二叉树

Huffman编码树：算法

05-J1

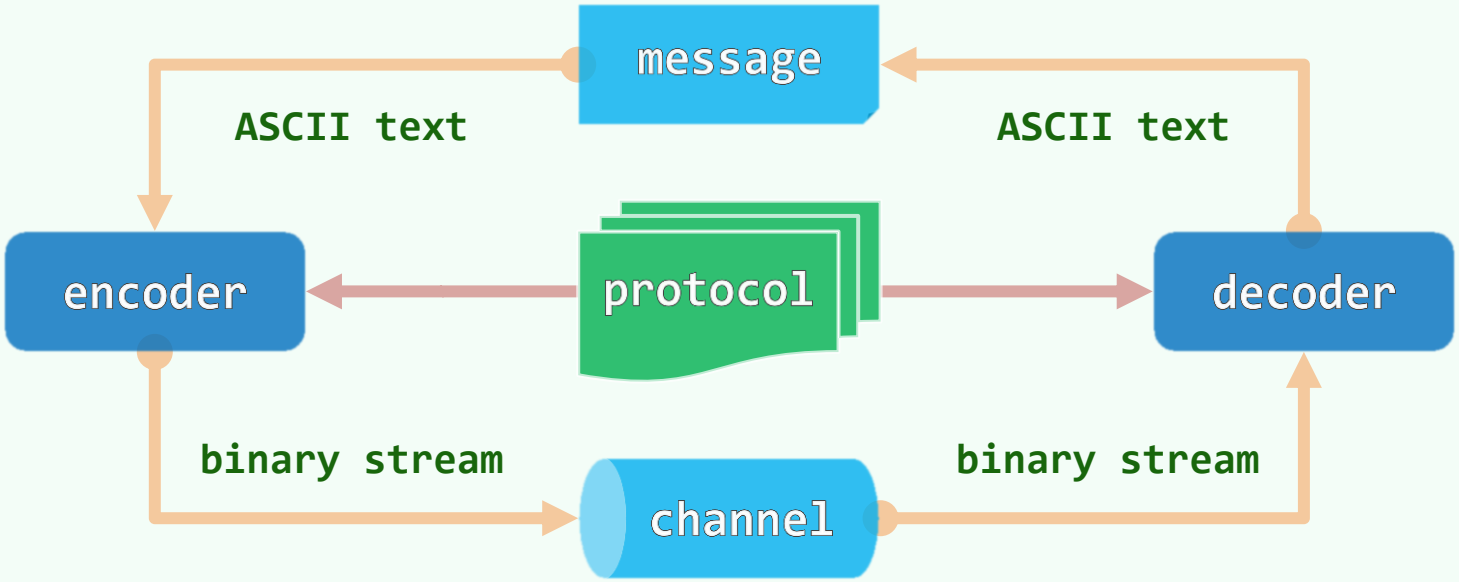
句读之不知，惑之不解，或师焉，或不焉，小学而大遗，吾未见其明也

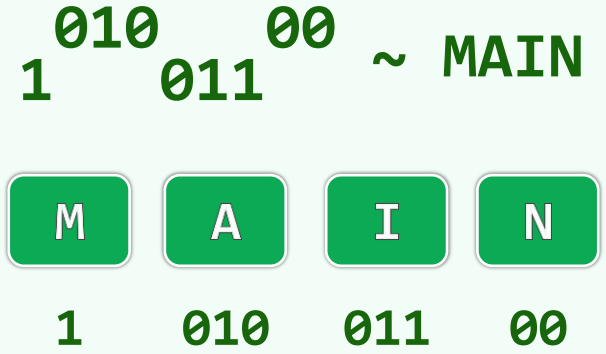
两年的时间，在你看来，也许就是一眨眼的功夫，对不对？可对我来说，
它实在长得没边。我用不着为两年后的事情操心。

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 × 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？





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

  由根到v(x)的通路（root path）确定

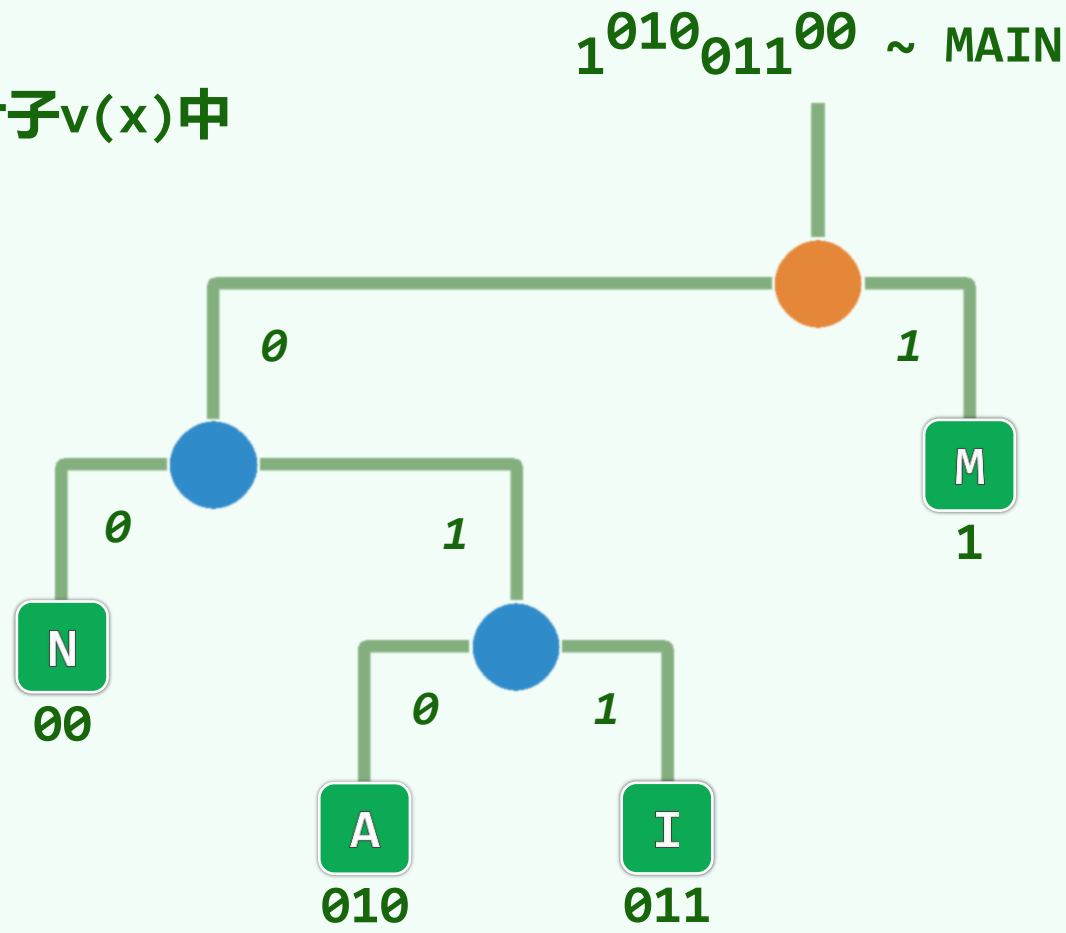
- 优点：字符编码不必等长，且

  不致出现解码歧义

- 这属于“前缀无歧义”编码/Prefix-Free Code

  不同字符的编码互不为前缀，故不致歧义

- 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

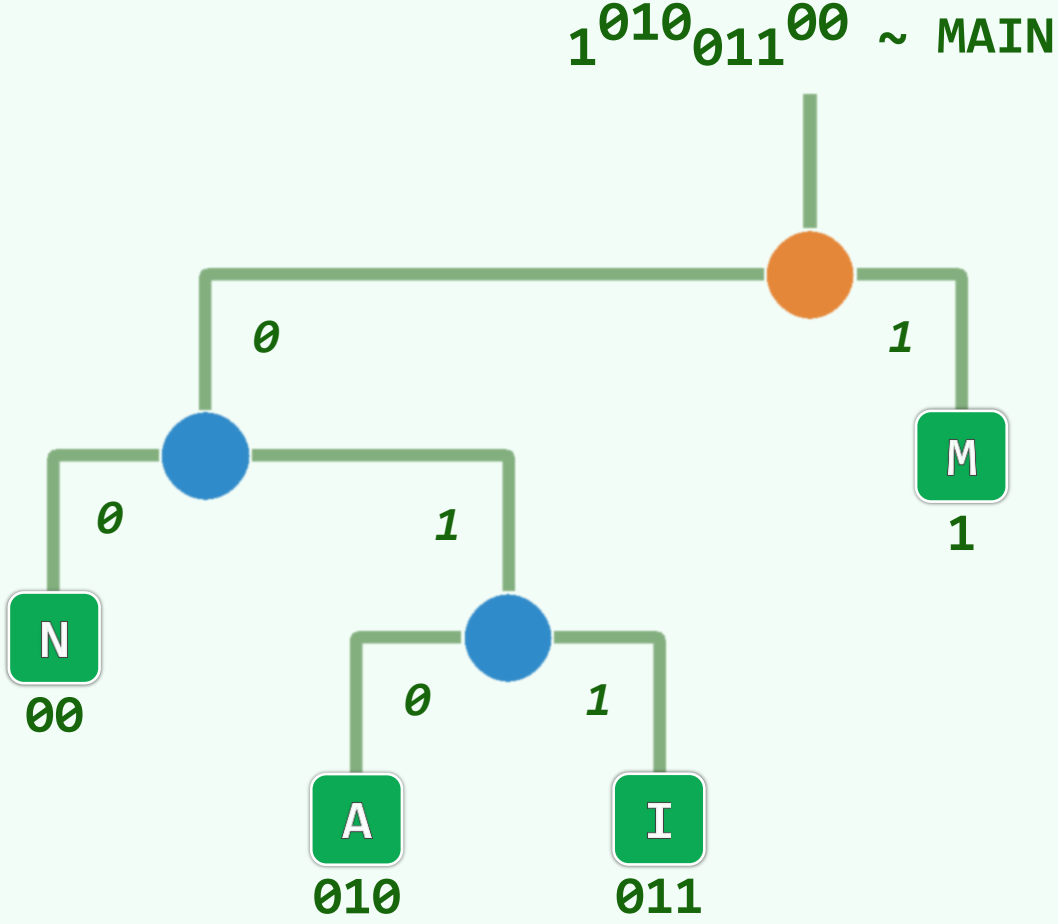
$$\mathrm{ald}(T) = \sum_{x \in \Sigma} \mathrm{depth}(v(x))$$

- 对于特定的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？



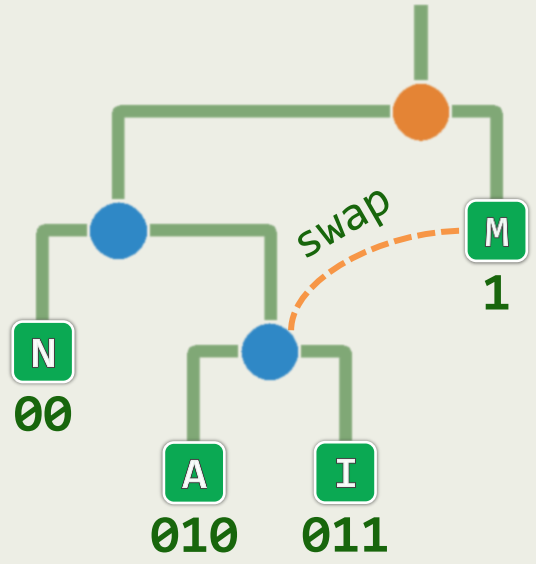
最优编码树

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

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...

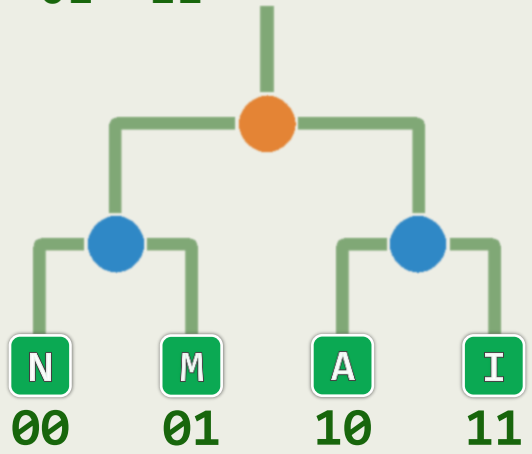
ald(T)*4 = 2+3+3+1 = 9

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



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

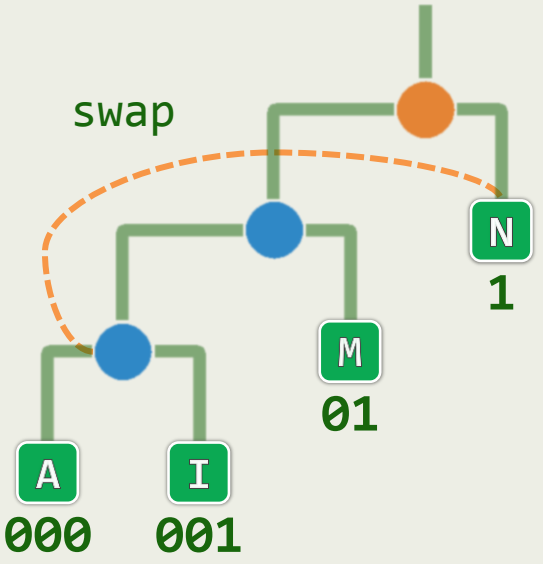
"$01^{10}_{11}{}^{00}$" = "MAIN"



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

❖ 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...

序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
字	不	人	山	无	风	一	日	云	有	何	来	天	中	时	花	上	水	春	月	相	年	为	生	君	长	心	自	如	知	白	归	秋
次	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1													
	6	0	6	5	5	5	4	3	2	2	2	2	1	1	1	1	1	1	1	9	9	9	9	9	9	9	9	8	8	8	8	8
	4	9	0	8	7	2	9	4	7	3	3	2	4	3	3	1	0	0	0	6	5	5	4	3	1	0	0	7	6	5	3	1
	8	9	8	2	1	7	8	2	0	7	1	0	5	7	6	4	9	9	0	6	6	0	2	2	8	7	2	5	7	3	6	2
	2	5	0	3	8	2	4	1	1	9	5	2	9	7	7	8	7	6	4	6	2	6	8	5	5	8	5	1	7	0	4	1

❖ 已知各字符的期望频率，如何构造最优编码树？

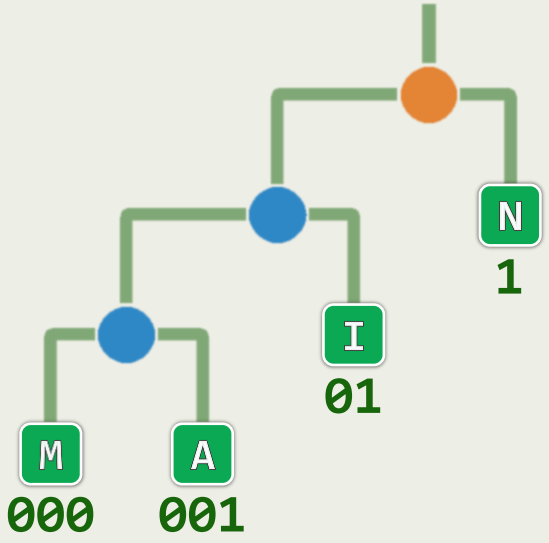
带权编码长度 vs. 叶节点平均带权深度

❖ 文件长度 ∝ 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"...

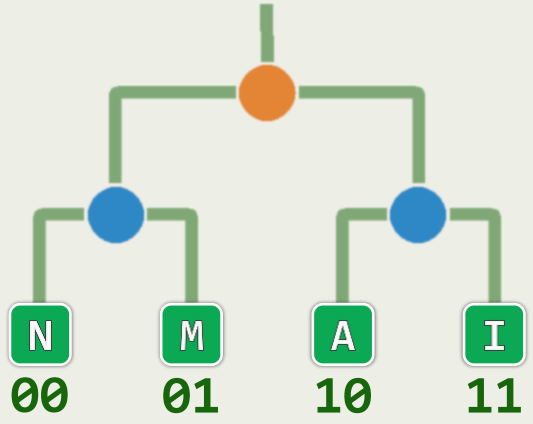
|"000001000001101"| = 15

|"000001000000000100001001"| = 23



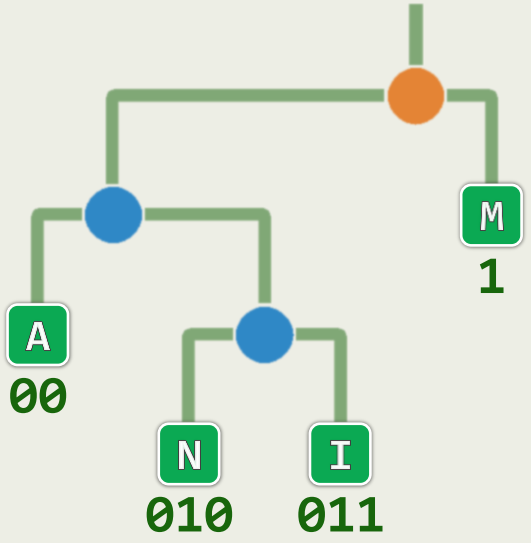
|"011001100011"| = 12

|"0110010110011110"| = 16



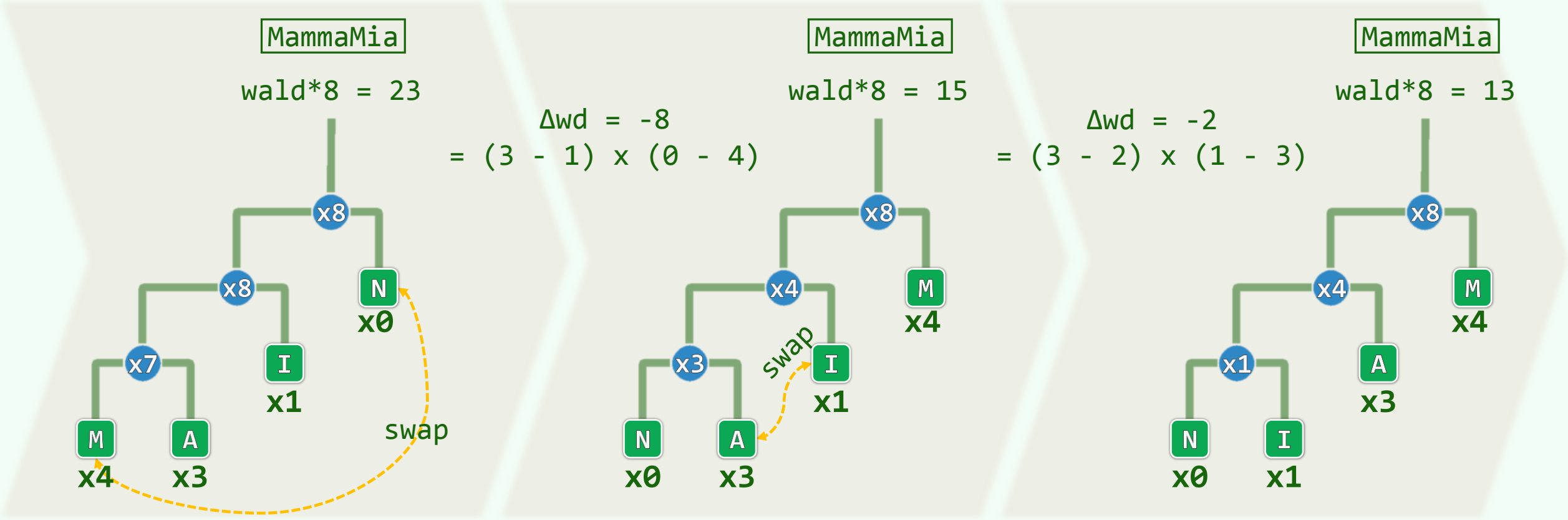
|"100100010011"| = 12

|"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)



Huffman算法

// 贪婪策略：频率低的字符优先引入，从而使其位置更低

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

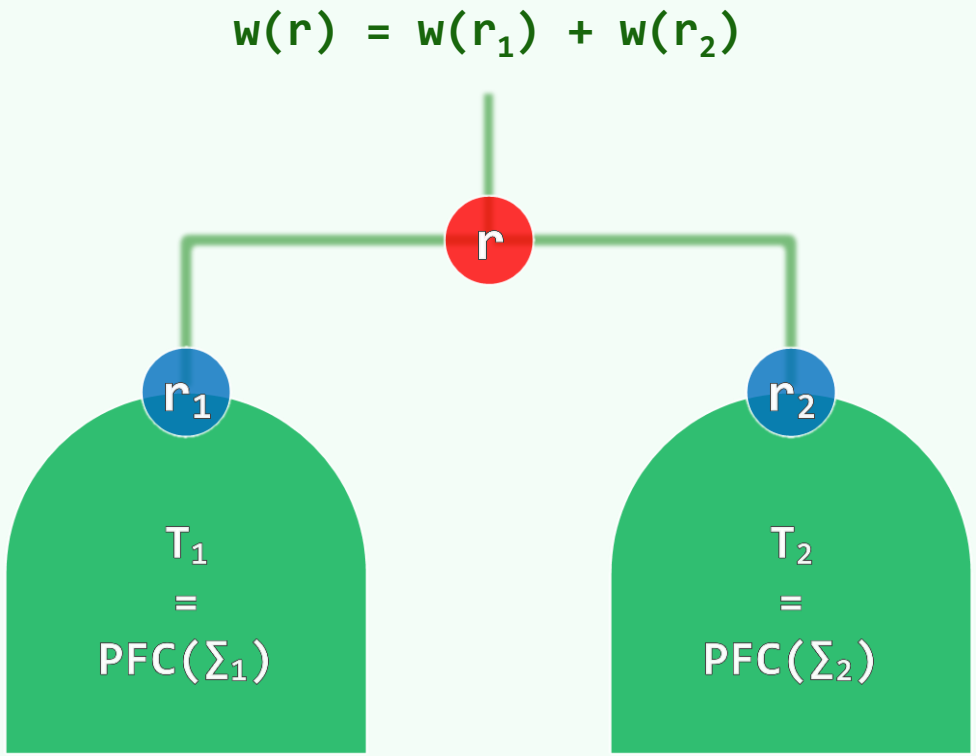
while (F中的树不止一棵)

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：

$lchild(T) = T_1$ 且 $rchild(T) = T_2$

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



// 尽管贪心策略未必总能得到最优解，但这里非常幸运，如上的确能够得到最优编码树之一