# 05-J1

# 二叉树

## Huffman编码树：问题与算法

句读之不知，惑之不解，或师焉，或不焉，小学而大遗，吾未见其明也

两年的时间，在你看来，也许就是一眨眼的功夫，对不对？可对我来说，它实在长得没边。我用不着为两年后的事情操心

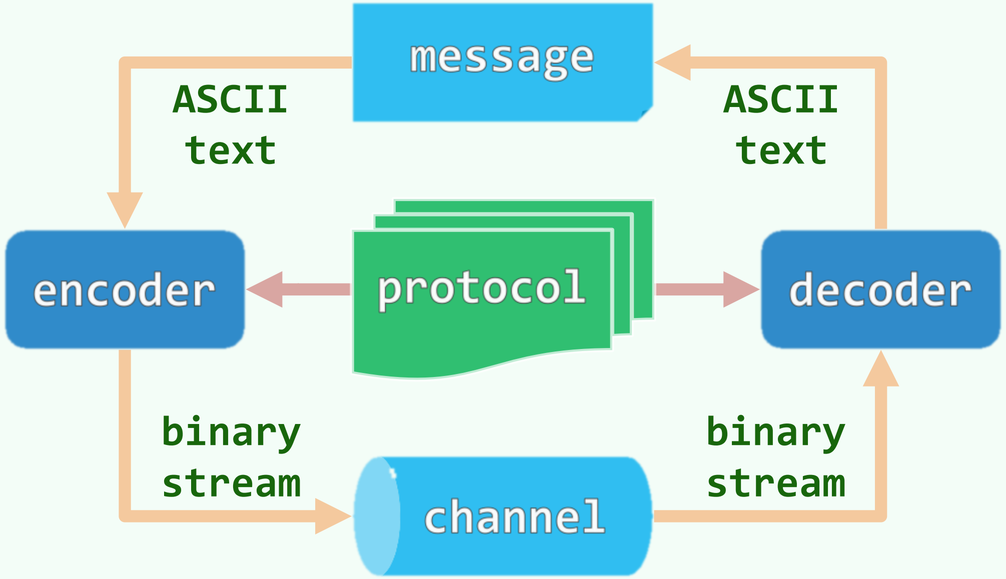
邓俊辉
deng@tsinghua.edu.cn

# 二进制编码 ~ PFC编码

❖ 通讯 / 编码 / 译码

❖ 二进制编码

- 组成数据文件的字符来自字符集Σ
- 字符被赋予**互异**的二进制串




A N

1 0 1 0 0 1 1 0 0 . . .

M I

| M | A | I | N |
|---|---|---|---|
| 1 | 010 | 011 | 00 |

❖ **句读难题**

- X ~ 01010 //某字符的编码
- Y ~ 0101 //恰是另一字符编码的**前缀**

❖ 如何避免这类**歧义**?

Prefix-Free Code!

# PFC编码 ~ 二叉*编码树

❖ Σ中的每个字符x

对应于二叉树的叶节点v(x)

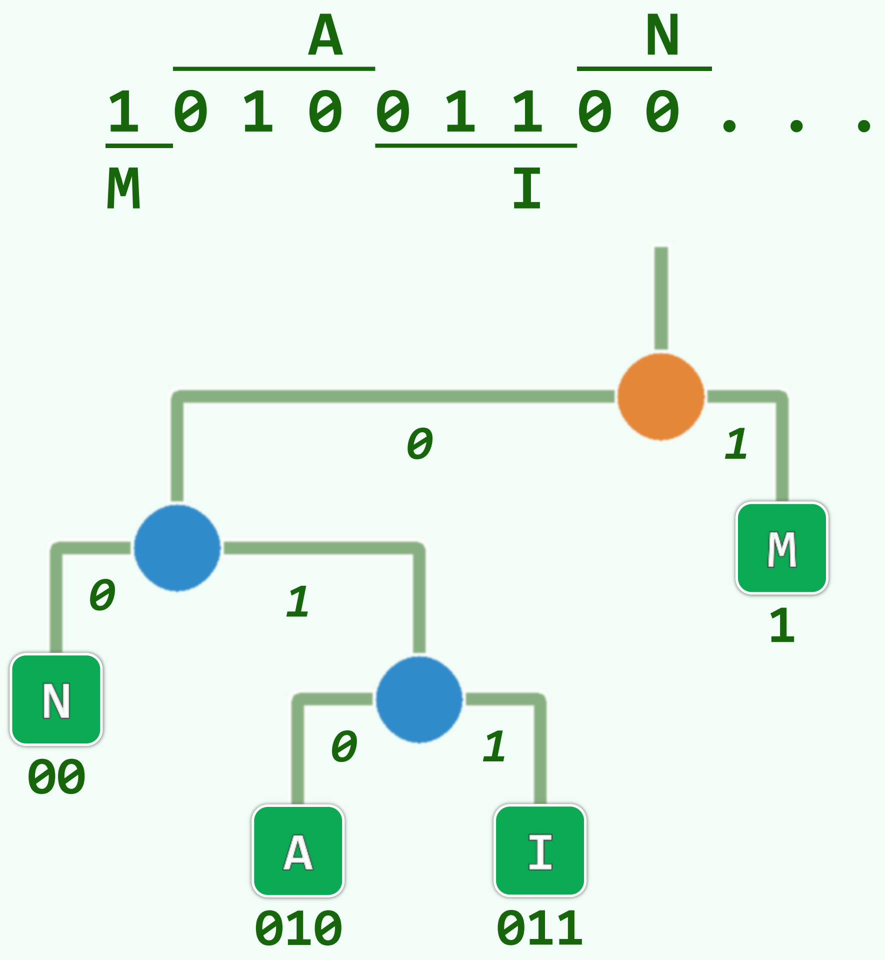
❖ x的编码串

- 由根到v(x)的通路（root path）确定
- 向左、向右分别对应于0、1

$$rps(v(x)) = rps(x)$$

❖ 如此，自然就保证了Prefix-Free

❖ PFC编码并不唯一，其中何者的编码**效率**最高?



# 编码长度 vs. 叶节点平均深度

❖ 效率的**度量**：平均编码长度

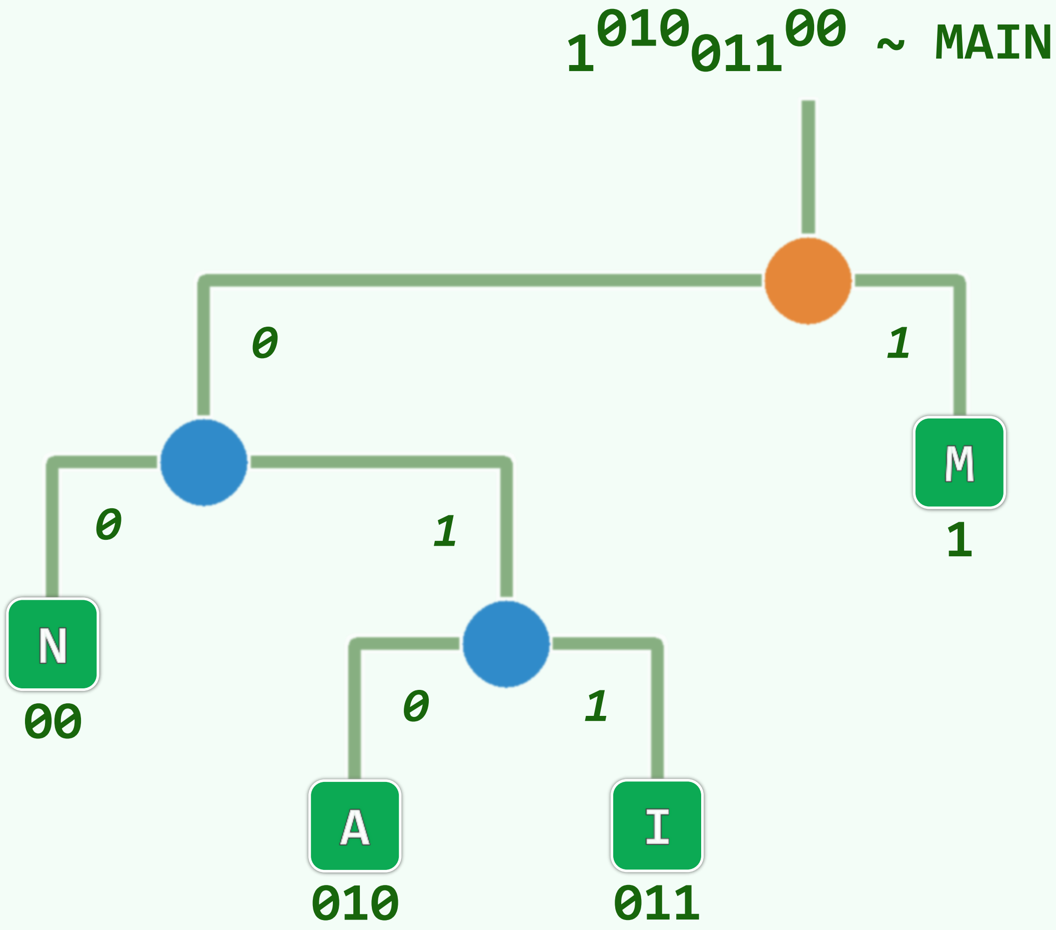
$$\text{ald}(T) \;=\; \sum\nolimits_{x\in\Sigma} \text{depth}(v(x)) \;/\; |\Sigma|$$

❖ 对于特定的字符集Σ

ald()最小者即为最优编码树$T_{opt}$

❖ 最优编码树必然**存在**，但不见得**唯一**

它们具有哪些特征？



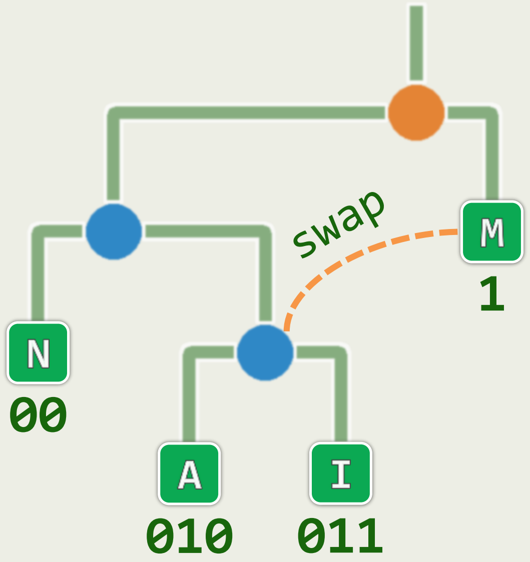
# 最优编码树

$$\forall\ v \in T_{opt},\ deg(v) = 0 \quad \textcolor{red}{\text{only if}} \quad depth(v) \geq height(T_{opt}) - 1$$

**亦即，叶子只能出现在倒数两层以内——否则，通过节点交换即可...**

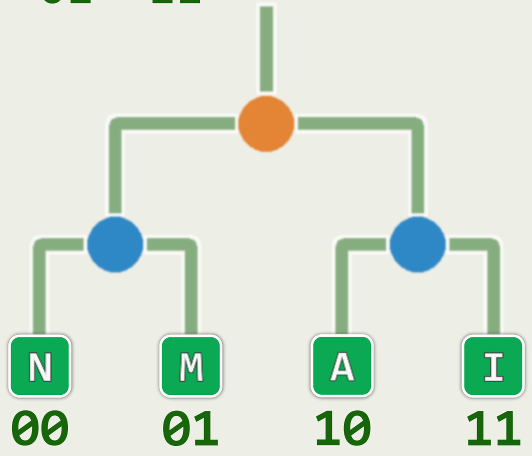
`ald(T)*4 = 2+3+3+1 = 9`

"$1^{010}011^{00}$" = "MAIN"




`ald(T)*4 = 2+2+2+2 = 8`

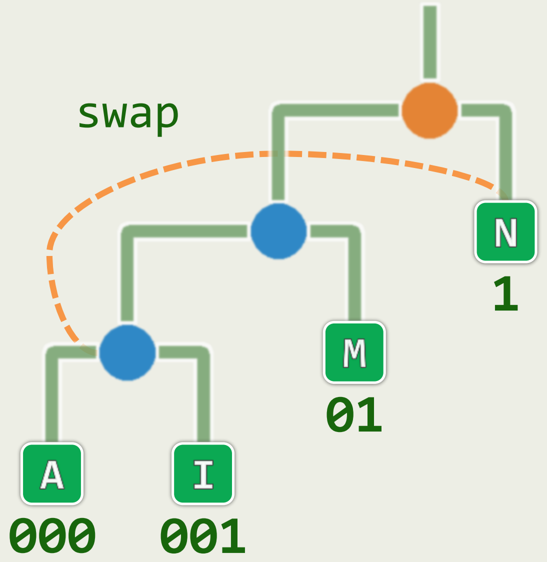
"$01^{10}11^{00}$" = "MAIN"




**特别地，真完全树即是最优编码树**

`ald(T)*4 = 2+3+3+1 = 9`

"$01^{000}001^{1}$" = "MAIN"



# 字符频率：不 ～ 埠


❖ 在不同的文化、时代、专业领域中

字符的出现**概率**或**频度**不尽相同

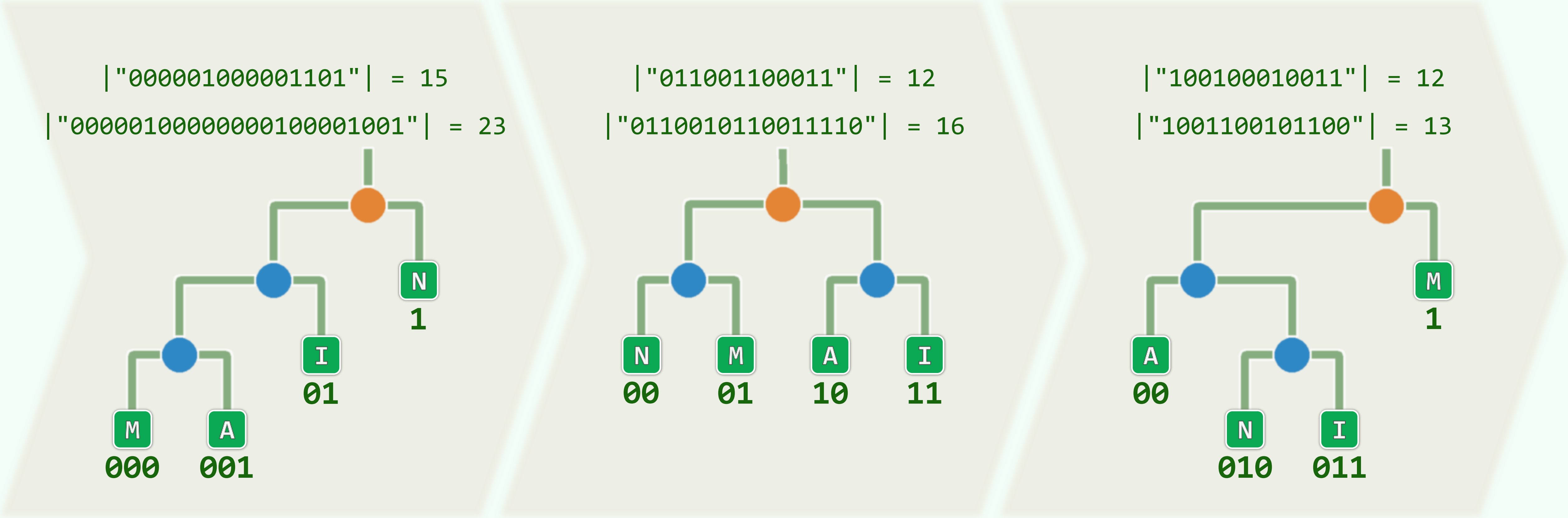
甚至，往往相差极大...


❖ 已知各字符的**期望频率**

如何构造最优编码树？

# 带权编码长度 vs. 叶节点平均带权深度

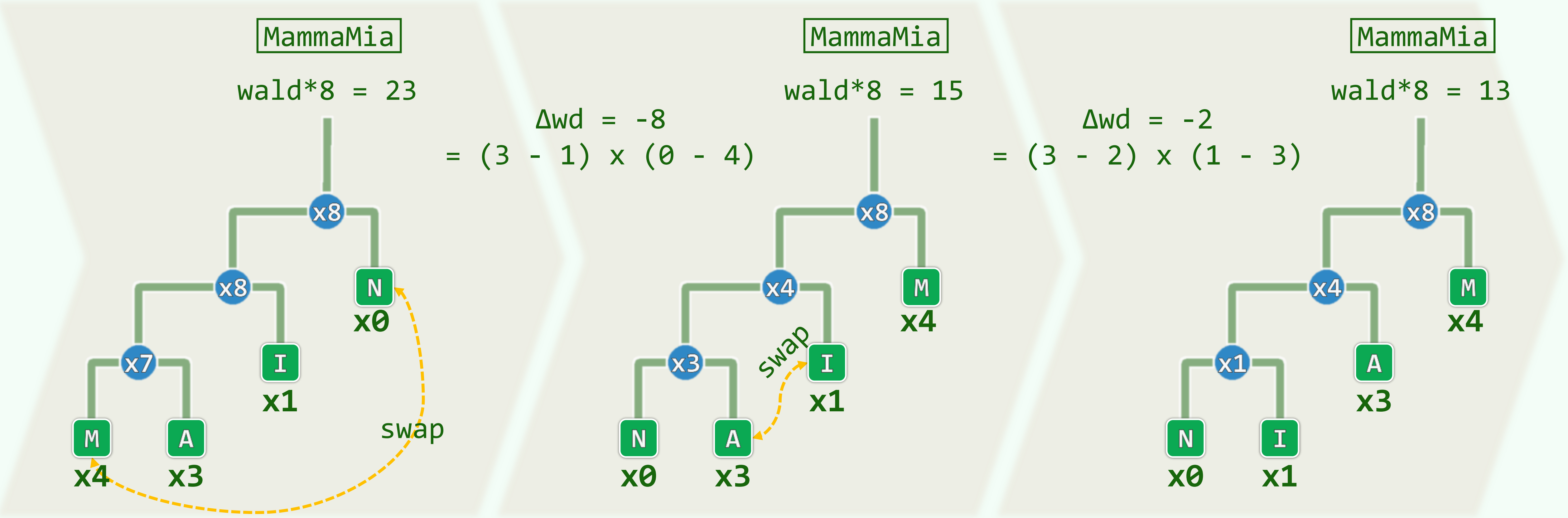
- 文件长度 $\propto$ 平均带权深度 $wald(T) = \sum_x rps(x) \times w(x)$
- 此时，完全树**未必**就是最优编码树——比如，考查"mamani"和"mammamia"...



# 最优带权编码树

❖ 同样，频率高/低的（超）字符，应尽可能放在高/低处

❖ 故此，通过适当交换，同样可以缩短wald(T)



# Huffman的贪心策略：频率低的字符优先引入，其位置亦更低

为每个**字符**创建一棵单节点的**树**，组成**森林**F

按照出现频率，对所有树**排序**

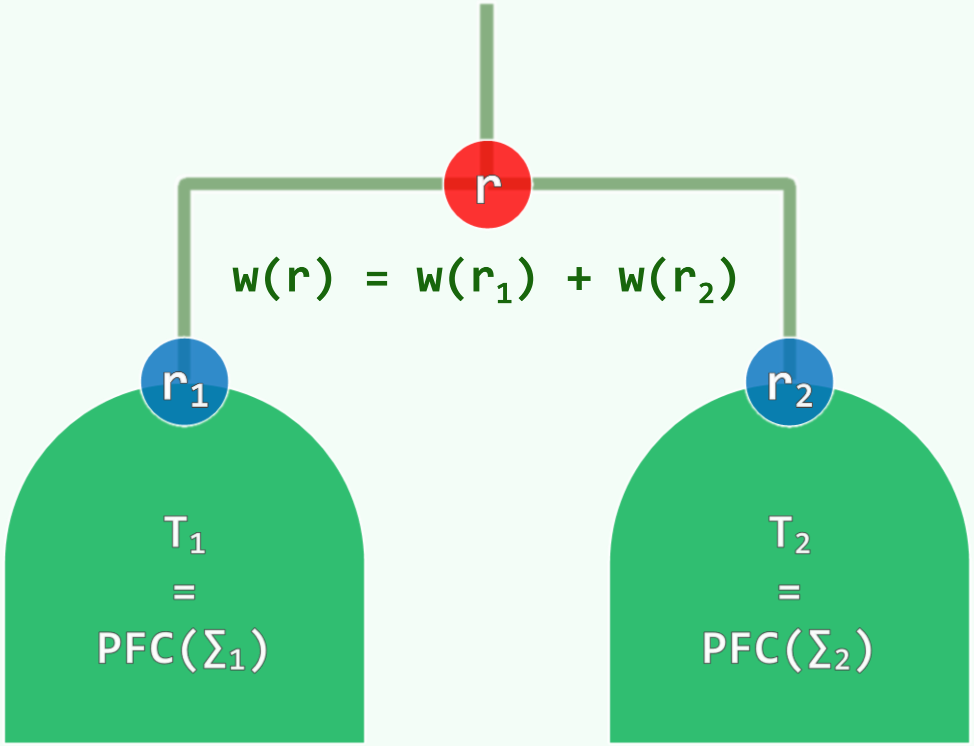
while ( F中的树不止一棵 )

取出频率**最小**的两棵树：$T_1$和$T_2$

将它们**合并**成一棵新树T，并令：

$lc(T) = T_1$ 且 $rc(T) = T_2$

$w(\ root(T)\ ) = w(\ root(T_1)\ ) + w(\ root(T_2)\ )$




//尽管贪心策略未必总能得到最优解，但非常幸运，如上算法的确能够得到最优编码树之一