

二叉树

Huffman编码树：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

编码

❖ 通讯 / 编码 / 译码

❖ 二进制编码

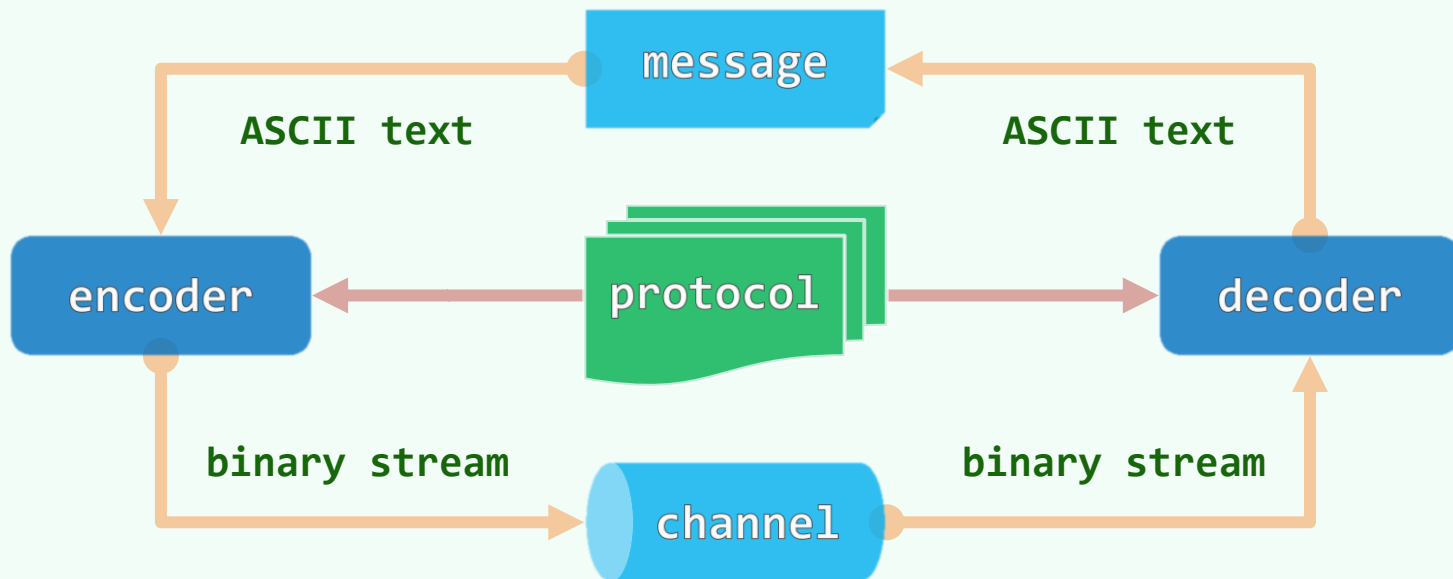
- 组成数据文件的字符来自字符集 Σ
- 字符被赋予互异的二进制串

❖ 文件的大小取决于

- 字符的数量 \times 各字符编码的长短

❖ 通讯带宽有限时

- 如何对各字符编码，使文件最小？



1⁰¹⁰ 011⁰⁰ ~ MAIN

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

PFC编码

❖ 将 Σ 中的字符组织成一棵二叉树，以0/1表示左/右孩子

各字符 x 分别存放于对应的叶子 $v(x)$ 中

❖ 字符 x 的编码串 $rpc(v(x)) = rpc(x)$

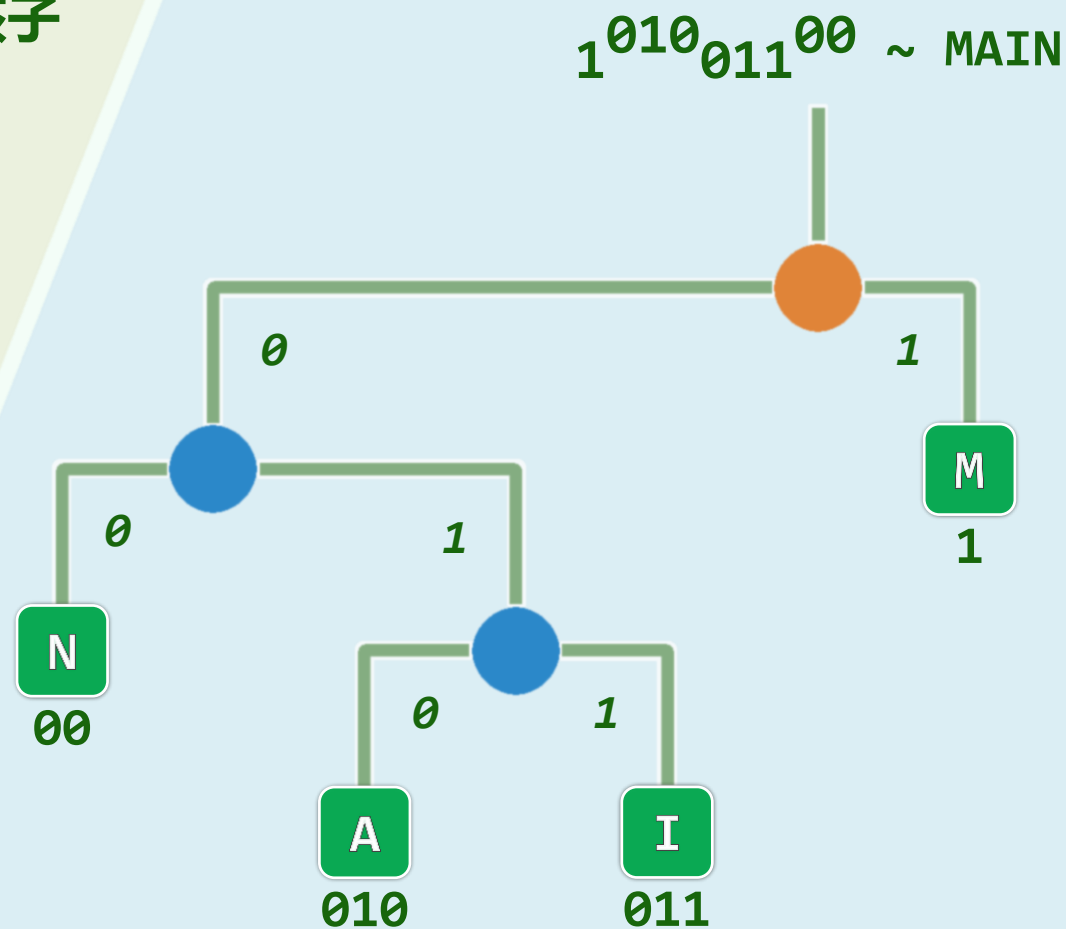
由根到 $v(x)$ 的通路 (root path) 确定

❖ 字符编码**不必**等长，而且...

❖ 不同字符的编码**互不为前缀**，故不致歧义

(Prefix-Free Code)

❖ 缺点：你能发现吗？



编码长度 vs. 叶节点平均深度

❖ 平均编码长度

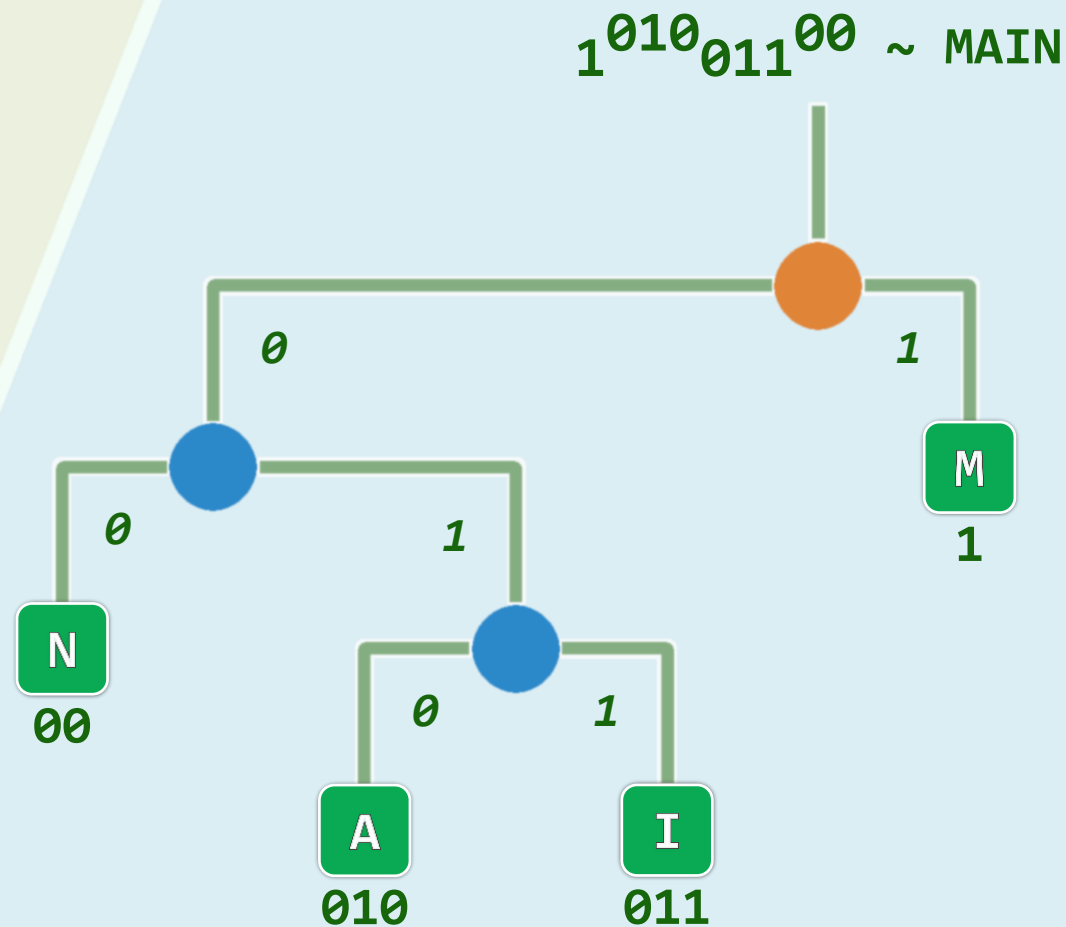
$$\text{ald}(T) = \sum_{x \in \Sigma} \text{depth}(v(x)) / |\Sigma|$$

❖ 对于特定的 Σ

$\text{ald}()$ 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？



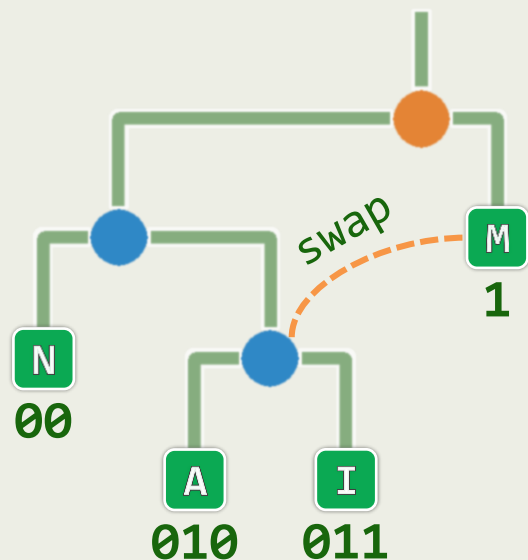
最优编码树

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

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...

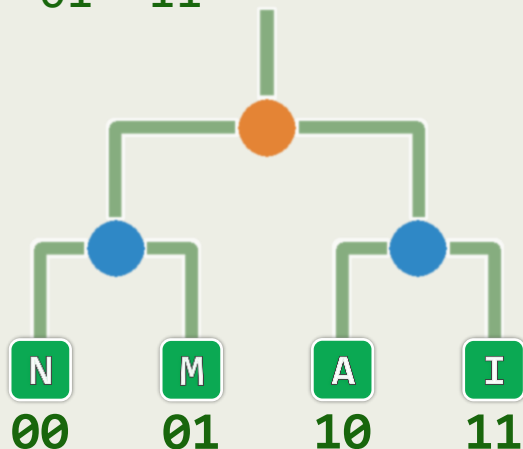
$$\text{ald}(T) * 4 = 2 + 3 + 3 + 1 = 9$$

"1⁰10₀11⁰⁰" = "MAIN"



$$\text{ald}(T) * 4 = 2 + 2 + 2 + 2 = 8$$

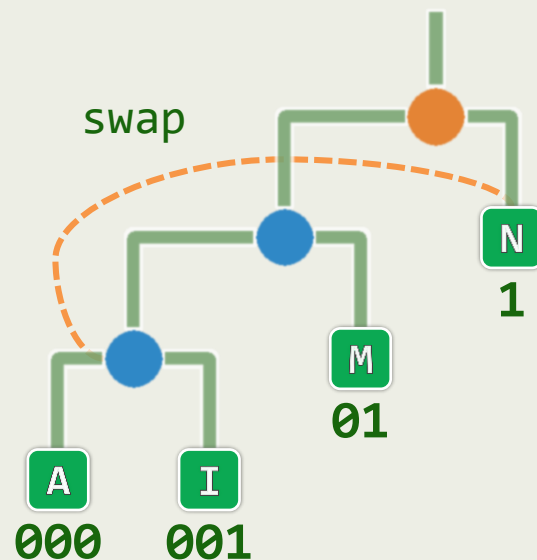
"01¹0₁11⁰⁰" = "MAIN"



特别地，**真完全树**即是**最优编码树**

$$\text{ald}(T) * 4 = 2 + 3 + 3 + 1 = 9$$

"01⁰⁰⁰001¹" = "MAIN"



字符频率

❖ 实际上，字符的出现**概率**或**频度**不尽相同

甚至，往往相差极大...



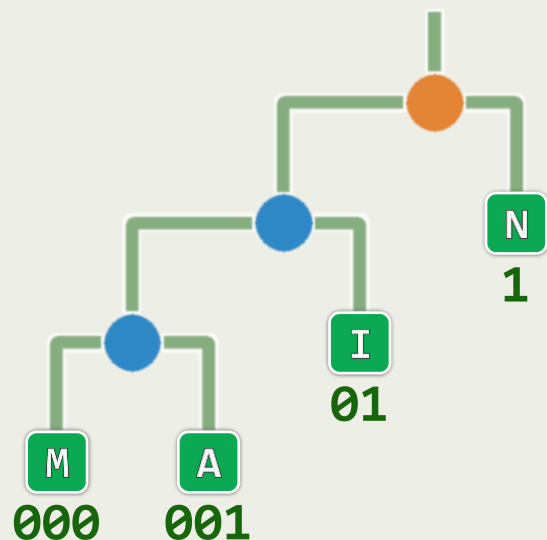
❖ 已知各字符的**期望频率**，如何构造最优编码树？

带权编码长度 vs. 叶节点平均带权深度

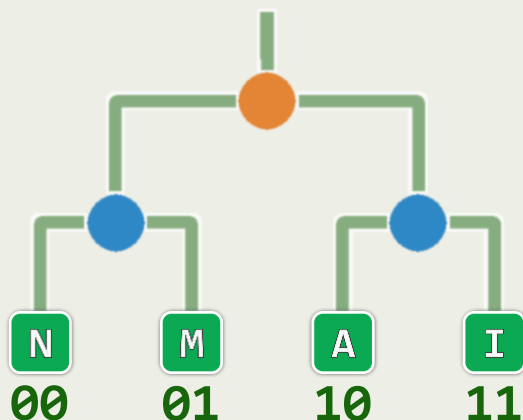
❖ 文件长度 \propto 平均带权深度 $wald(T) = \sum_x rps(x) \times w(x)$

❖ 此时，完全树**未必**就是最优编码树——比如，考查"mamani"和"mammamia"...

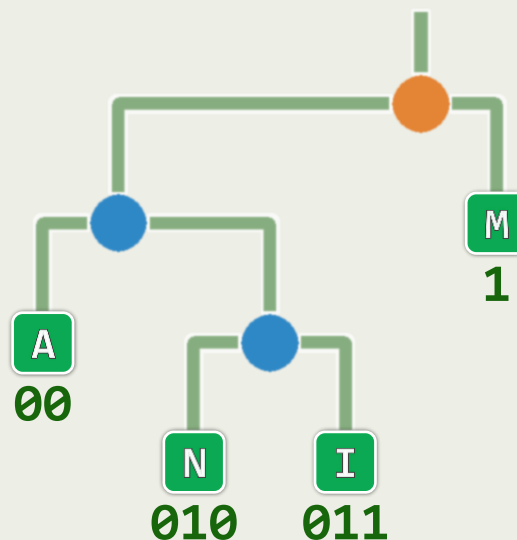
$|"000001000001101"| = 15$
 $|"00000100000000100001001"| = 23$



$|"011001100011"| = 12$
 $|"0110010110011110"| = 16$

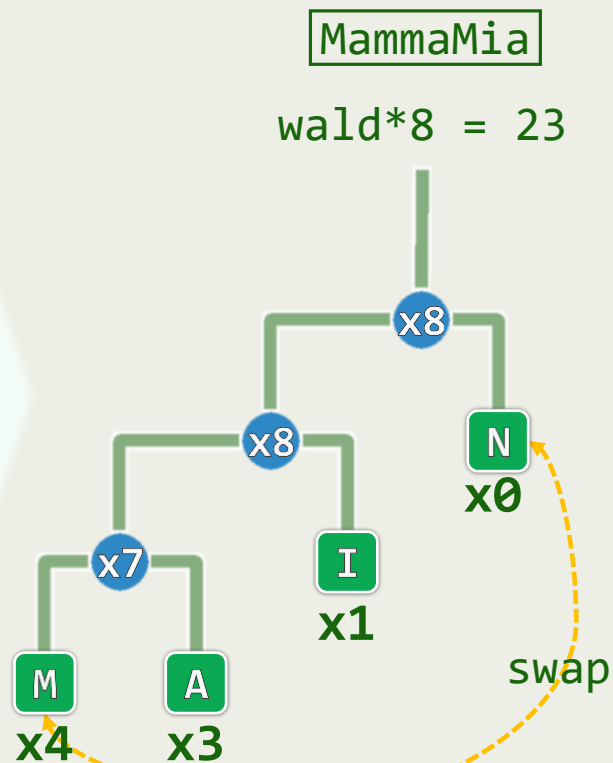


$|"100100010011"| = 12$
 $|"1001100101100"| = 13$

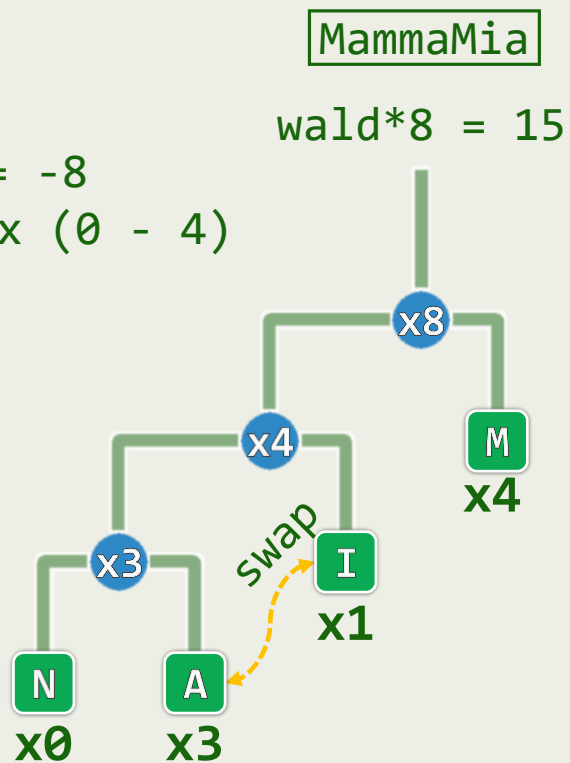


最优带权编码树

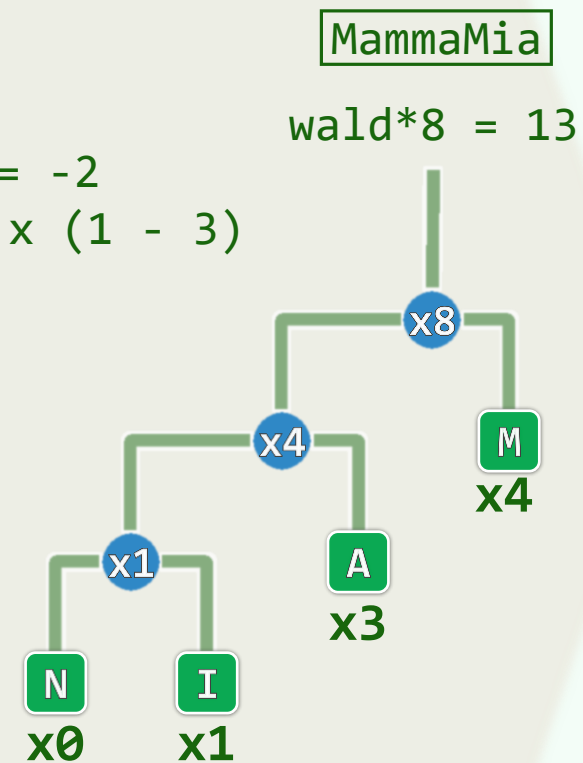
- ❖ 同样，频率高/低的（超）字符，应尽可能放在高/低处
- ❖ 故此，通过适当交换，同样可以缩短 $wald(T)$



$$\Delta wd = -8$$
$$= (3 - 1) \times (0 - 4)$$



$$\Delta wd = -2$$
$$= (3 - 2) \times (1 - 3)$$



Huffman算法

// 贪婪策略：频率低的字符优先引入，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：

$$lc(T) = T_1 \text{ 且 } rc(T) = T_2$$

$$w(\text{root}(T)) = w(\text{root}(T_1)) + w(\text{root}(T_2))$$

// 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

