

串

KMP算法：再改进

13-C6

有颜回者好学，不迁怒，不贰过

母亲心疼地看了我好久，然后叹口气：“好吧，你这个倔强的孩子，那条路很难走，一路小心！”

邓俊辉

deng@tsinghua.edu.cn

反例

❖ 在 $T[3]$ 处

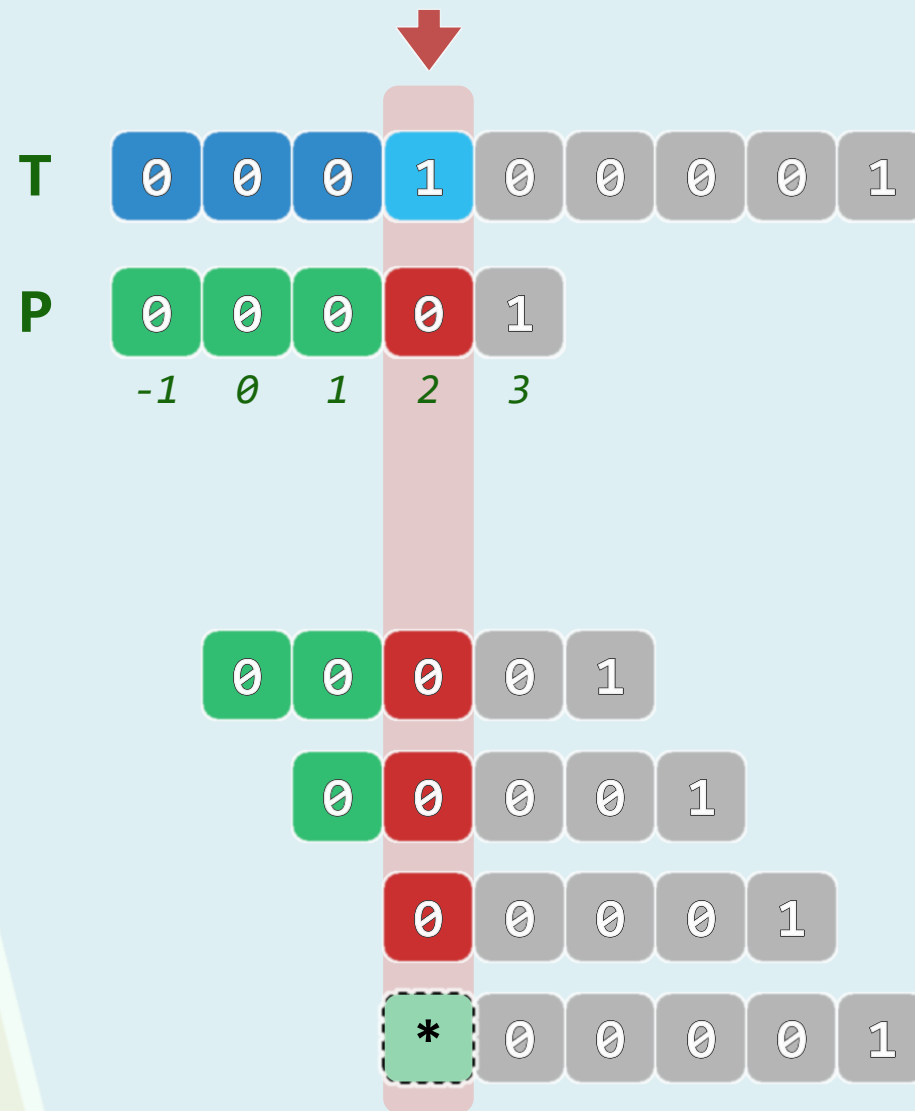
又：与 $P[3]$ 比对，失败

双：与 $P[2] = P[\text{next}[3]]$ 比对，失败

姦：与 $P[1] = P[\text{next}[2]]$ 比对，失败

姦：与 $P[0] = P[\text{next}[1]]$ 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：

$P[3] =$

$P[2] =$

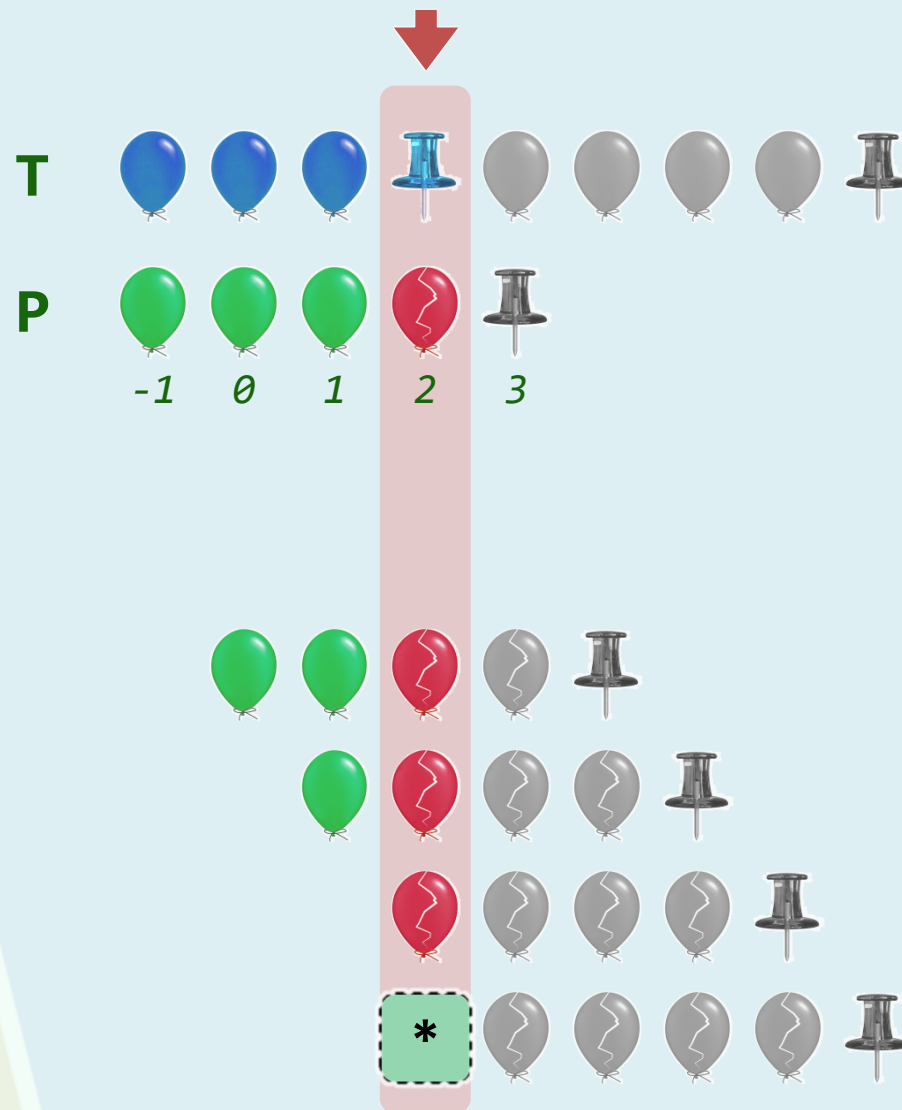
$P[1] =$

$P[0] = 0$

既然如此...

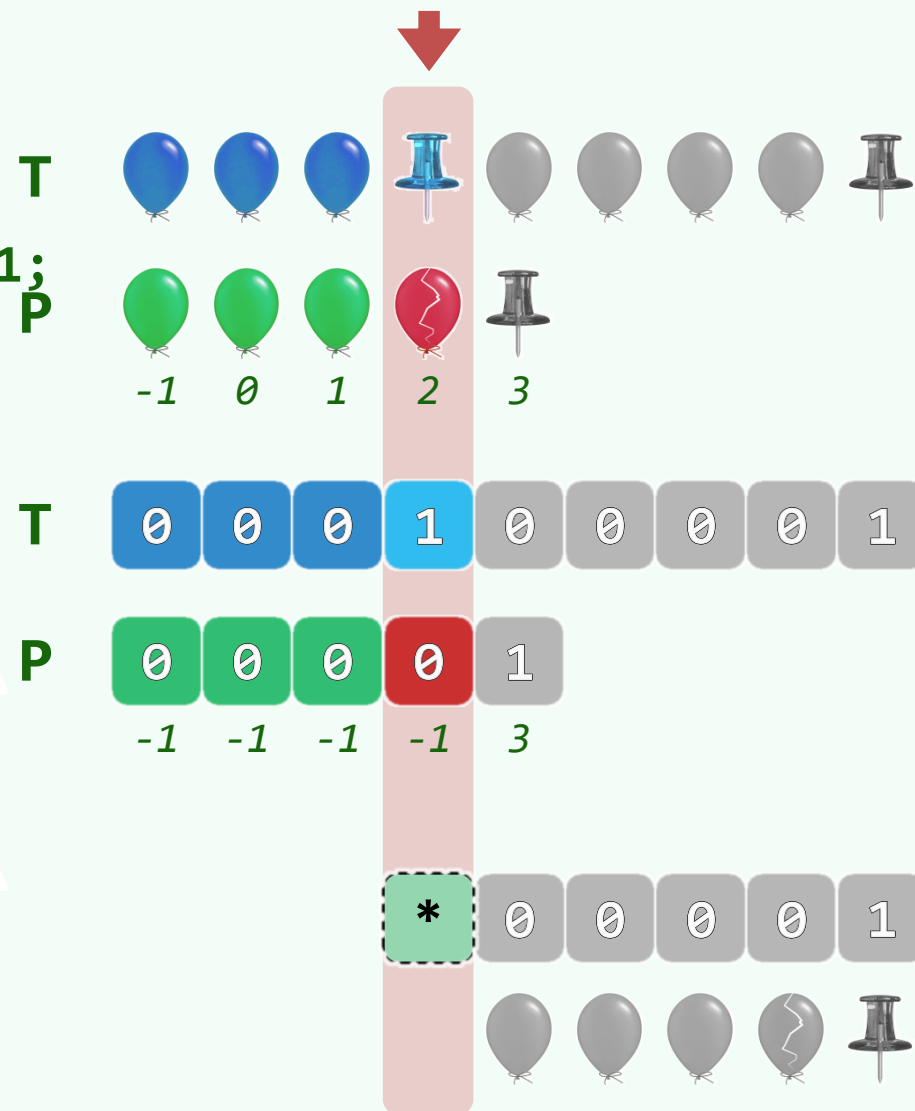
❖ 在发现 $T[3] \neq P[3]$ 之后，为何还要一错再错？

❖ 事实上，后三次比对本来都是可以避免的！



改进

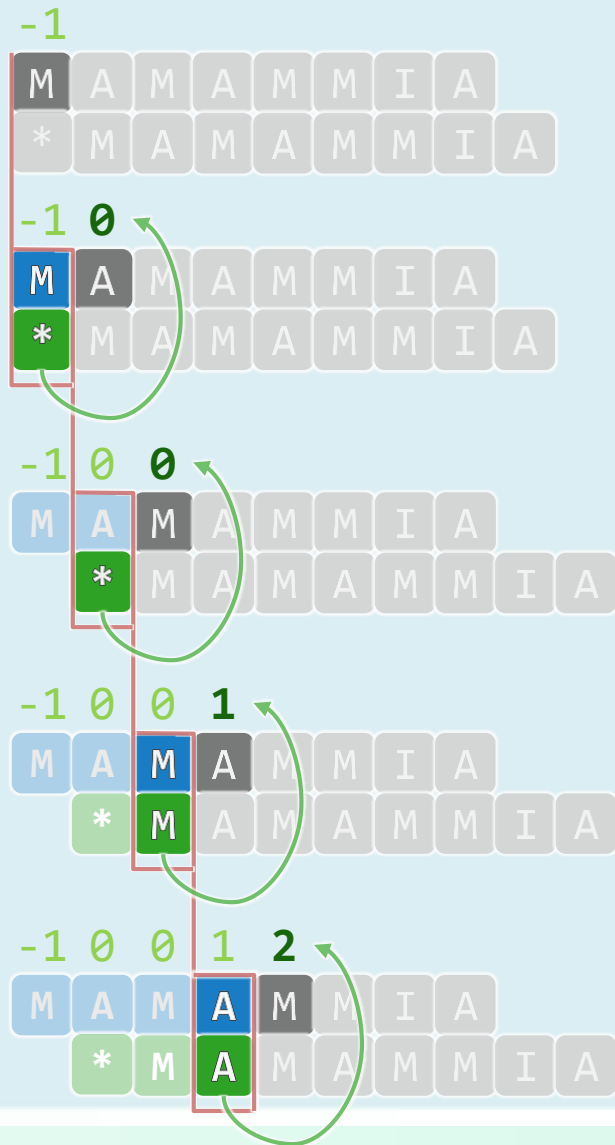
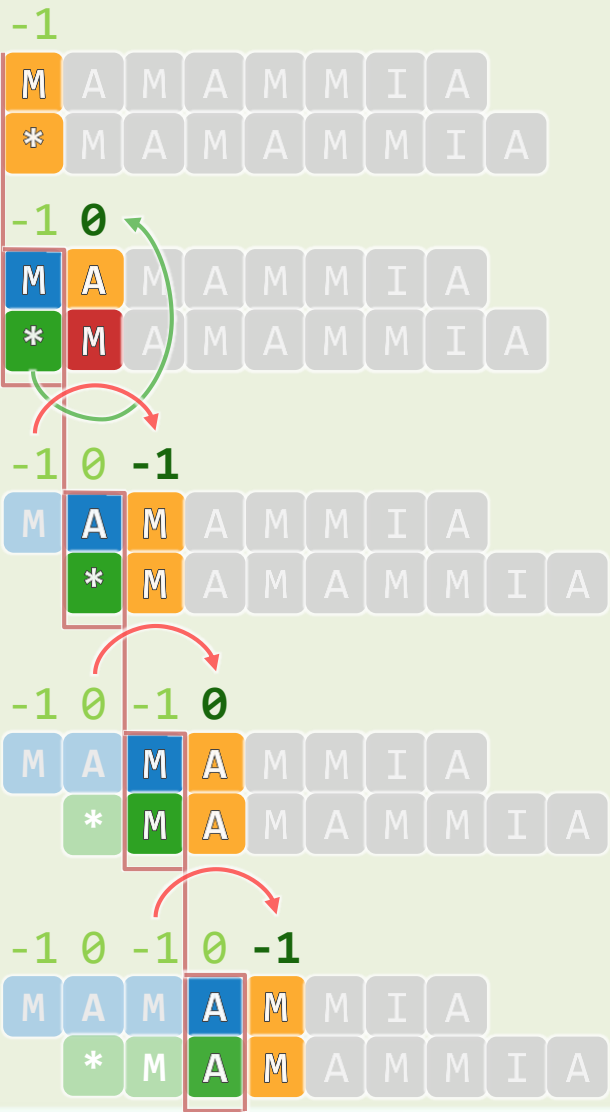
```
int* buildNext( char* P ) {  
    size_t m = strlen(P), j = 0;  
    int* next = new int[m]; int t = next[0] = -1;  
    while ( j < m - 1 )  
        if ( 0 > t || P[t] == P[j] ) {  
            if ( P[++t] != P[++j] )  
                next[j] = t;  
            else //P[next[t]] != P[t] == P[j]  
                next[j] = next[t];  
        } else  
            t = next[t];  
    return next;  
}
```



对比

-1 0 -1 0 -1 3 1 0
M A M A M M I A

-1 0 0 1 2 3 1 0
M A M A M M I A

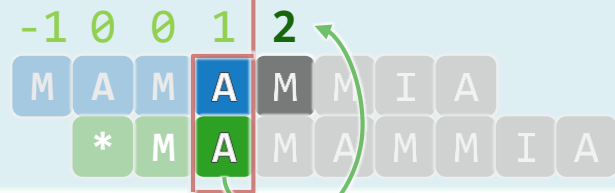
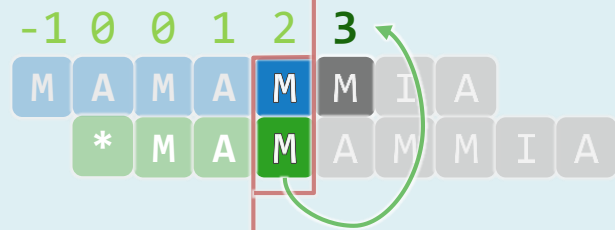
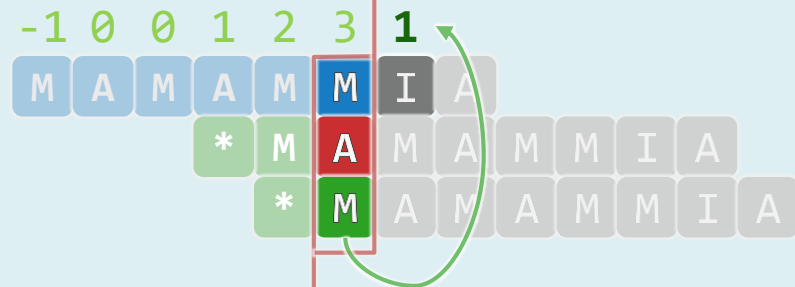
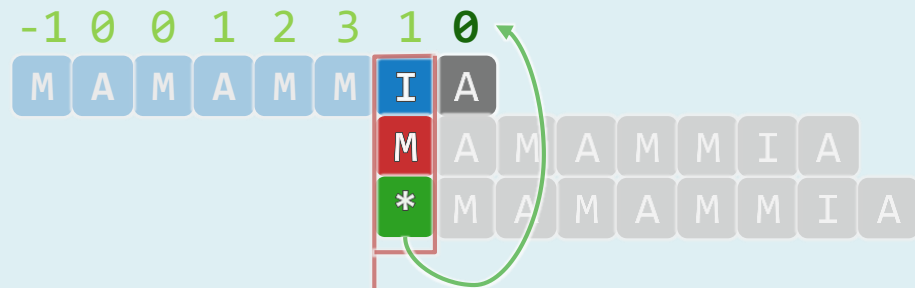
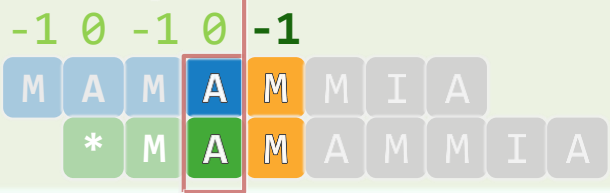
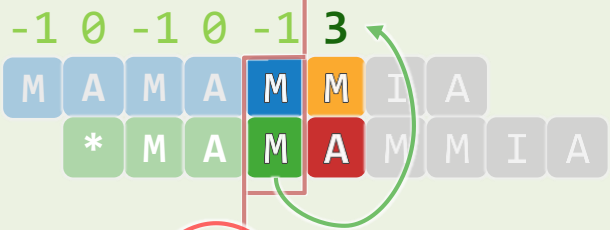
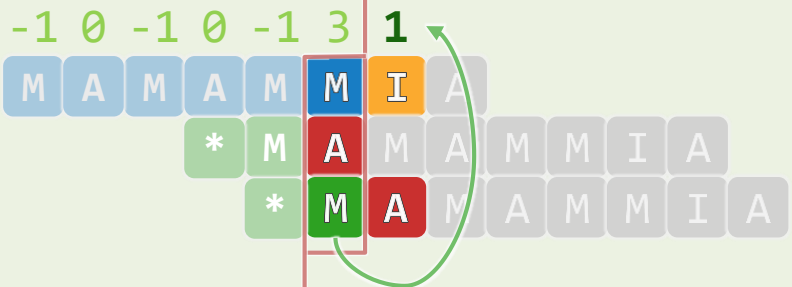
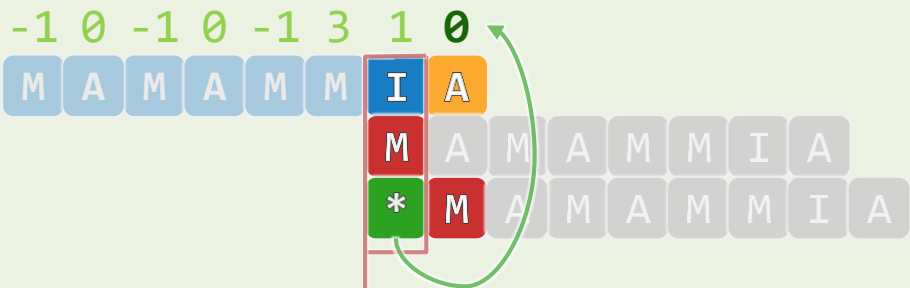


对比

-1 0 -1 0 -1 3 1 0
M A M A M M I A

-1 0 0 1 2 3 1 0

M A M A M M I A



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**（字符集越**小**），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...

