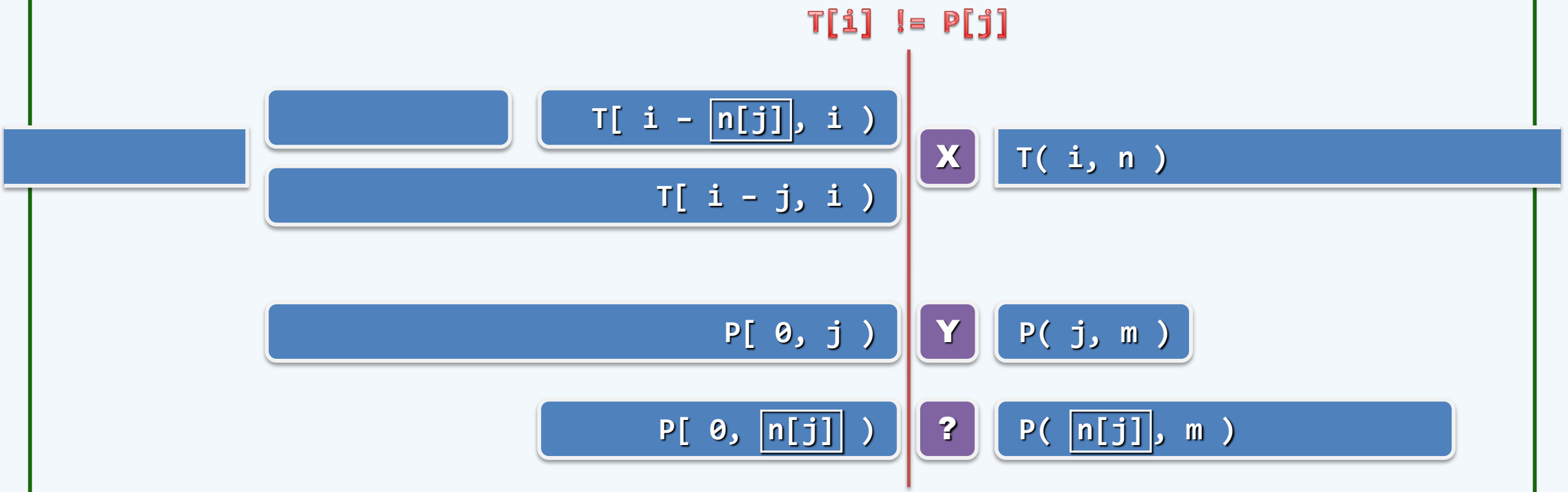
邓俊辉

deng@tsinghua.edu.cn

好记性不如烂笔头

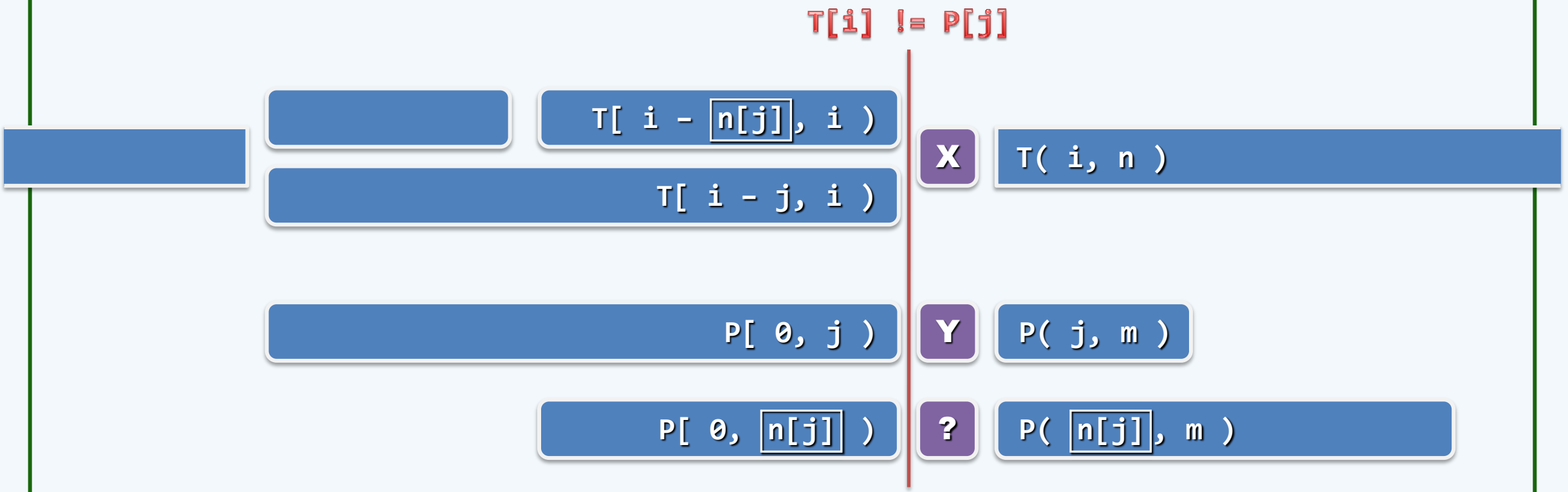
事先确定t

- 不仅可以事先确定，而且仅根据P即可确定（与T无关！）
- 根据失败位置P[j]，无非m种情况...



事先确定t

- 构造查询表next[0, m)：在任一位置P[j]处失败之后，将j替换为next[j]
- 与其说是借助强大的记忆，不如说是做好充分的预案



KMP算法

```
❖int match( char * P, char * T ) {
    int * next = buildNext(P); //构造next表
    int n = (int) strlen(T), i = 0; //文本串指针
    int m = (int) strlen(P), j = 0; //模式串指针
    while ( j < m && i < n ) //自左向右，逐个比对字符
        if ( 0 > j || T[i] == P[j] ) { //若匹配
            i ++;  j ++; //则携手共进
        } else //否则，P右移，T不回退
            j = next[j];
    delete [] next; //释放next表
    return i - j;
}
```

D. E. Knuth

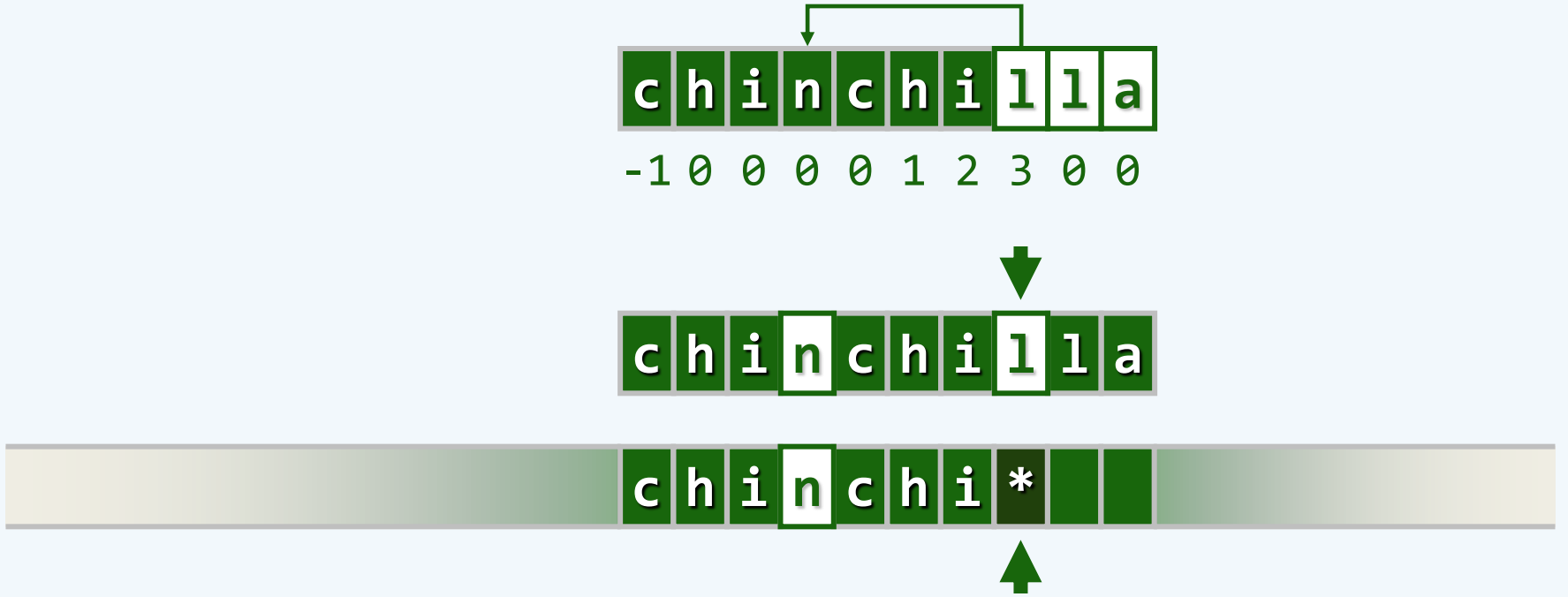
J. H. Morris

V. R. Pratt

T[i] != P[j]

P[0, n(j))	?	P(n(j), m)
P[0, j)	Y	P(j, m)
T[0, i)	X	T(i, n)

实例


c h i n c h i l l a
-1 0 0 0 0 1 2 3 0 0
c h i n c h i l l a
c h i n c h i *
c h i n c h i l l a
c h i n c h i l l a

自动机

❖ 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
}
```

自动机