

11. 串

(c4) KMP算法：构造next[]表

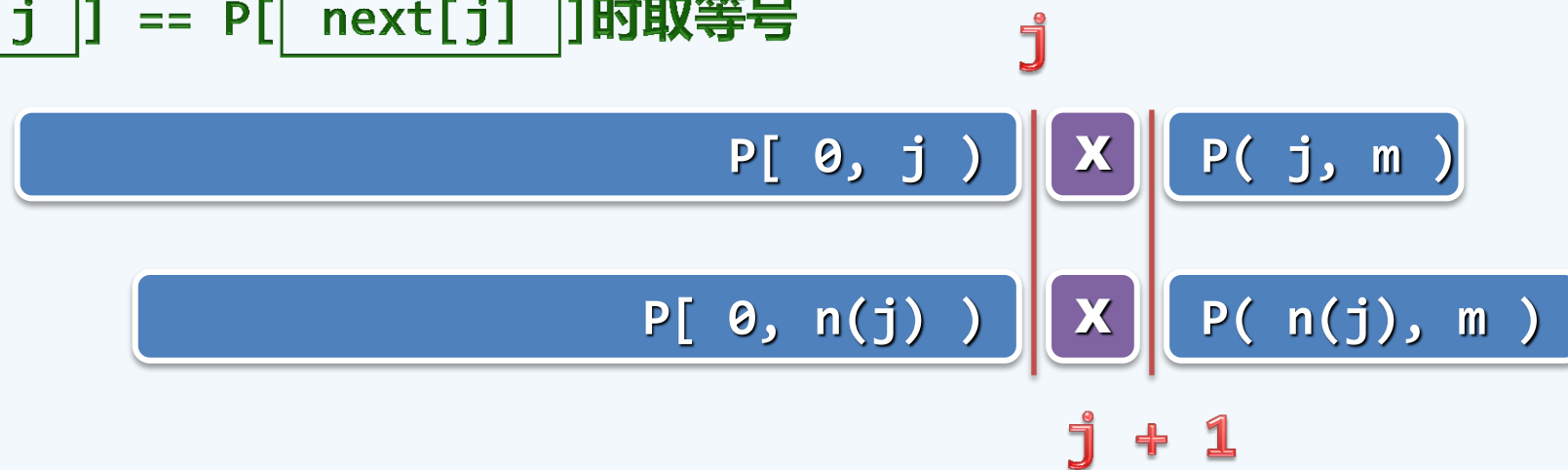
邓俊辉

deng@tsinghua.edu.cn

递推

- ❖ 根据已知的 $\text{next}[0, j]$, 如何高效地计算 $\text{next}[j + 1]$?
- ❖ 所谓 $\text{next}(j)$, 即是在 $P[0, j)$ 中, 最大自匹配的 **真前缀** 和 **真后缀** 的长度
- ❖ 故: $\text{next}[j + 1] \leq \text{next}[j] + 1$

特别地, 当且仅当 $P[j] == P[\text{next}[j]]$ 时取等号



- ❖ 一般地, $P[j] \neq P[\text{next}[j]]$ 时, 又该如何得到 $\text{next}[j + 1]$?

算法

❖ $\text{next}[j + 1]$ 的候选者
依次应该是：

$1 + \text{next}[j]$

$1 + \text{next}[\text{next}[j]]$

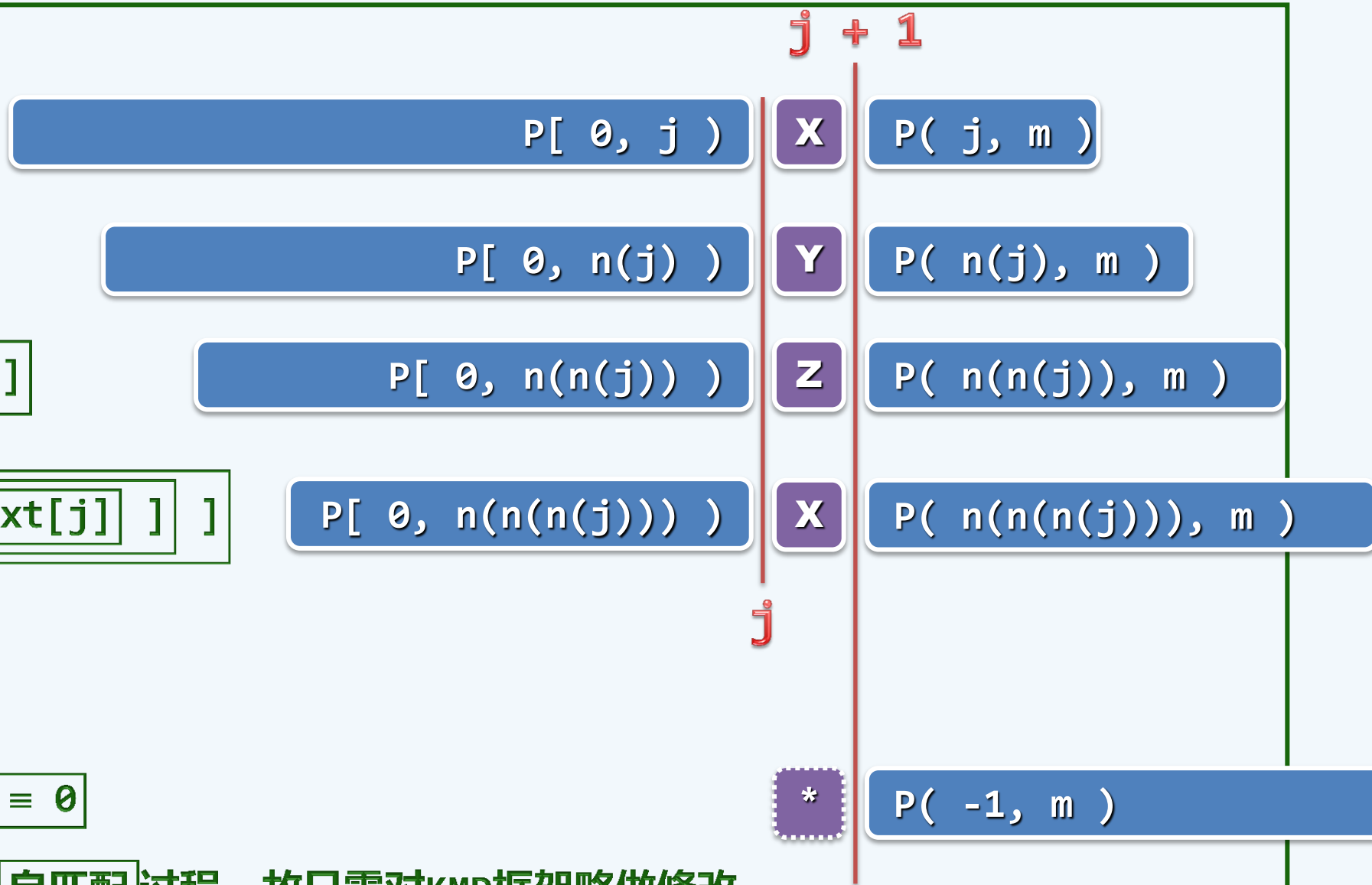
$1 + \text{next}[\text{next}[\text{next}[j]]]$

...

❖ 这个序列严格递减，且

必收敛于 $1 + \text{next}[0] \equiv 0$

❖ 以上递推过程，即是P的自匹配过程，故只需对KMP框架略做修改...



实现

❖ `int * buildNext(char * P) { //构造模式串P的next[]表`

`size_t m = strlen(P), j = 0; //“主” 串指针`

`int * N = new int[m]; //next[]表`

`int t = N[0] = -1; //模式串指针 (P[-1] 通配符)`

`while (j < m - 1)`

`if (0 > t || P[j] == P[t]) //匹配`

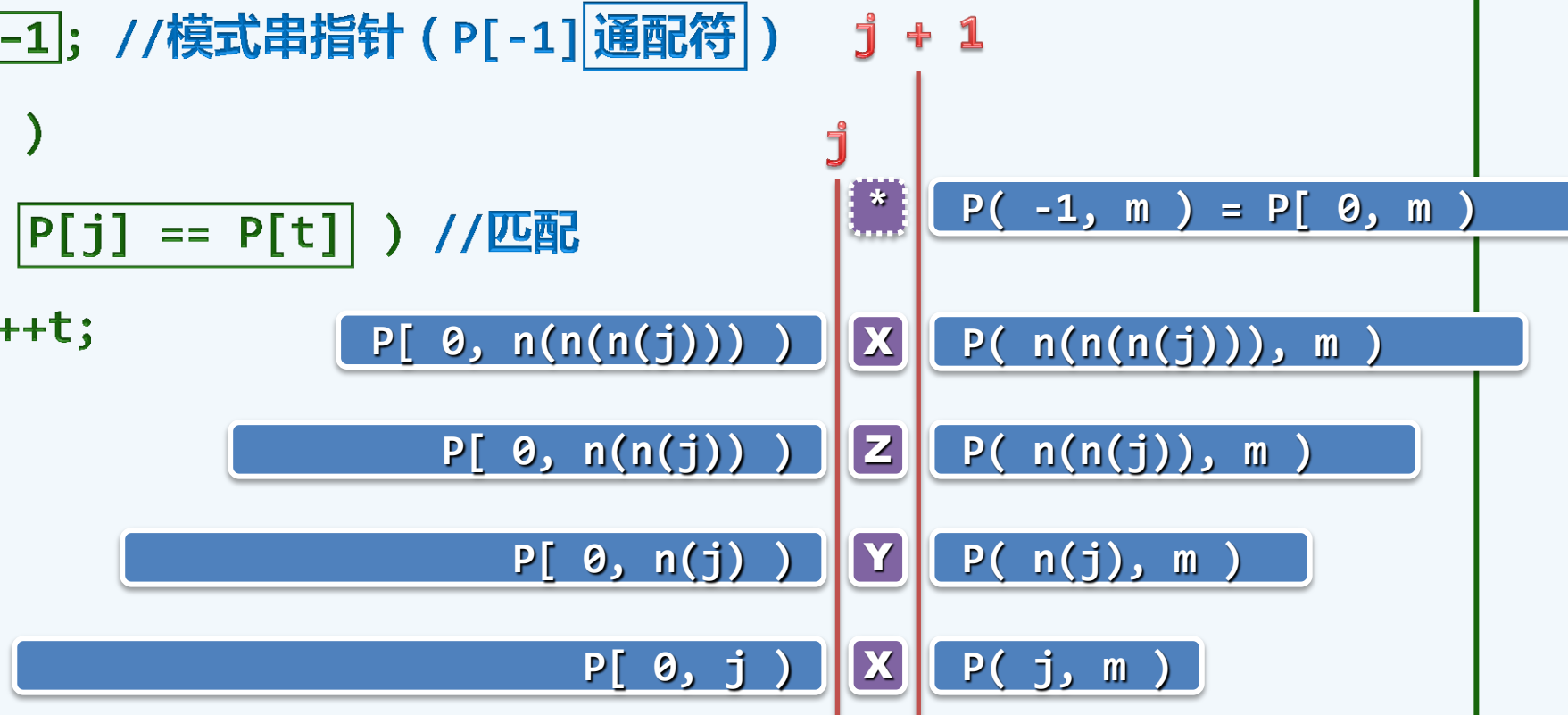
`N[++j] = ++t;`

`else //失配`

`t = N[t];`

`return N;`

`}`



实例

* c h i n e c h i n c h i l l a
|
c h i n e c h i n c h i l l a
-1 0

* c h i n e c h i n c h i l l a
|
c h i n e c h i n c h i l l a
-1 0

* c h i n e c h i n c h i l l a
|
c h i n e c h i n c h i l l a
-1 0 0

实例

