11.串

（c6）KMP算法：再改进

邓俊辉

deng@tsinghua.edu.cn

前车之覆，后车之鉴

反例

❖ T = 0 0 0 1 0 0 0 0 1

P = 0 0 0 0 1

❖ T[3]：

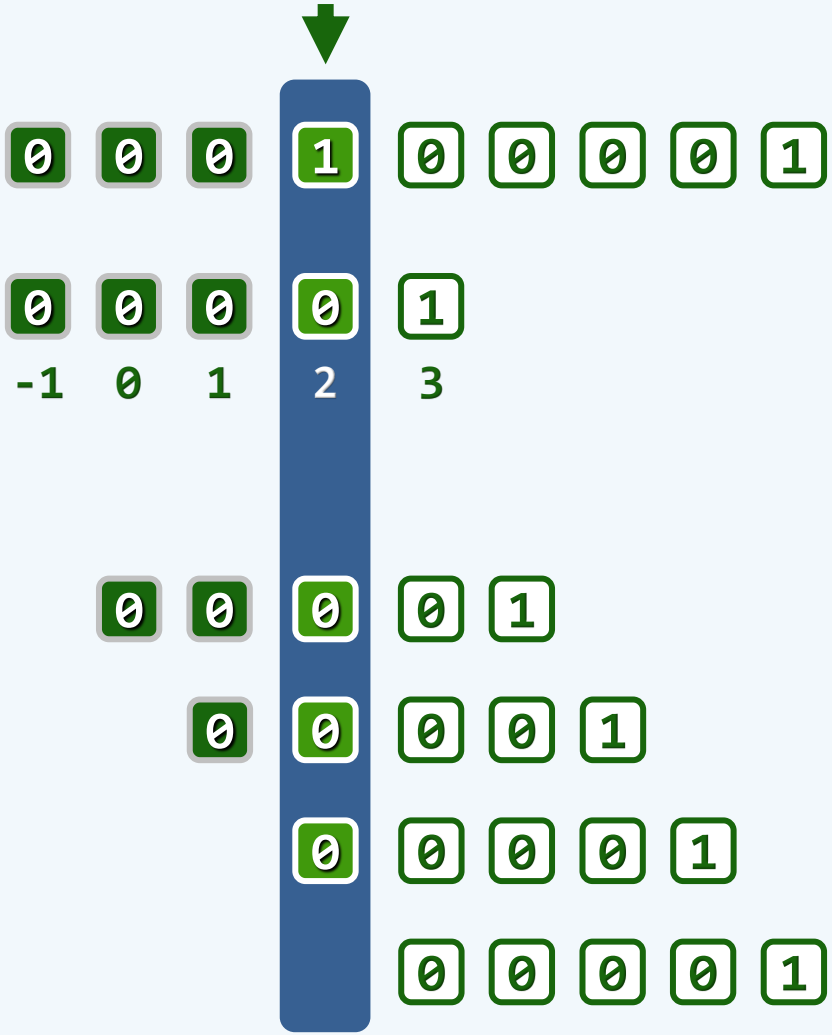
与 P[3] 比对，失败

与 P[2] = P[next[3]] 继续比对，失败

与 P[1] = P[next[2]] 继续比对，失败

与 P[0] = P[next[1]] 继续比对，失败

最终，才前进到T[4]



根源

❖ 无需T串，即可在事先确定：

P[3] =

P[2] =

P[1] =

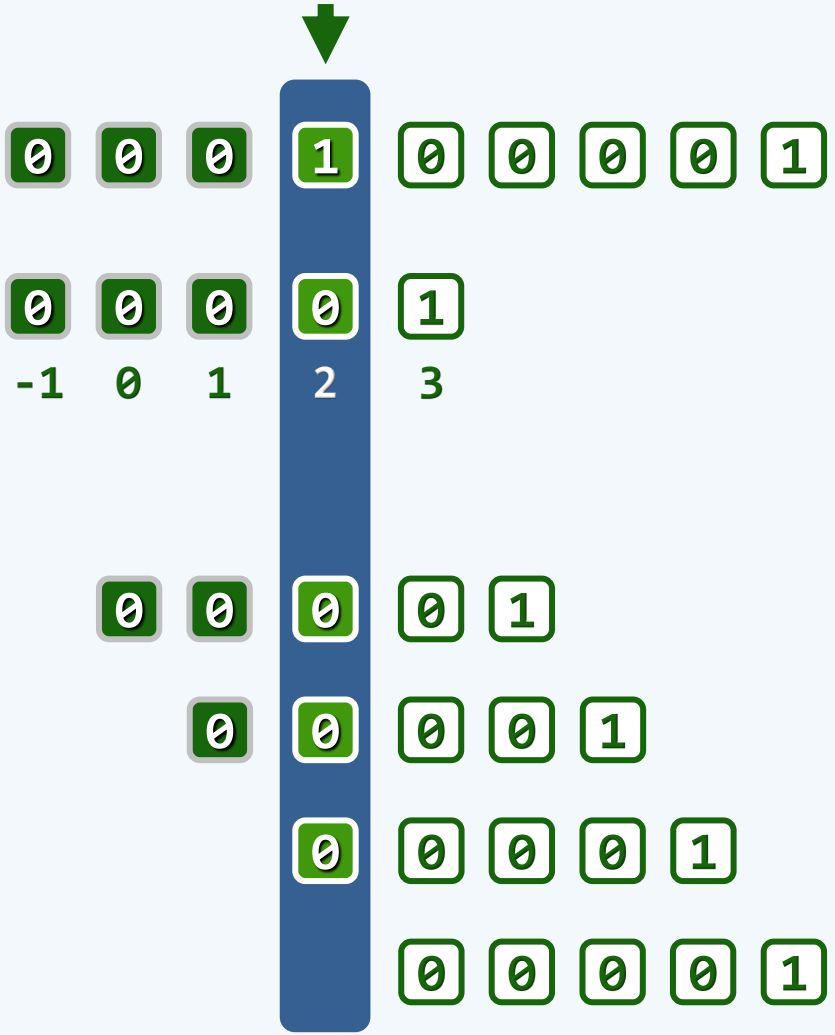
P[0] = 0

既然如此...

❖ 在发现 T[3] != P[3] 之后

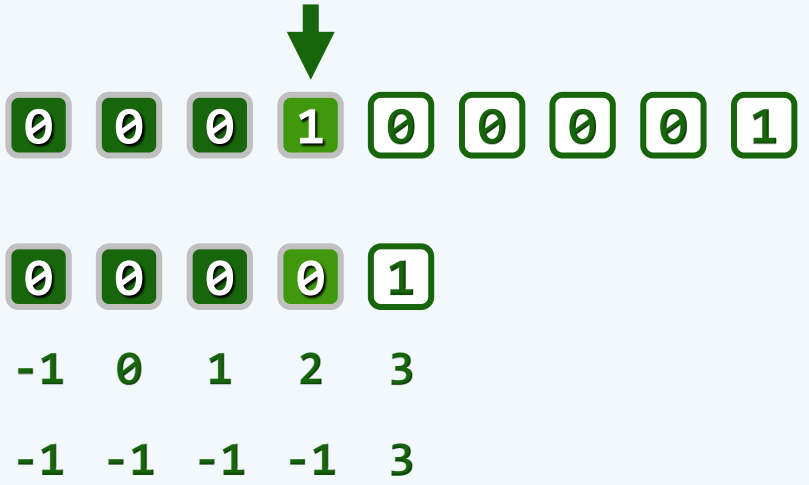
为何还要 一错再错 ？

❖ 事实上，后三次比对本来都是 可以避免 的！

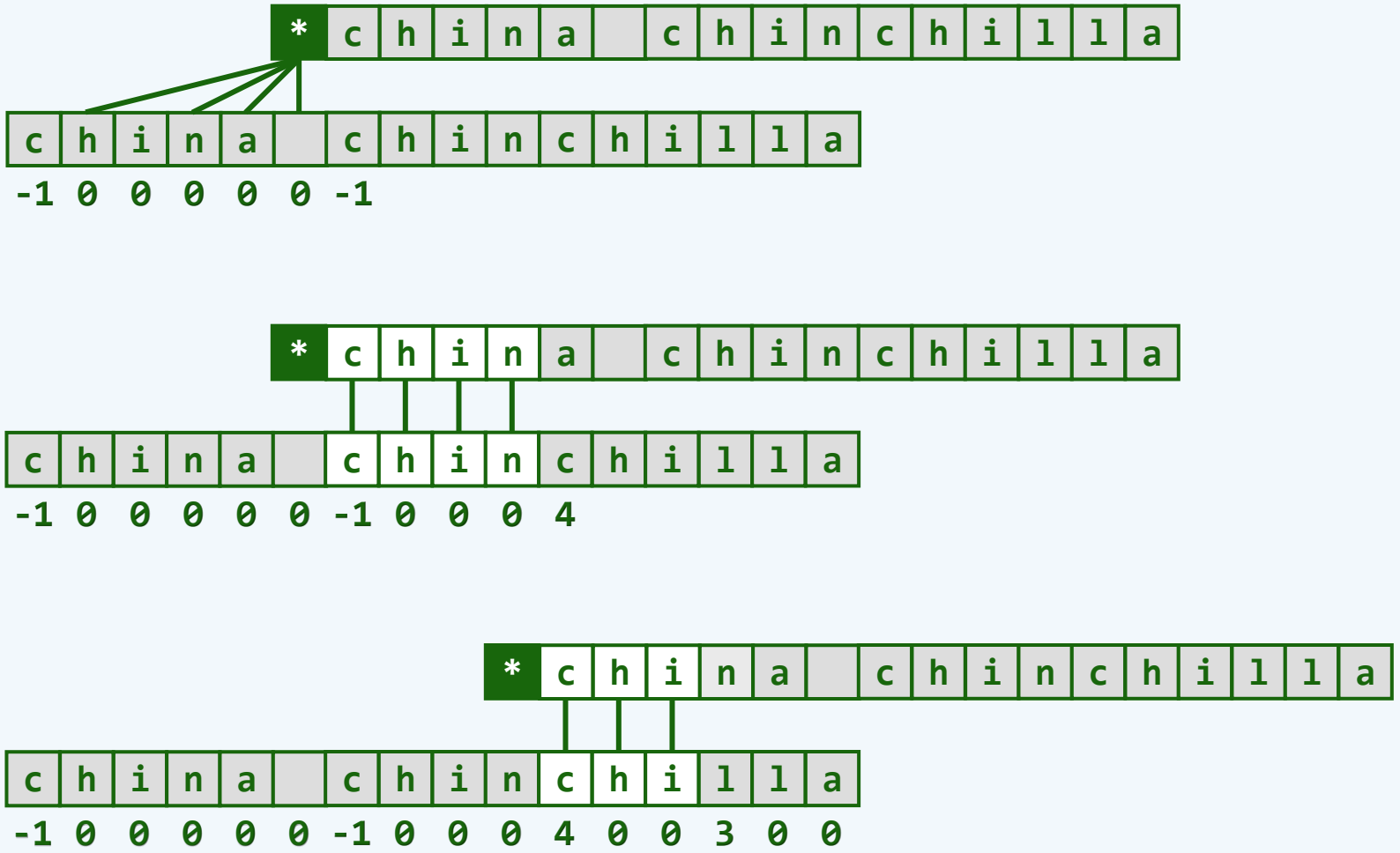


改进

```
❖int * buildNext( char * P ) {
    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
    while ( j < m - 1 )
        if ( 0 > t || P[j] == P[t] ) { //匹配
            j ++; t ++; N[j] = P[j] != P[t] ? t : N[t];
        } else //失配
            t = N[t];
    return N;
}
```



实例


* c h i n a c h i n c h i l l a
c h i n a c h i n c h i l l a
-1 0 0 0 0 0 -1
* c h i n a c h i n c h i l l a
c h i n a c h i n c h i l l a
-1 0 0 0 0 0 -1 0 0 0 4
* c h i n a c h i n c h i l l a
c h i n a c h i n c h i l l a
-1 0 0 0 0 0 -1 0 0 0 4 0 0 3 0 0

小结

- 充分利用以往的比对所提供的信息

  模式串快速右移，文本串无需回退

- 经验：以往成功的比对 —— T[i - j, i)是什么

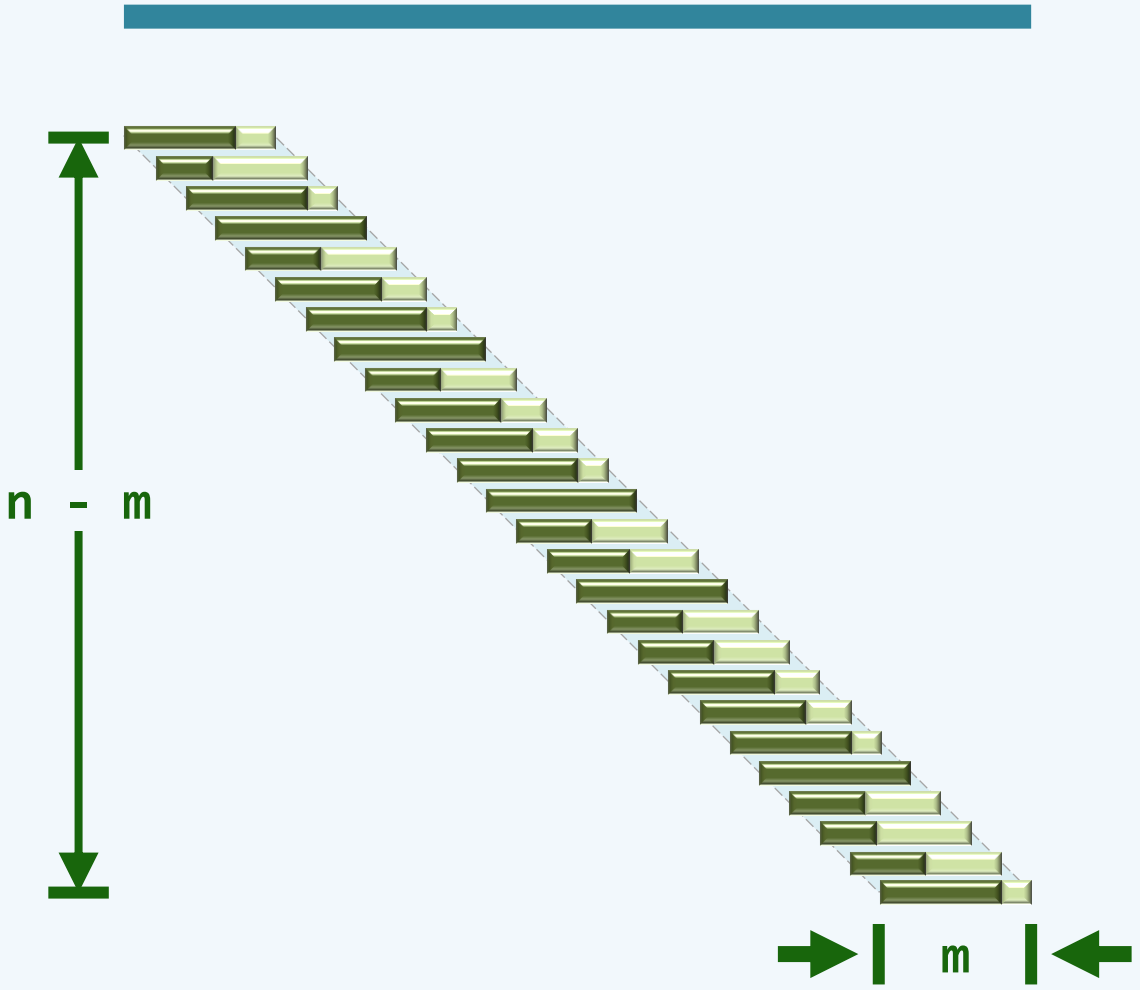
  教训：以往失败的比对 —— T[i]不是什么

- 特别适用于顺序存储介质

- 单次匹配概率越大（|Σ|越小）的场合，优势越明显 //比如二进制串

  否则，与蛮力算法的性能相差无几...

失败情况：Brute-force



失败情况：KMP