

Design and Analysis of Algorithms

Part III: Greedy Algorithms

Lecture 18: Huffman Coding

盛浩

shenghao@buaa.edu.cn

北京航空航天大学
计算机学院

北航《算法设计与分析》



贪心策略篇概述

- 在算法课程第三部分“贪心策略”主题中，我们将主要聚焦于如下经典问题：
 - Fractional Knapsack Problem (部分背包问题)
 - Huffman Coding Problem (赫夫曼编码问题)
 - Activity Selection Problem (活动选择问题)



贪心策略篇概述

- 在算法课程第三部分“贪心策略”主题中，我们将主要聚焦于如下经典问题：
 - Fractional Knapsack Problem (部分背包问题)
 - Huffman Coding Problem (赫夫曼编码问题)
 - Activity Selection Problem (活动选择问题)

问题背景：字符编码

- 在计算机中，常用二进制串对不同字符进行编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式	000	001	010	011	100	101

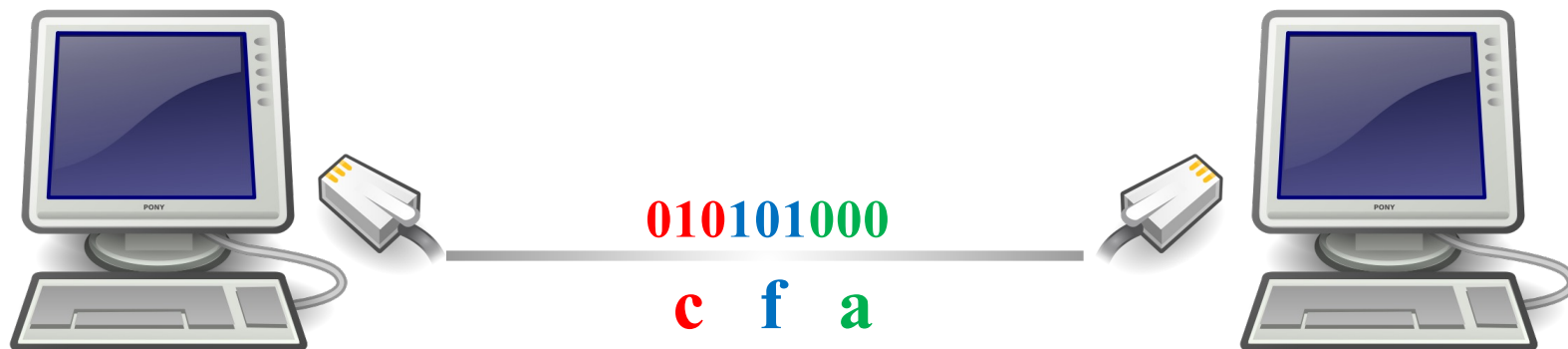




问题背景：字符编码

- 在计算机中，常用二进制串对不同字符进行编码

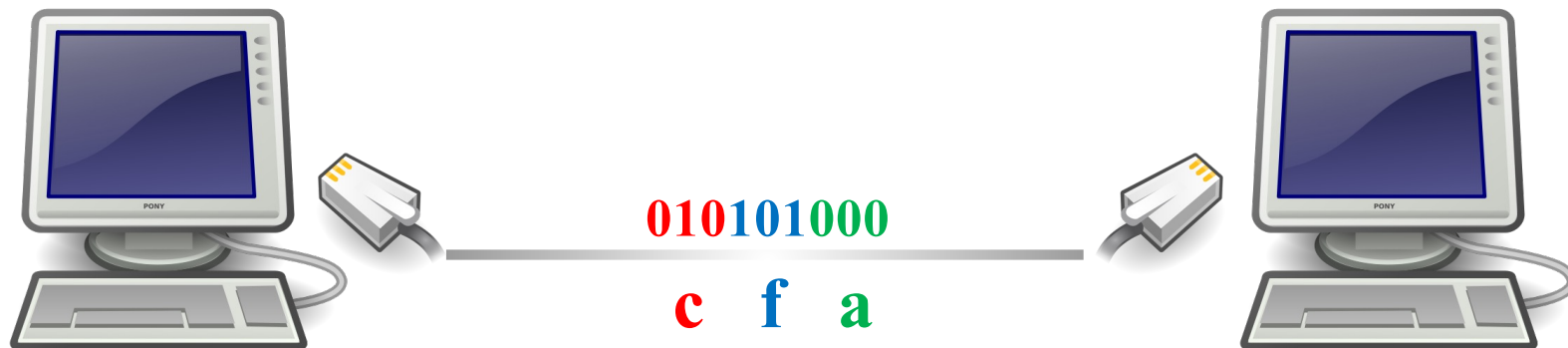
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式	000	001	010	011	100	101



问题背景：字符编码

- 在计算机中，常用二进制串对不同字符进行编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式	000	001	010	011	100	101



问题：哪些编码方式是可行的？

问题背景：字符编码



	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码
 - 编码：*cfa* → 100



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码
 - 编码：*cfa* → 1001100



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码
 - 编码：*cf***a** → 1001100**0**



问题背景：字符编码

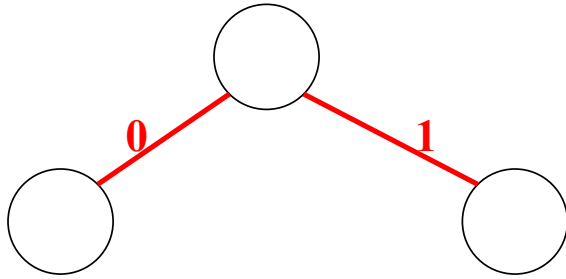
	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码
 - 编码：*cfa* → 10011000
 - 解码：10011000 → ?



问题背景：字符编码

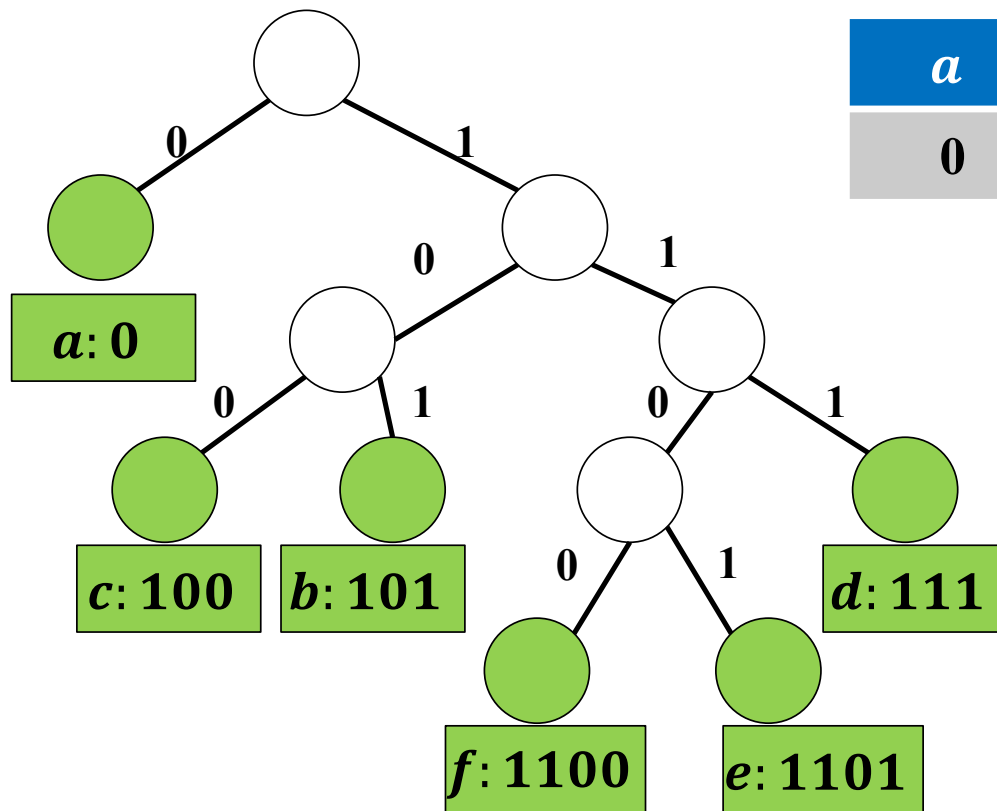
- 编码树
 - 顶点到左结点的边**标记0**，到右结点的边**标记1**



问题背景：字符编码

- 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树



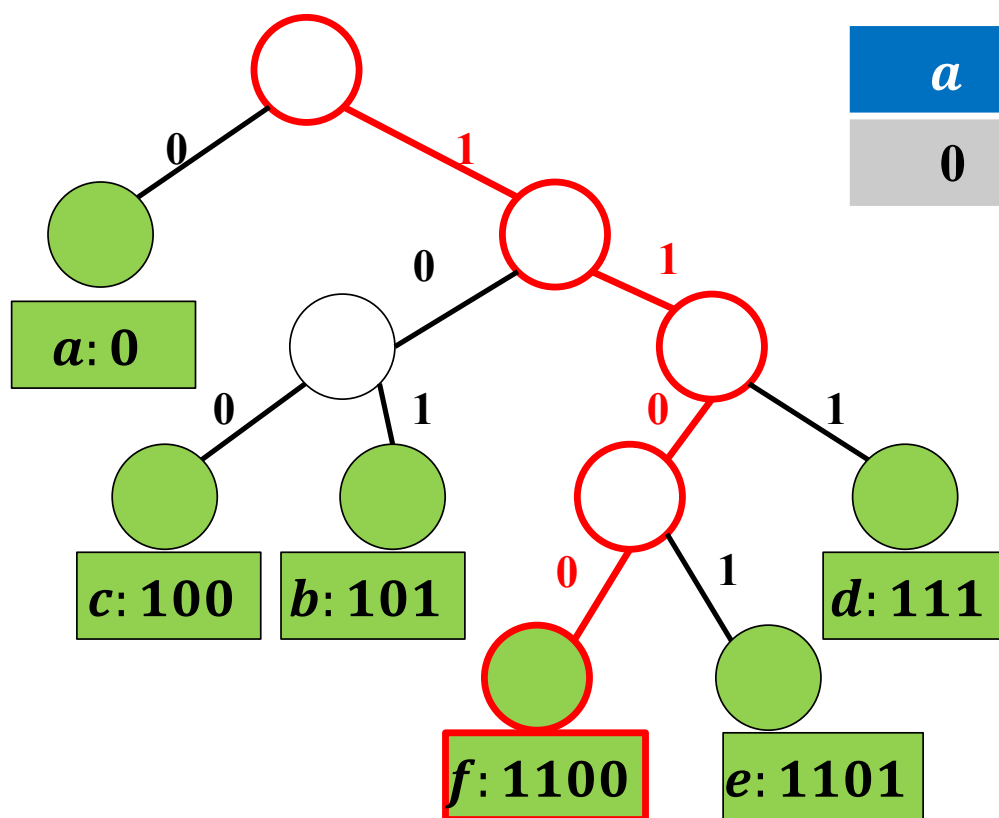
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100



问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



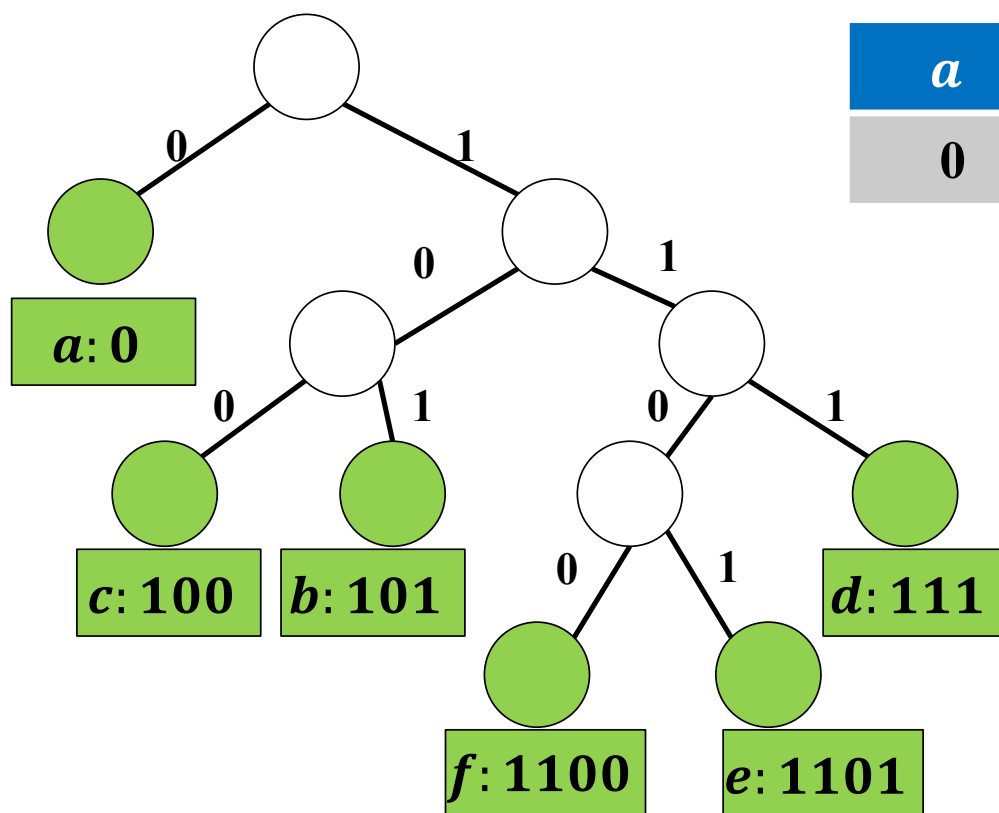
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

1100表示*f*

问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



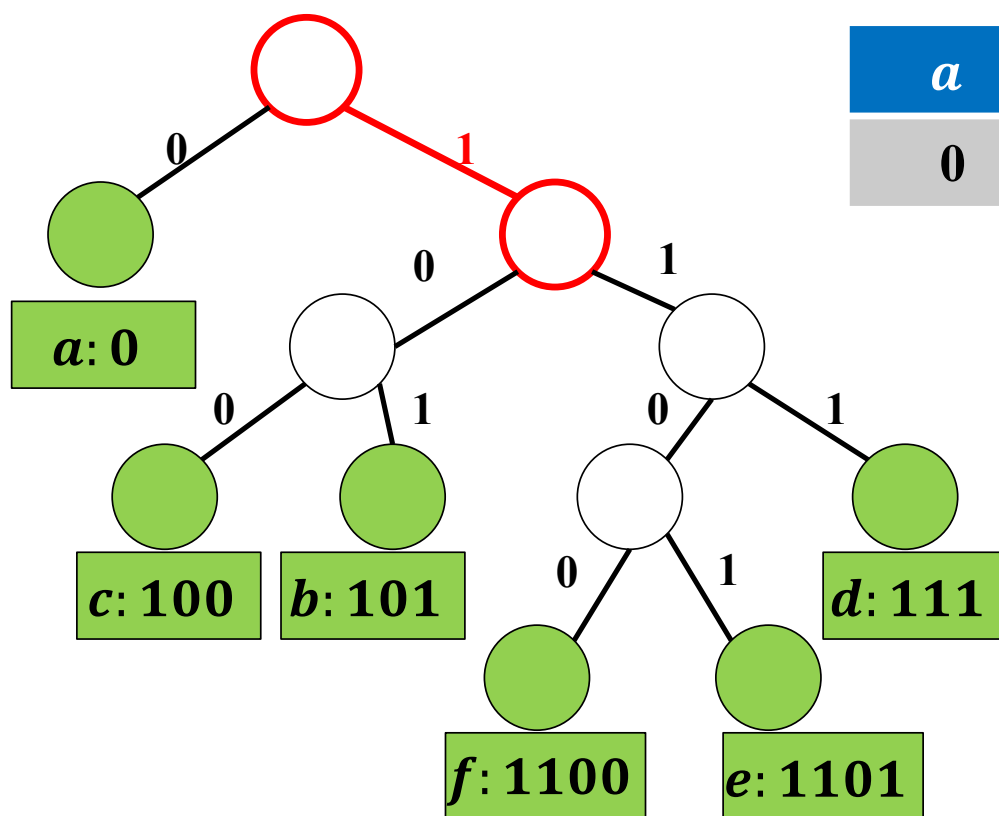
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

解析1101

问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



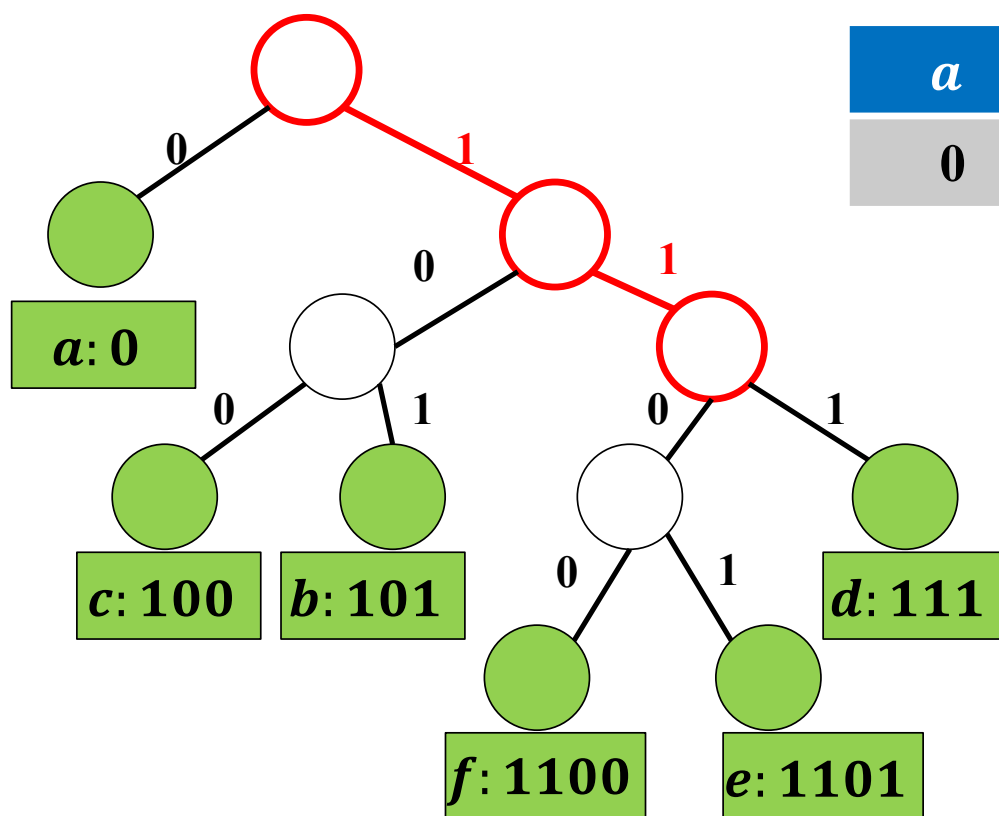
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

解析**1101**

问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



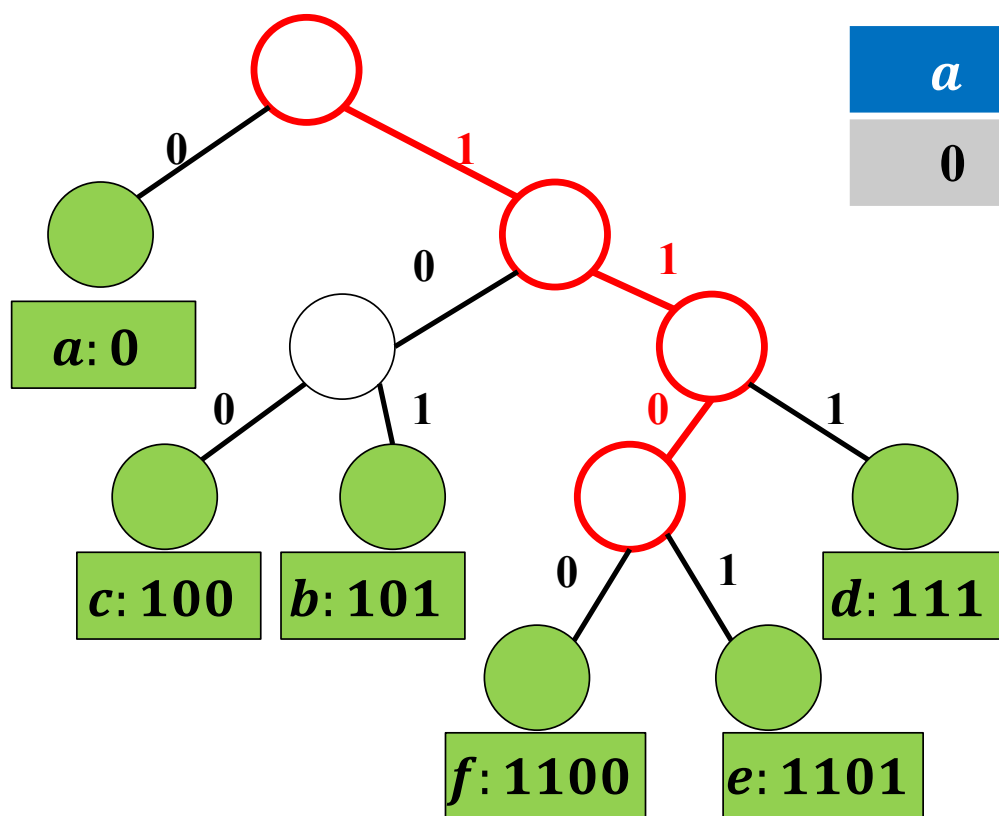
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

解析**1101**

问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



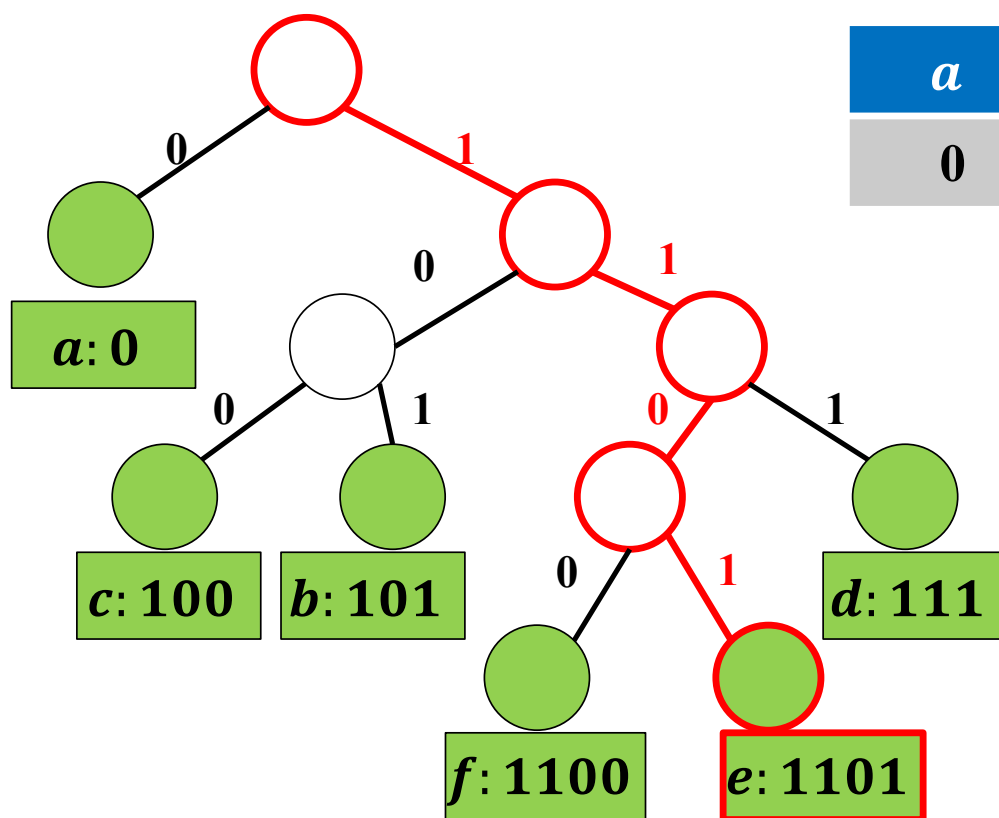
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

解析**1101**

问题背景：字符编码

• 编码树

- 顶点到左结点的边标记0，到右结点的边标记1，通过编码方案构造编码树
- 每条根到叶子的路径对应每个字符的二进制串



<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
0	101	100	111	1101	1100

1101解析结果为e



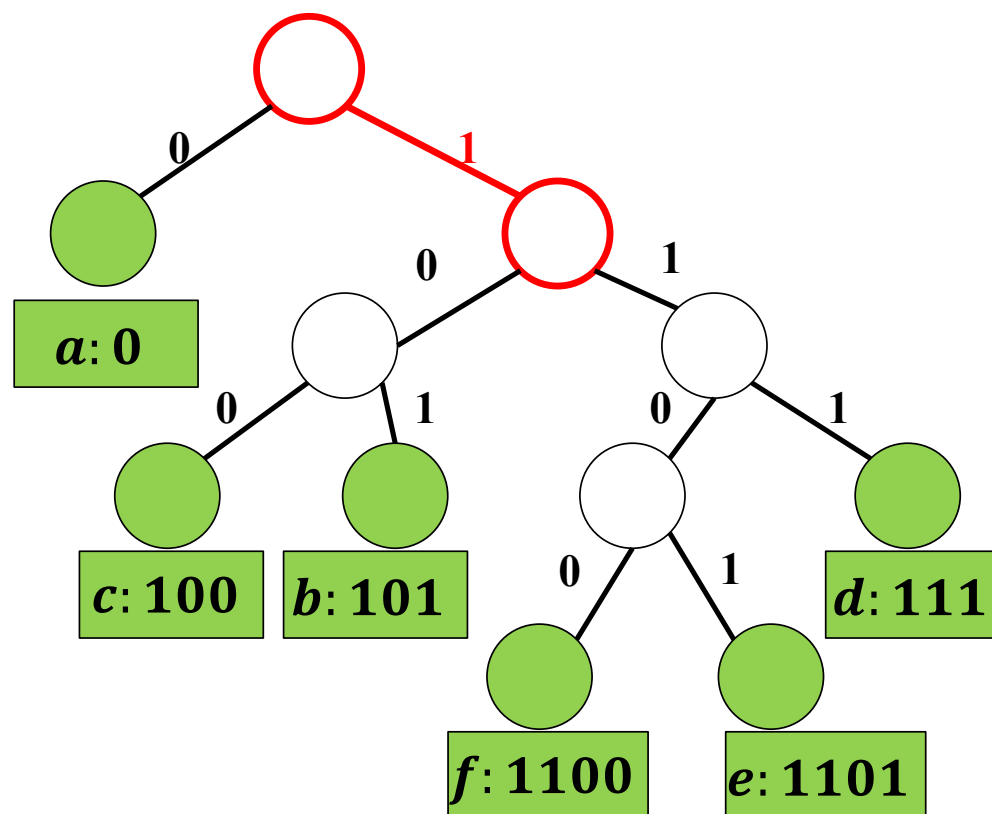
问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码

- 编码：*cfa* → 10011000

- 解码：10011000 →



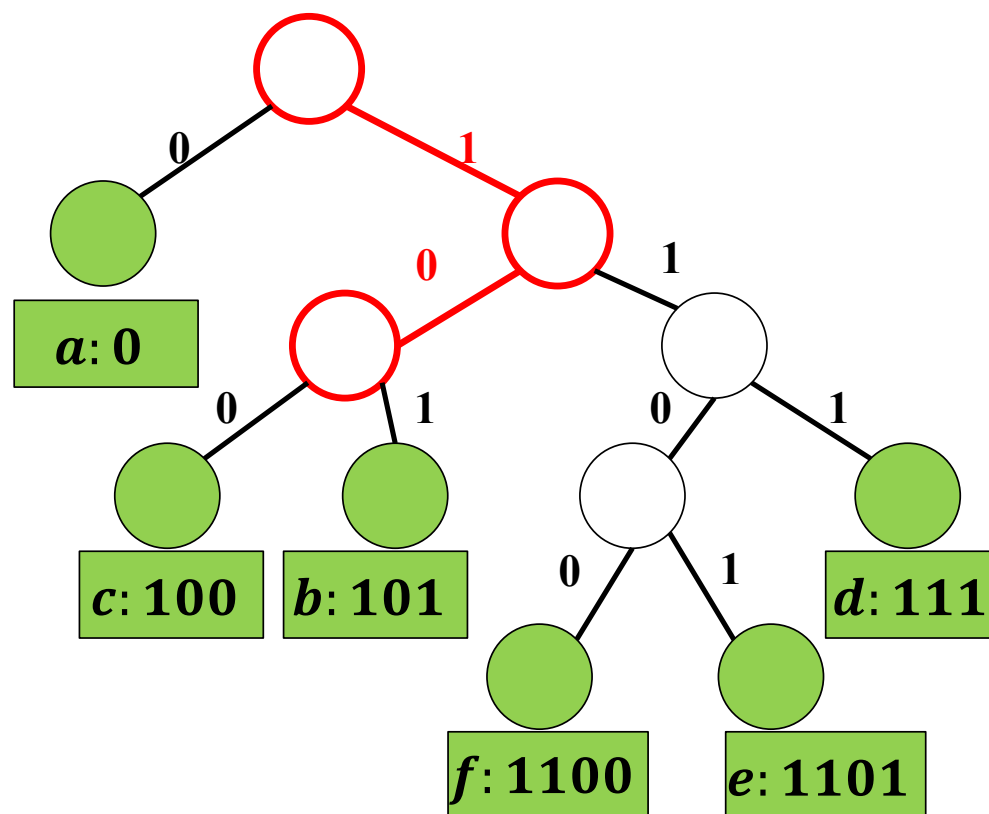
问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码

- 编码：*cfa* → 10011000

- 解码：**1**0011000 →

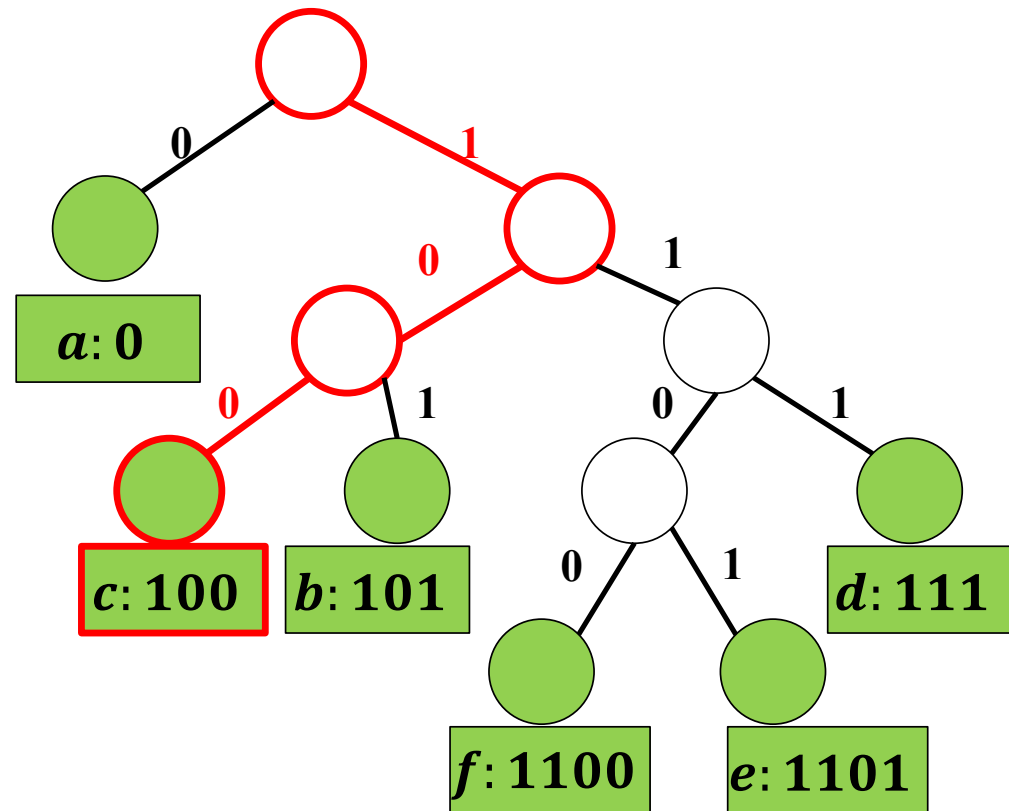


问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码

- 编码：*cfa* → 10011000
- 解码：**100**11000 → ***c***

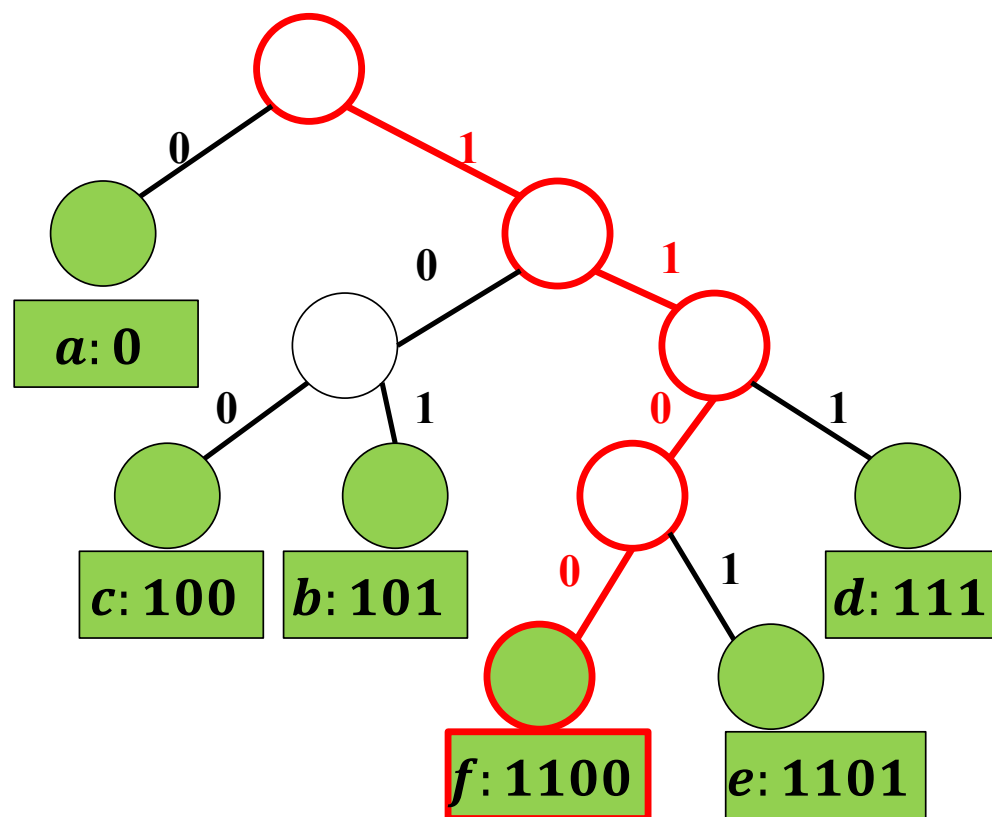


问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100

- 编码方式1：对"*cfa*"编码

- 编码：*cfa* → 10011000
- 解码：10011000 → *cf*





问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

- 编码方式1：对"*cfa*"编码
 - 编码： $cfa \rightarrow 10011000$
 - 解码： $10011000 \rightarrow cfa$
- 编码方式2：对"*cfa*"编码
 - 编码： $cfa \rightarrow 00$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

- 编码方式1：对"*cfa*"编码
 - 编码： $cfa \rightarrow 10011000$
 - 解码： $10011000 \rightarrow cfa$
- 编码方式2：对"*cfa*"编码
 - 编码： $cfa \rightarrow 0011$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

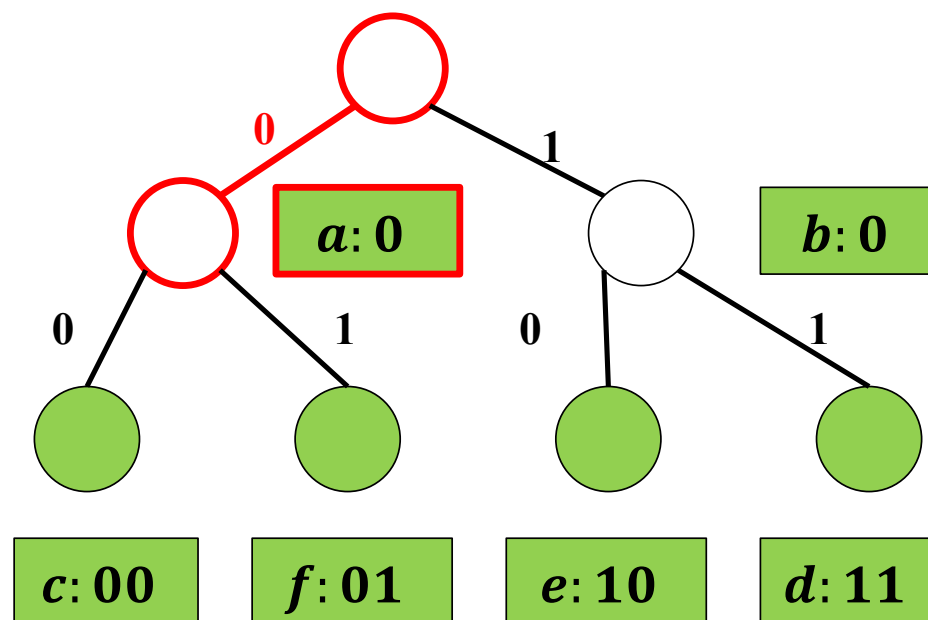
- 编码方式1：对"*cfa*"编码
 - 编码： $cfa \rightarrow 10011000$
 - 解码： $10011000 \rightarrow cfa$
- 编码方式2：对"*cfa*"编码
 - 编码： $cf\textcolor{red}{a} \rightarrow 0011\textcolor{red}{0}$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

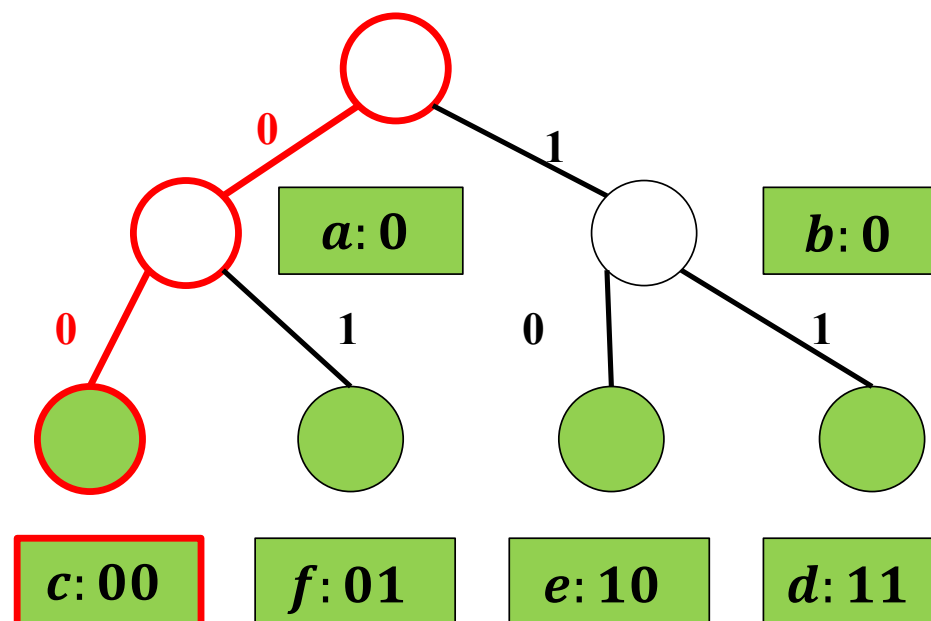
- 编码方式1：对"*cfa*"编码
 - 编码： $cfa \rightarrow 10011000$
 - 解码： $10011000 \rightarrow cfa$
- 编码方式2：对"*cfa*"编码
 - 编码： $cfa \rightarrow 00110$
 - 解码： $00110 \rightarrow a$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

- 编码方式1：对"*cfa*"编码
 - 编码：*cfa* → 10011000
 - 解码：10011000 → *cfa*
- 编码方式2：对"*cfa*"编码
 - 编码：*cfa* → 00110
 - 解码：00110 → *c*

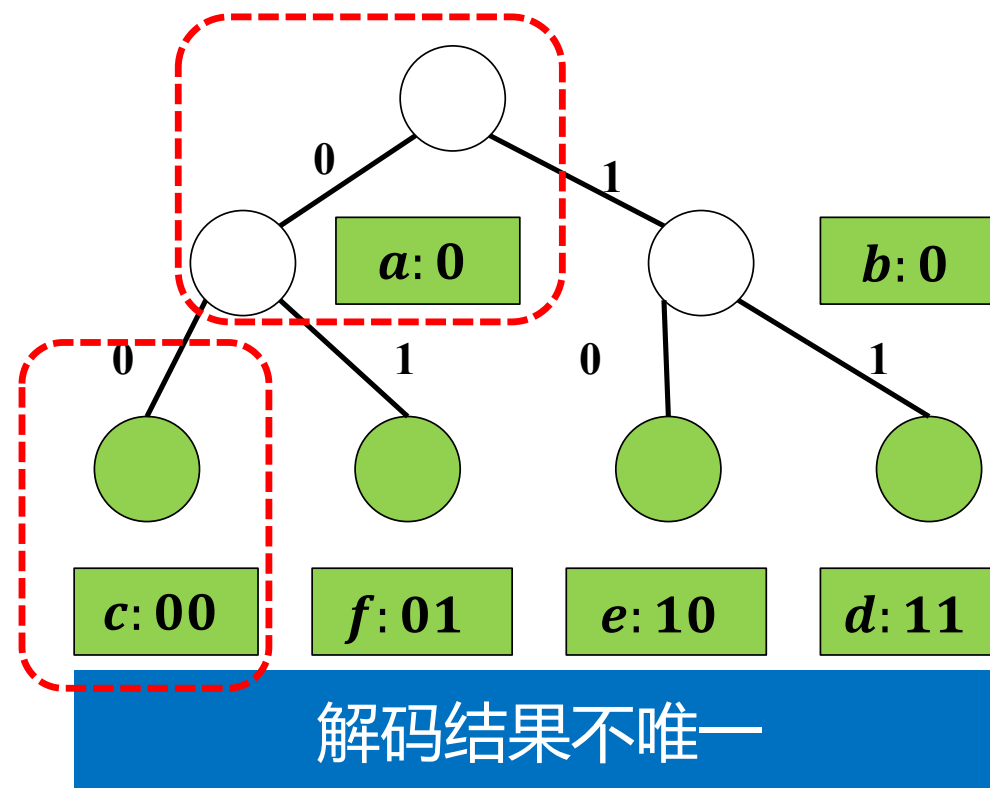




问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

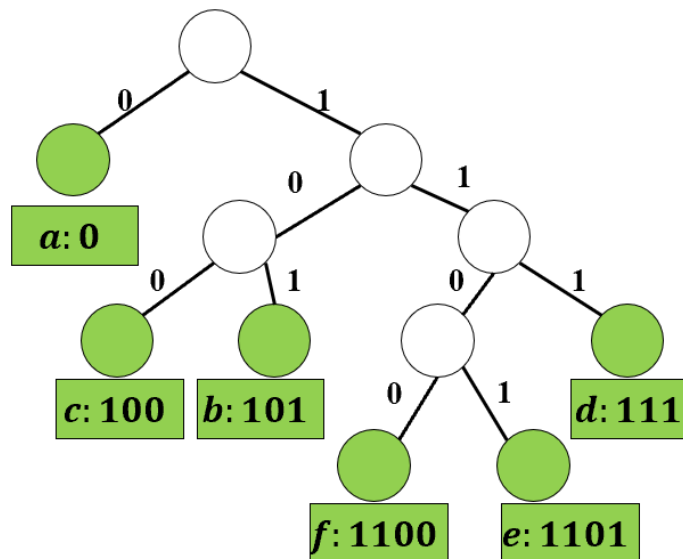
- 编码方式1：对"*cfa*"编码
 - 编码： $cfa \rightarrow 10011000$
 - 解码： $10011000 \rightarrow cfa$
- 编码方式2：对"*cfa*"编码
 - 编码： $cfa \rightarrow 00110$
 - 解码： $00110 \rightarrow aabba$
 - 解码： $00110 \rightarrow cfa$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

- 前缀码：编码的任意前缀不是其他编码



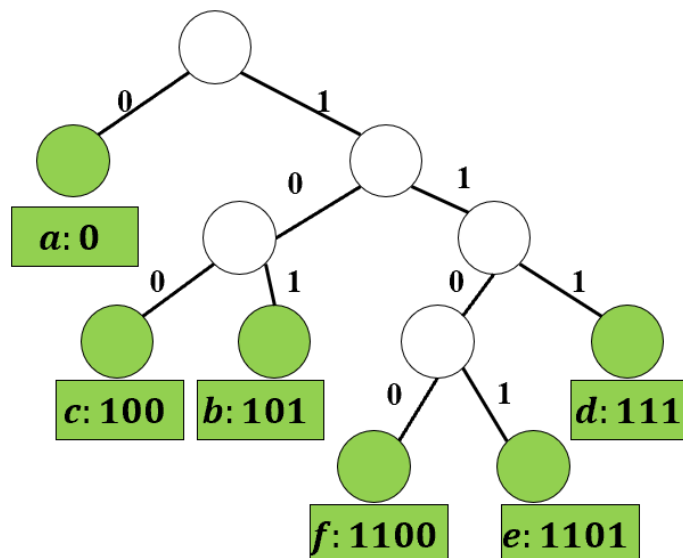
解码结果唯一，编码方式可行



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
编码方式1	0	101	100	111	1101	1100
编码方式2	0	1	00	01	10	11

- 前缀码：编码的任意前缀不是其他编码



问题：如何比较前缀码之间的优劣？



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5

- 比较编码后二进制串的总长，其依赖于待编码字符的频数



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5
前缀码1	0	101	100	111	1101	1100

- 比较编码后二进制串的总长，其依赖于待编码字符的频数
 - 使用前缀码1编码
 - $(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224\ 000$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5
前缀码1	0	101	100	111	1101	1100
前缀码2	1100	1101	111	100	101	0

- 比较编码后二进制串的总长，其依赖于待编码字符的频数
 - 使用前缀码1编码
 - $(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224\ 000$
 - 使用前缀码2编码
 - $(45 \times 4 + 13 \times 4 + 12 \times 3 + 16 \times 3 + 9 \times 3 + 5 \times 1) \times 1000 = 348\ 000$



问题背景：字符编码

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5
前缀码1	0	101	100	111	1101	1100
前缀码2	1100	1101	111	100	101	0

- 比较编码后二进制串的总长，其依赖于待编码字符的频数
 - 使用前缀码1编码
 - $(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224\ 000$
 - 使用前缀码2编码
 - $(45 \times 4 + 13 \times 4 + 12 \times 3 + 16 \times 3 + 9 \times 3 + 5 \times 1) \times 1000 = 348\ 000$

问题：如何求得编码后二进制串总长最短的前缀码？



问题定义

- 形式化定义

最优前缀码问题

Optimal Prefix Code Problem

输入

- 字符数 n 以及各个字符的频数 $F = \langle f_1, f_2, \dots, f_n \rangle$



问题定义

- 形式化定义

最优前缀码问题

Optimal Prefix Code Problem

输入

- 字符数 n 以及各个字符的频数 $F = \langle f_1, f_2, \dots, f_n \rangle$

输出

- 解析结果唯一的二进制编码方案 $C = \langle c_1, \dots, c_n \rangle$, 令



问题定义

- 形式化定义

最优前缀码问题

Optimal Prefix Code Problem

输入

- 字符数 n 以及各个字符的频数 $F = \langle f_1, f_2, \dots, f_n \rangle$

输出

- 解析结果唯一的二进制编码方案 $C = \langle c_1, \dots, c_n \rangle$, 令

$$\min \sum_{i=1}^n |c_i| \cdot f_i$$

$|c_i|$ 为字符 i 的编码二进制串长度



问题定义

- 形式化定义

最优前缀码问题

Optimal Prefix Code Problem

输入

- 字符数 n 以及各个字符的频数 $F = \langle f_1, f_2, \dots, f_n \rangle$

输出

- 解析结果唯一的二进制编码方案 $C = \langle c_1, \dots, c_n \rangle$, 令

$$\min \sum_{i=1}^n |c_i| \cdot f_i$$

优化目标

$|c_i|$ 为字符 i 的编码二进制串长度

贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择



证明策略正确

假设最优方案，通过替换证明



贪心策略：观察问题特征

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5
前缀码1	0	101	100	111	1101	1100
前缀码2	1100	1101	111	100	101	0

- 比较编码后二进制串的总长，其依赖于待编码字符的频数

最优解

- 使用前缀码1编码

$$(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224\ 000$$

- 使用前缀码2编码

$$(45 \times 4 + 13 \times 4 + 12 \times 3 + 16 \times 3 + 9 \times 3 + 5 \times 1) \times 1000 = 348\ 000$$



贪心策略：观察问题特征

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
频数 (千次)	45	13	12	16	9	5
前缀码1	0	101	100	111	1101	1100
前缀码2	1100	1101	111	100	101	0

- 比较编码后二进制串的总长，其依赖于待编码字符的频数

最优解

- 使用前缀码1编码

- $(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224\ 000$

- 使用前缀码2编码

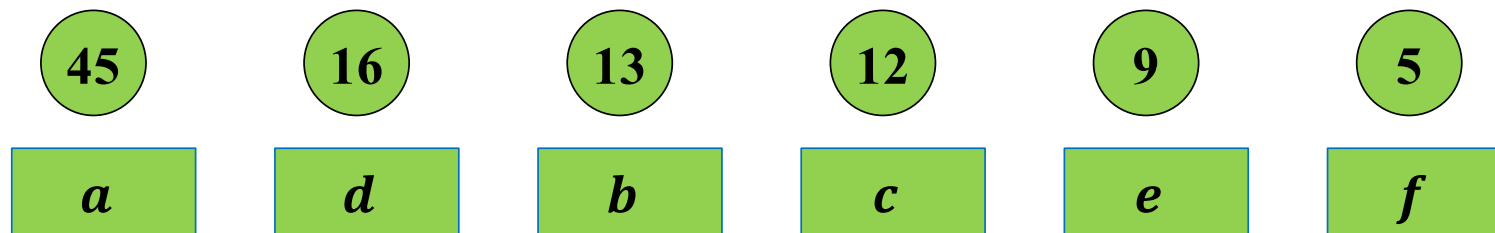
- $(45 \times 4 + 13 \times 4 + 12 \times 3 + 16 \times 3 + 9 \times 3 + 5 \times 1) \times 1000 = 348\ 000$

编码方案适应频数大小，短二进制串编码高频字符



贪心策略1

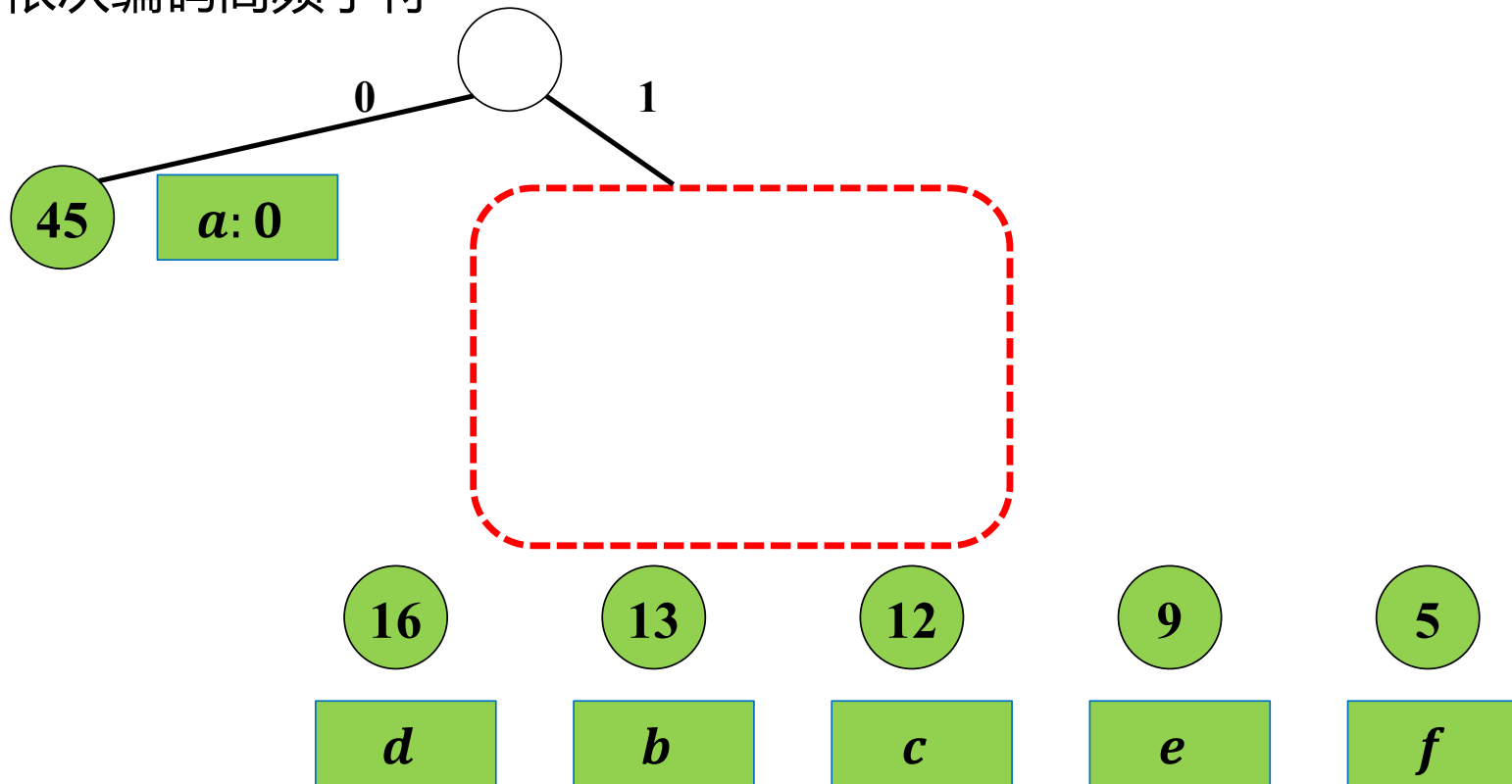
- 优先处理高频字符
 - 将字符频数从大到小排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \geq f_2 \geq \dots \geq f_n$)



贪心策略1

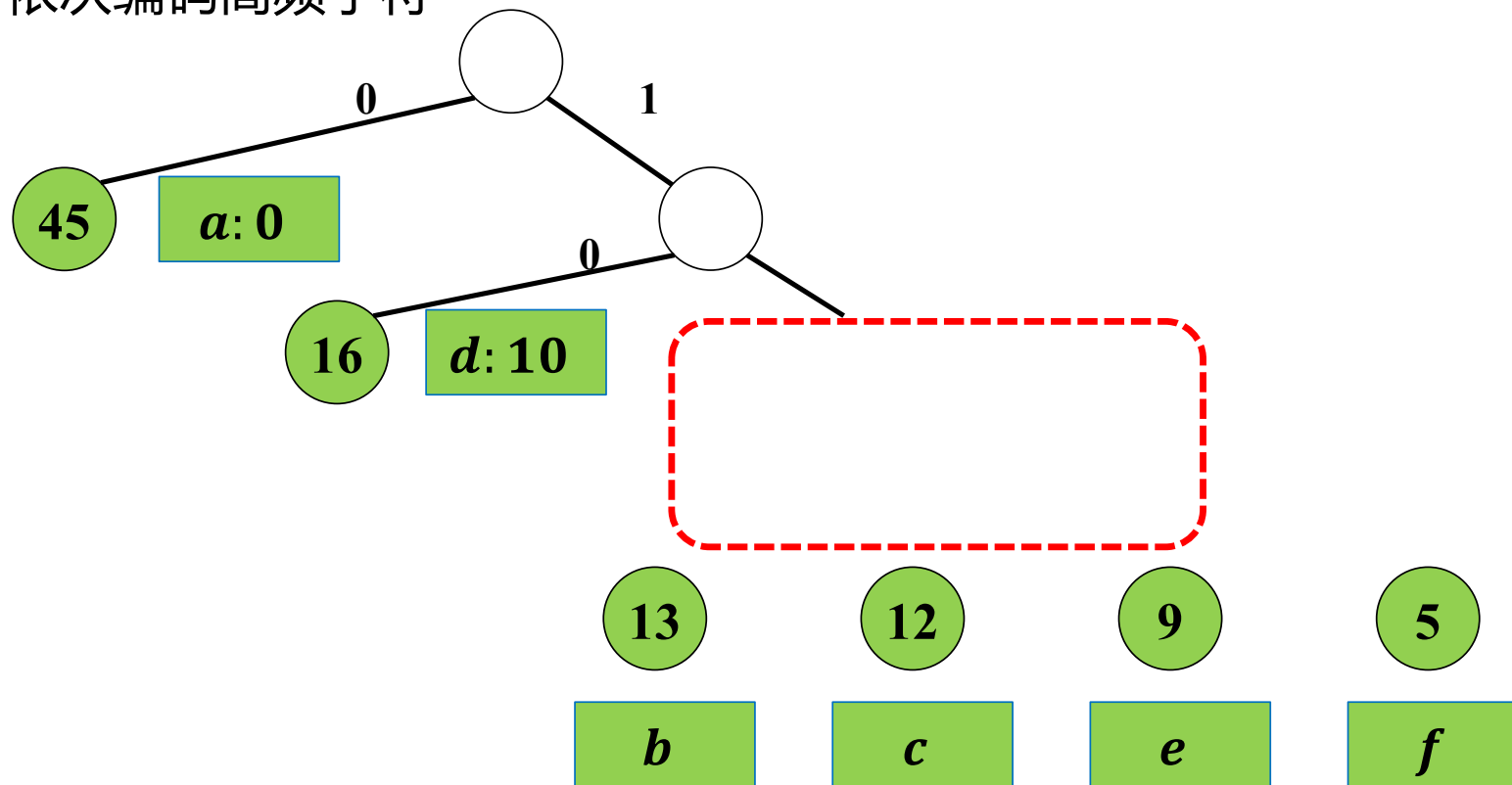
- 优先处理高频字符

- 将字符频数从大到小排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \geq f_2 \geq \dots \geq f_n$)
- 依次编码高频字符



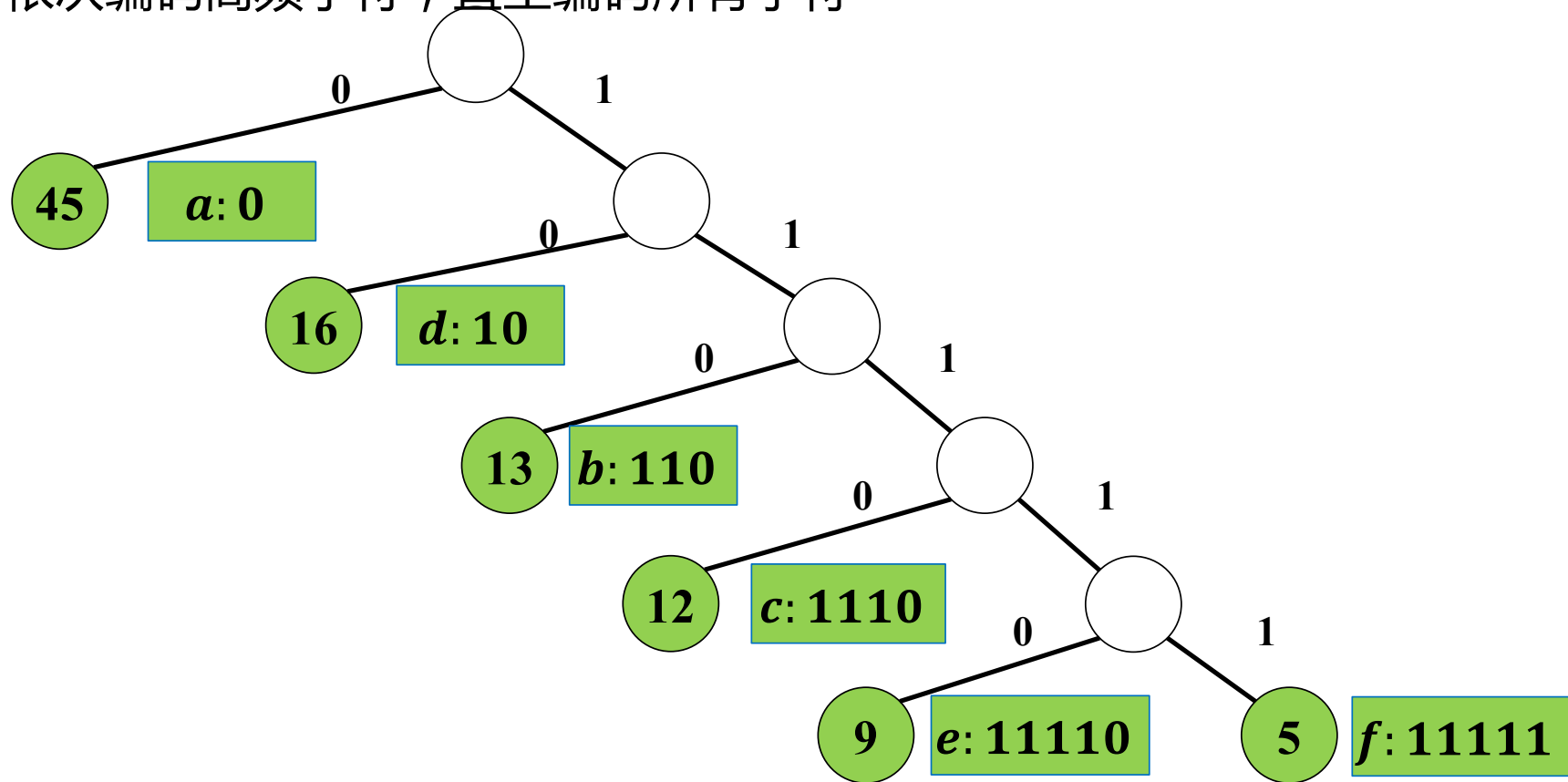
贪心策略1

- 优先处理高频字符
 - 将字符频数从大到小排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \geq f_2 \geq \dots \geq f_n$)
 - 依次编码高频字符



贪心策略1

- 优先处理高频字符
 - 将字符频数从大到小排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \geq f_2 \geq \dots \geq f_n$)
 - 依次编码高频字符，直至编码所有字符

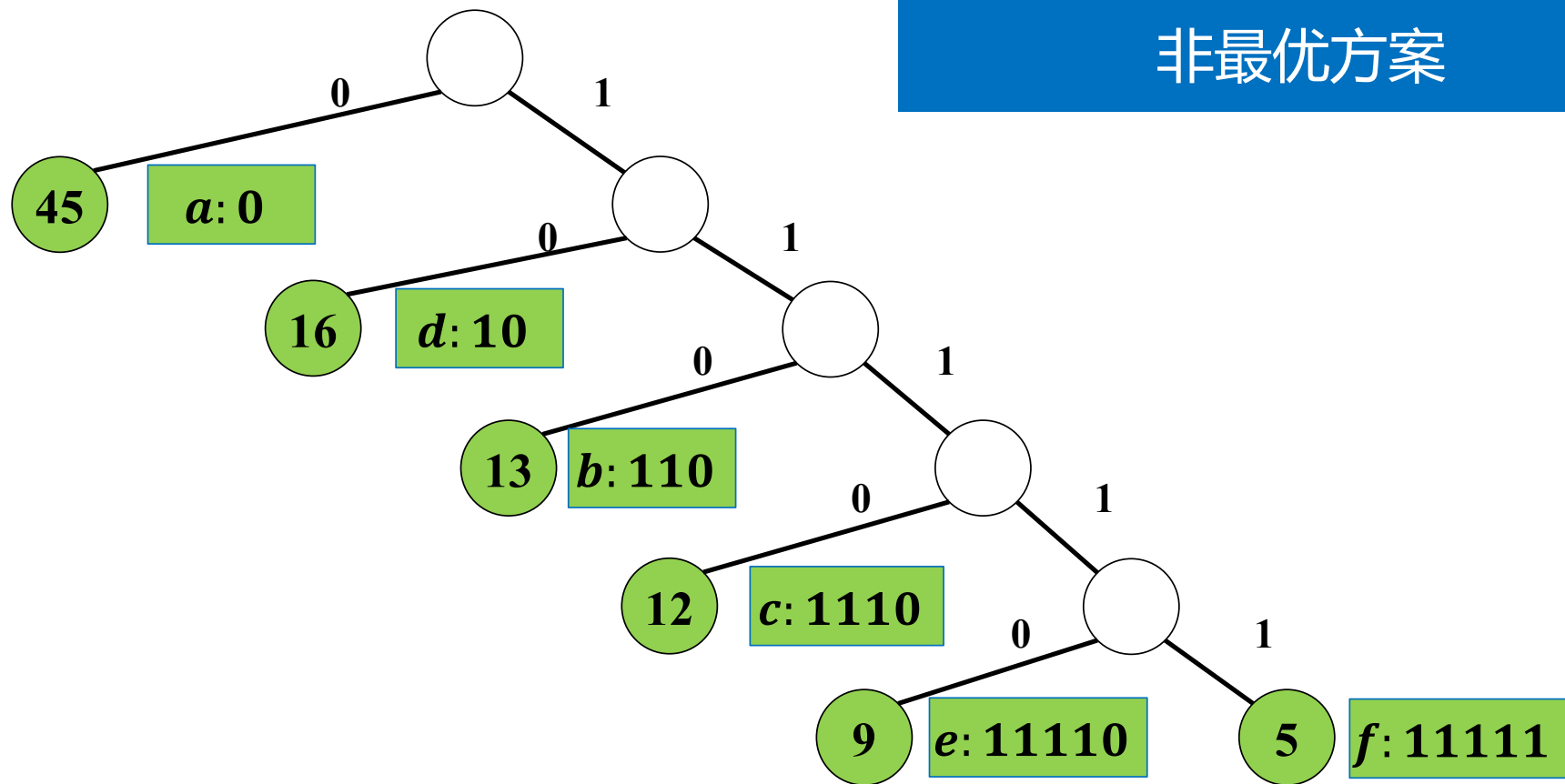




贪心策略1

- 优先处理高频字符

- $45 \times 1 + 16 \times 2 + 13 \times 3 + 12 \times 4 + 9 \times 5 + 5 \times 5 = 234 > 224$

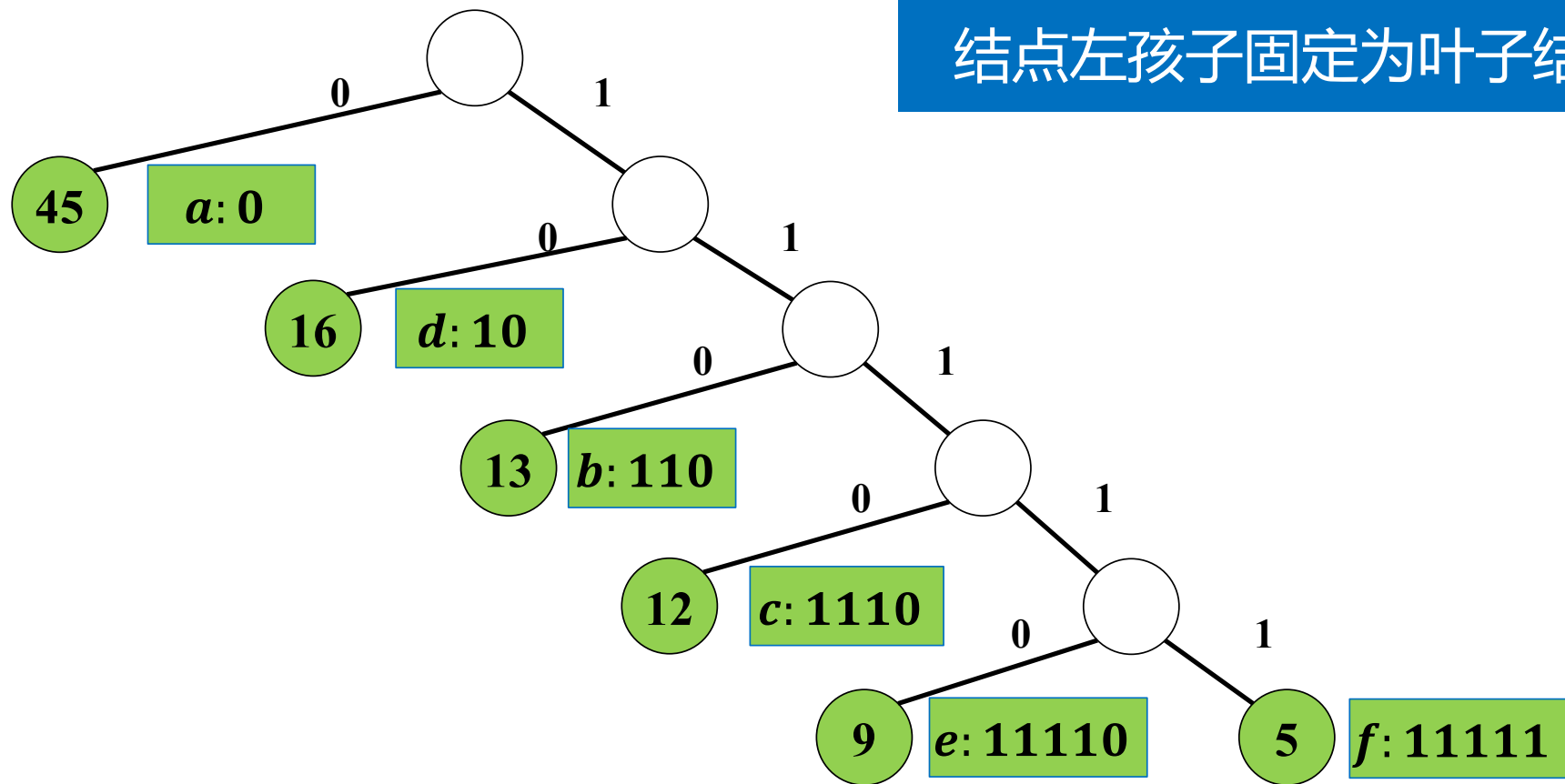




贪心策略1

- 优先处理高频字符

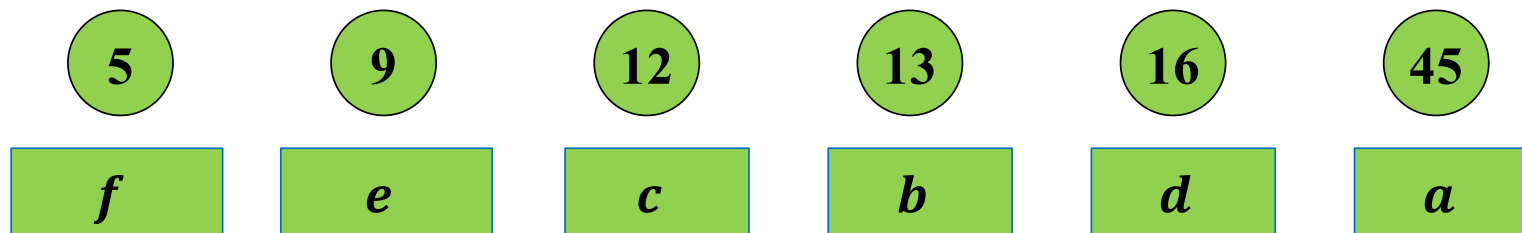
- $45 \times 1 + 16 \times 2 + 13 \times 3 + 12 \times 4 + 9 \times 5 + 5 \times 5 = 234 > 224$





贪心策略2

- 优先处理低频字符
 - 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)



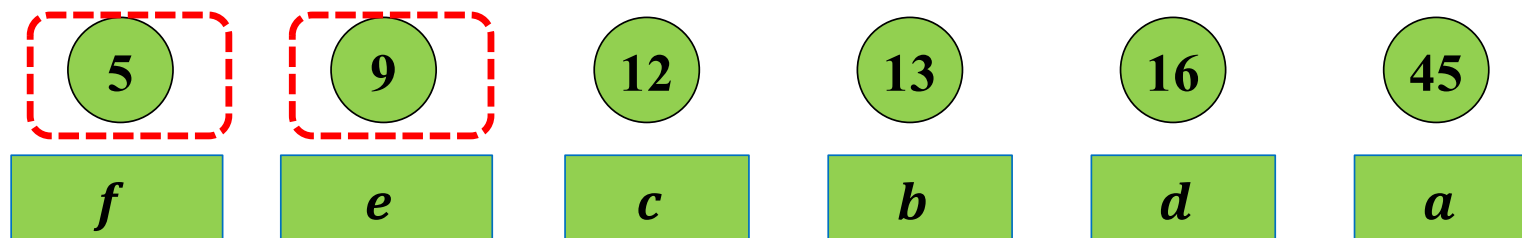


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$

叶子频数	5	9	12	13	16	45
合并频数						



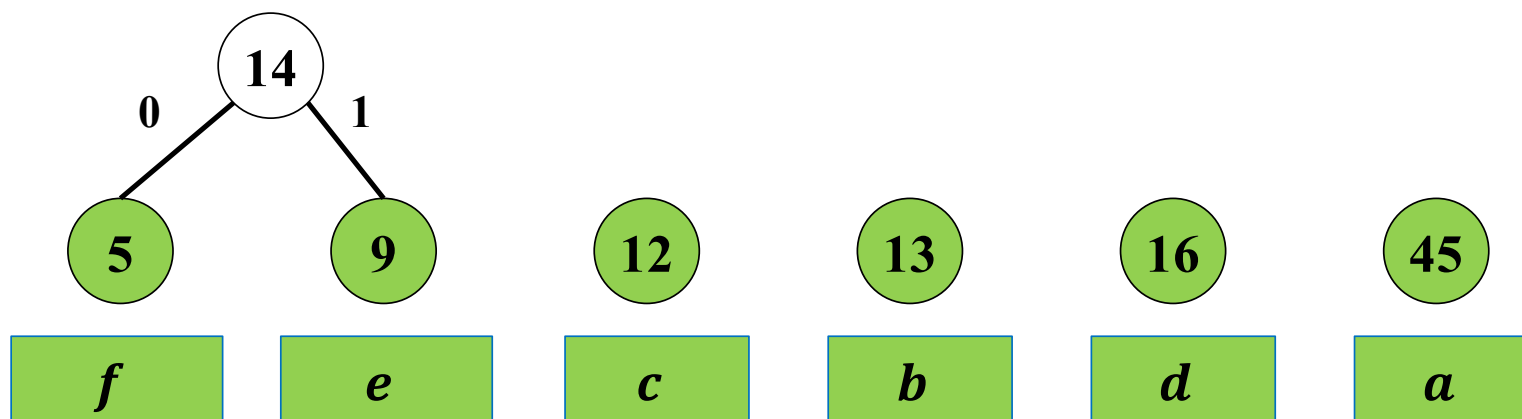


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$

叶子频数			12	13	16	45
合并频数	14					



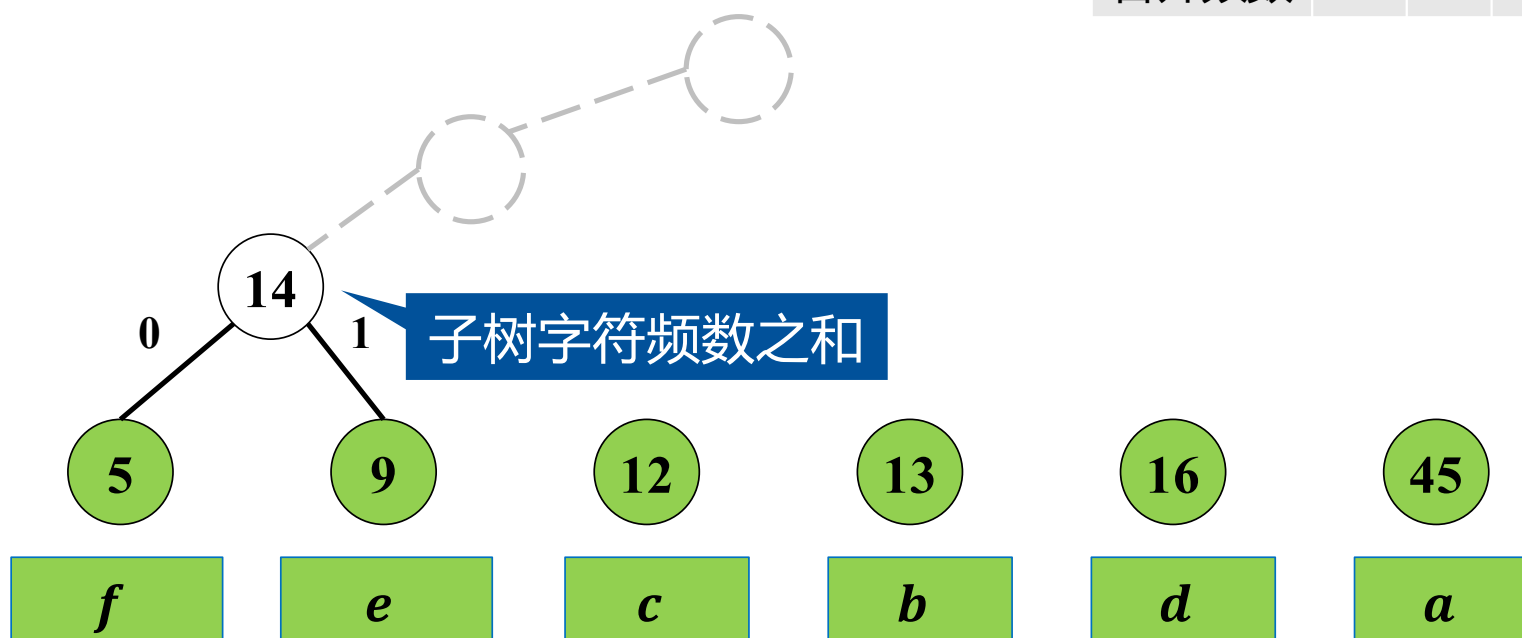


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$

叶子频数			12	13	16	45
合并频数	14					



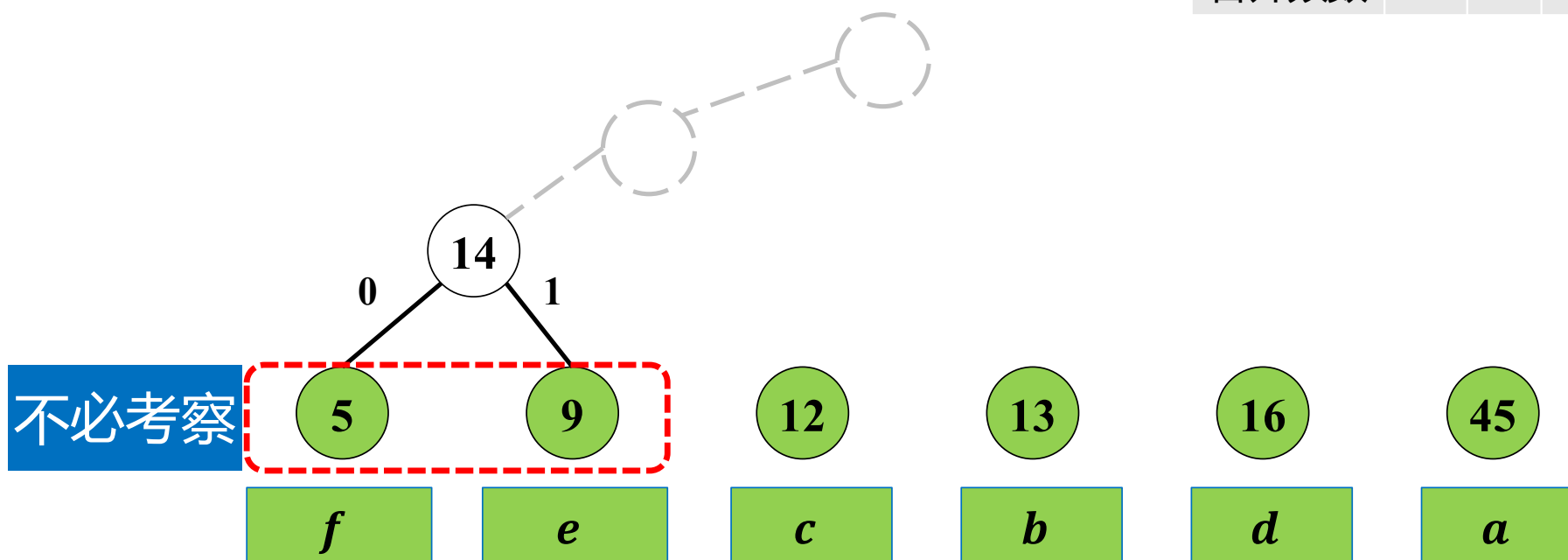


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$

叶子频数			12	13	16	45
合并频数	14					



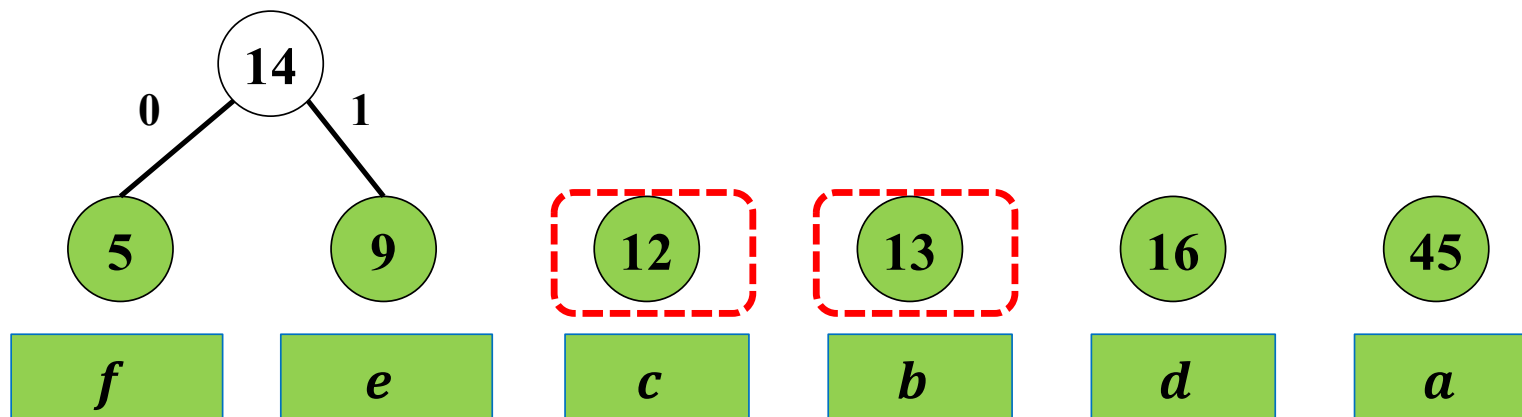


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数			12	13	16	45
合并频数	14					

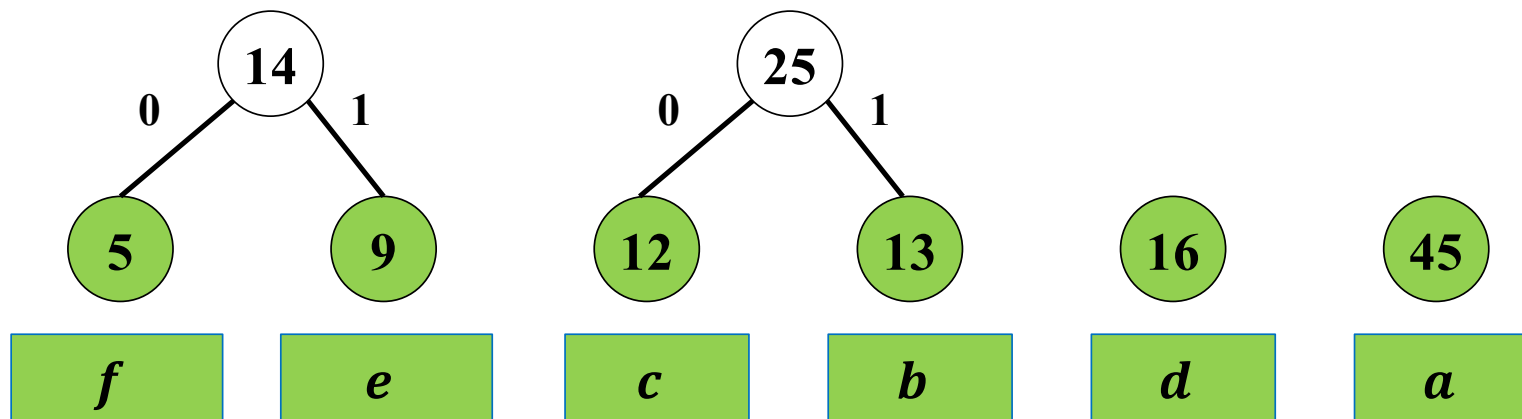


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数					16	45
合并频数	14	25				

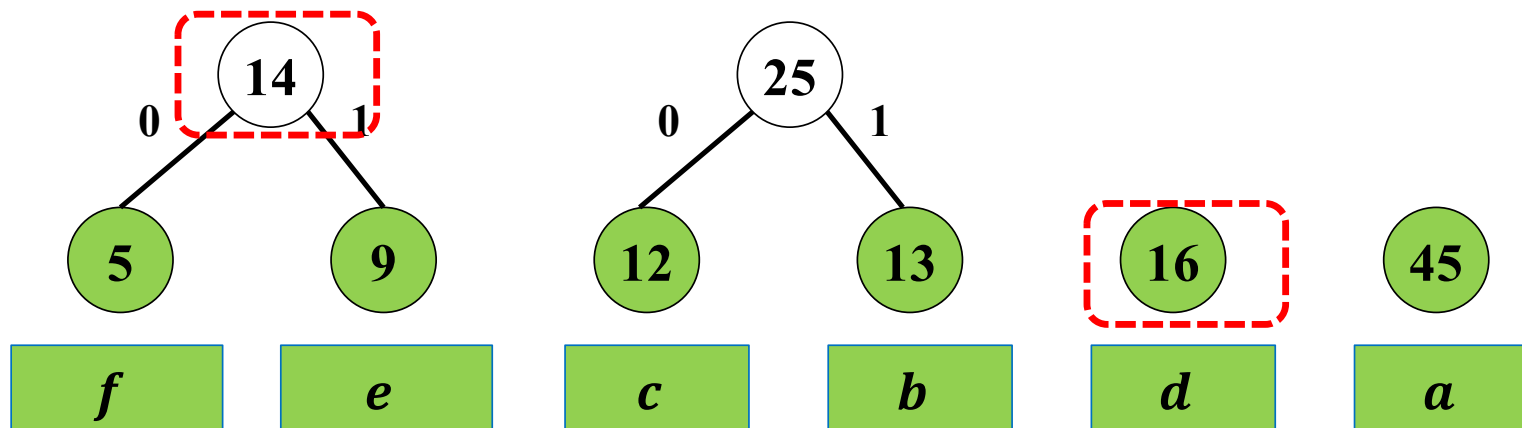


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数					16	45
合并频数	14	25				



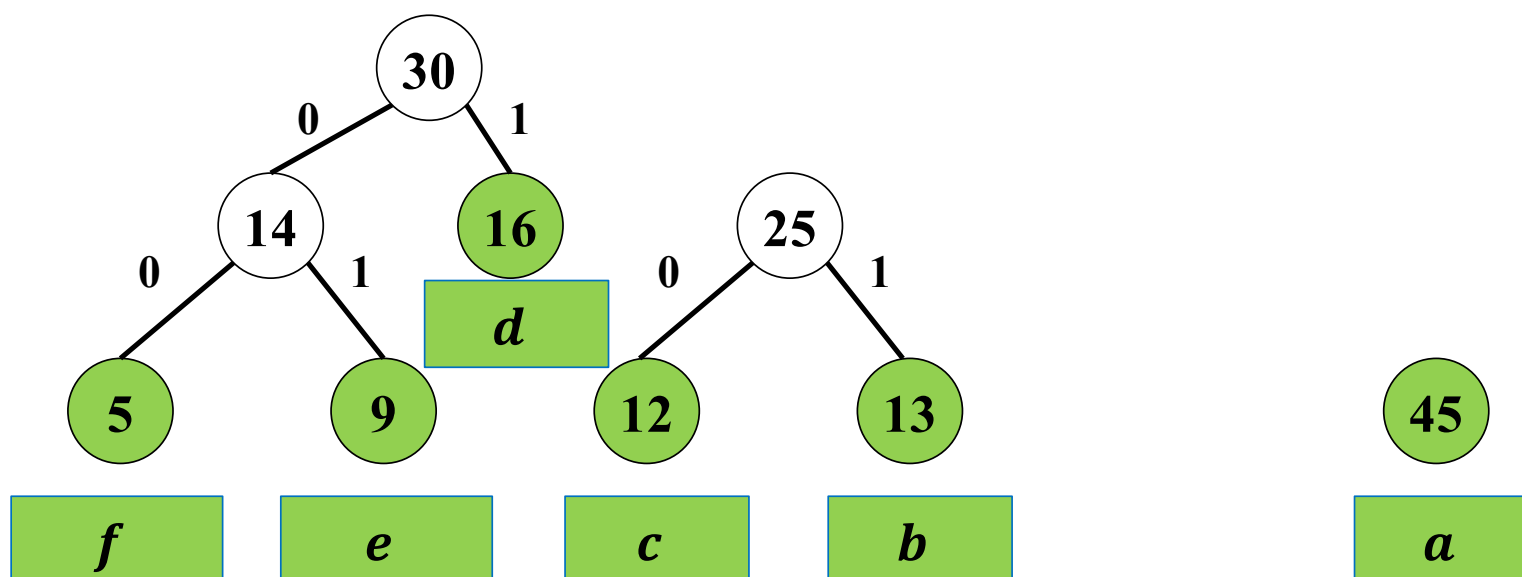


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						45
合并频数		25	30			



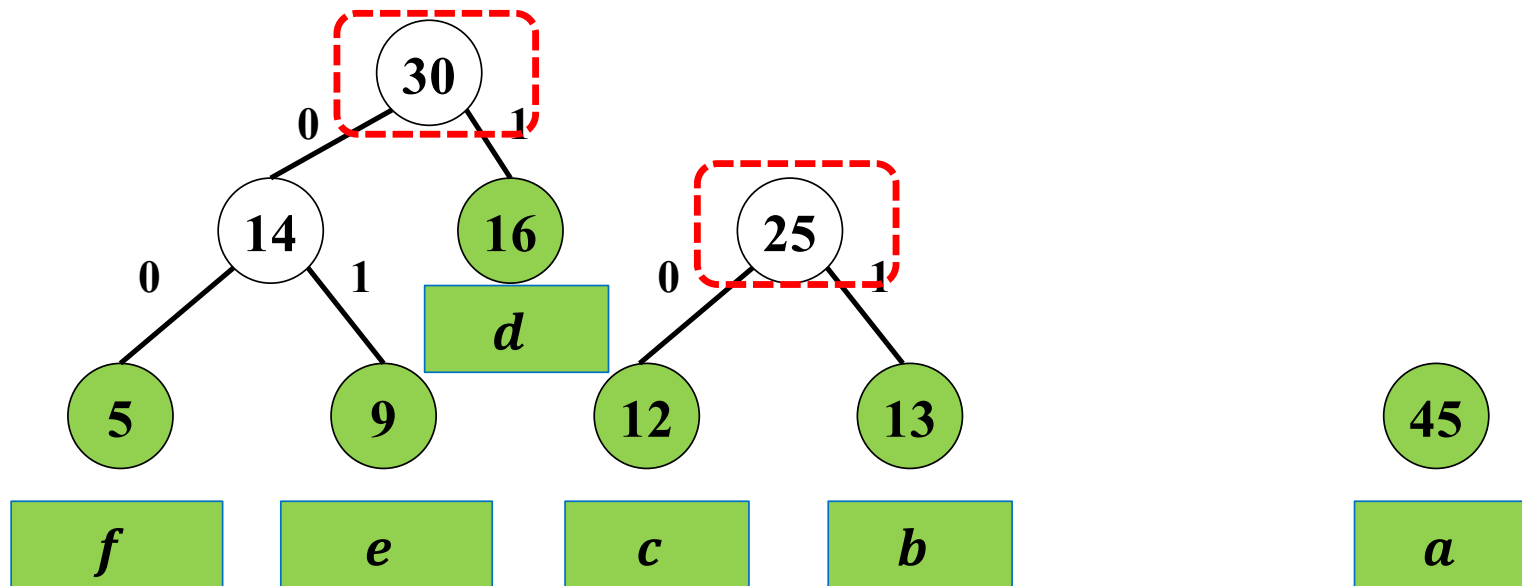


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						45
合并频数		25	30			

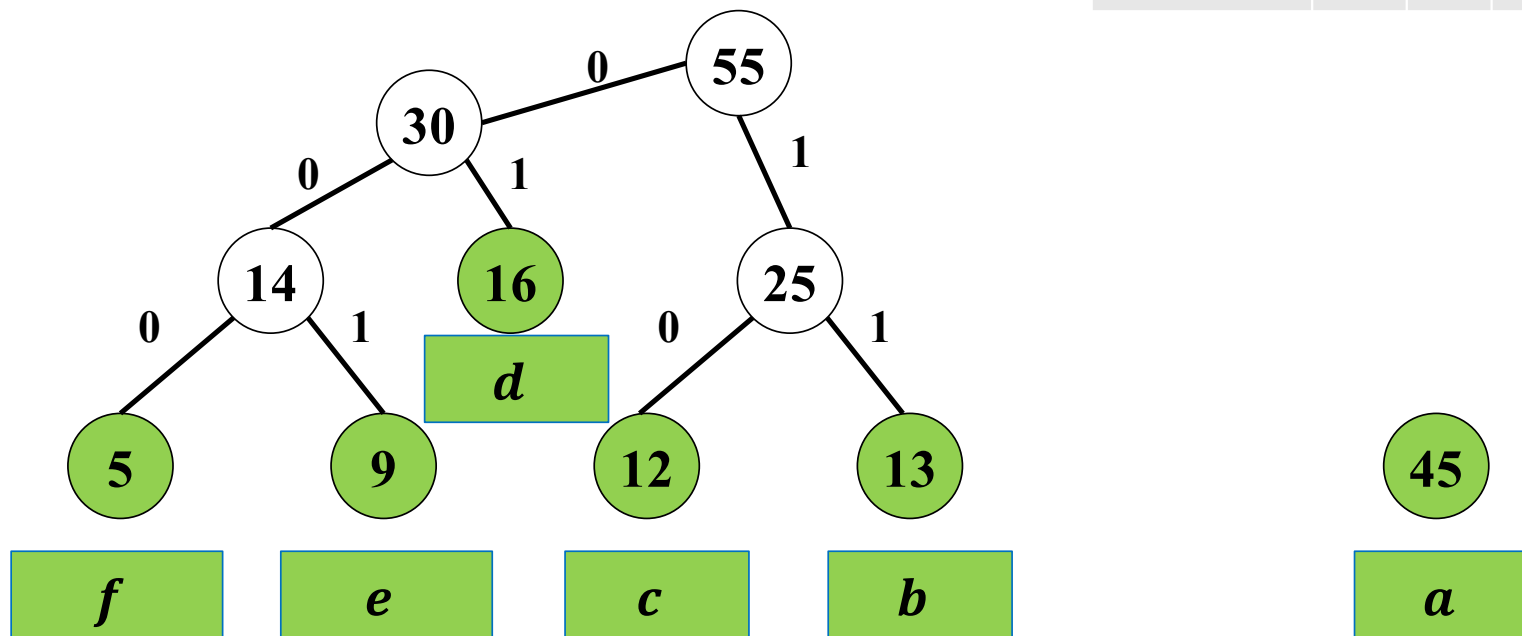


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						45
合并频数				55		

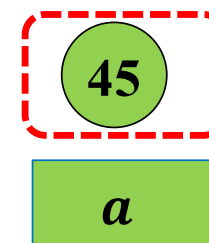
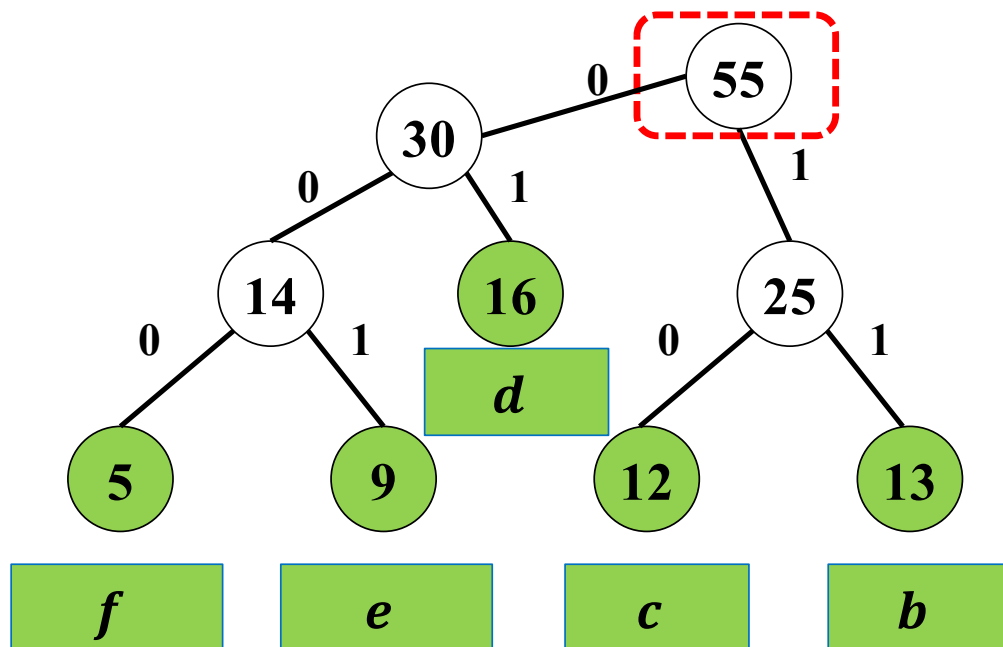


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						45
合并频数				55		

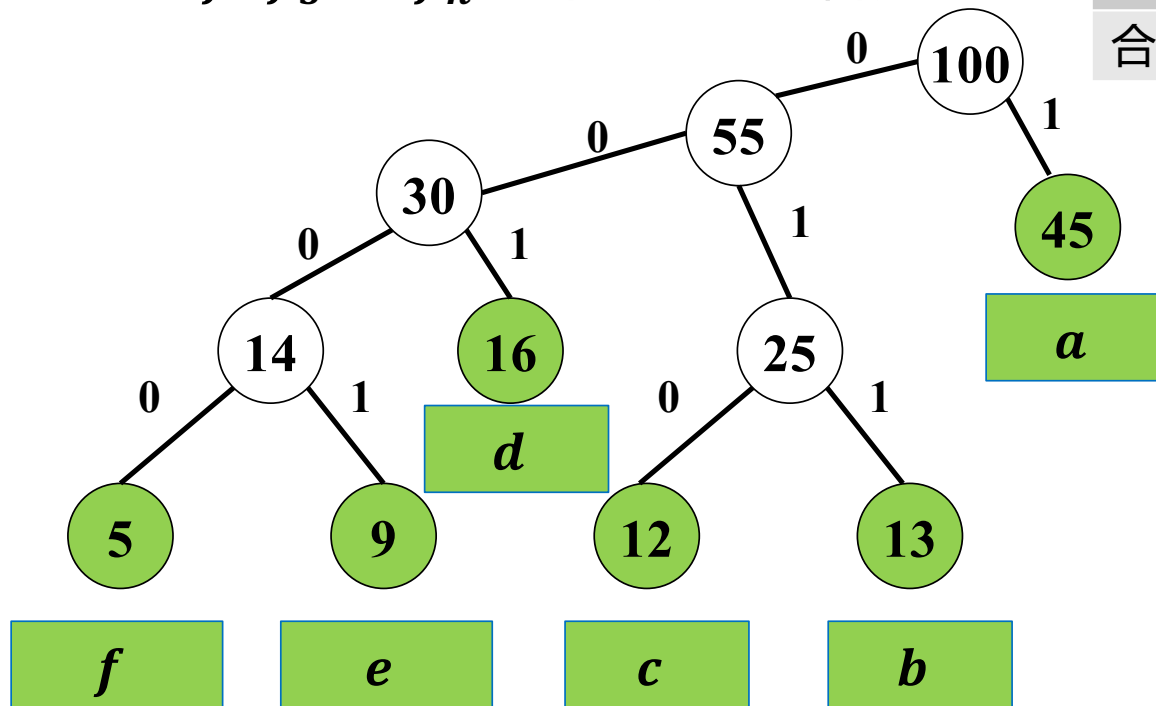


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						
合并频数					100	

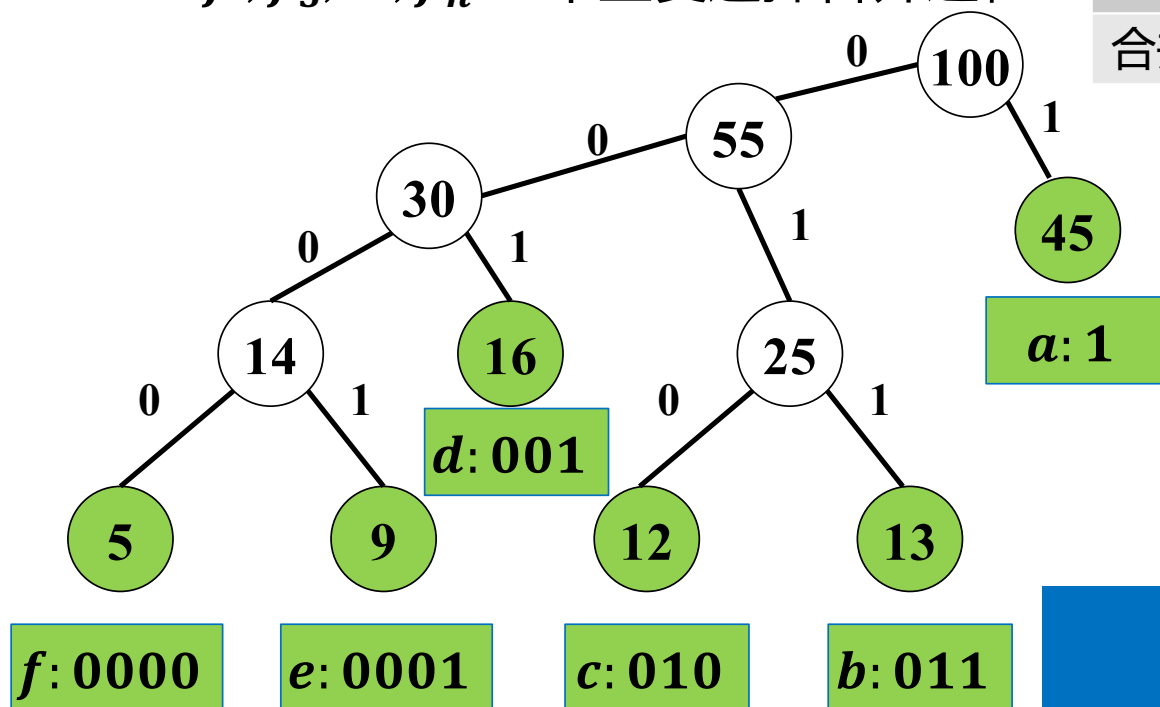


贪心策略2

- 优先处理低频字符

- 将字符频数从小到大排序 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 选择两个最小的频数 f_1, f_2 , 合并为 $f' = f_1 + f_2$
- 在 $F' = \langle f', f_3, \dots, f_n \rangle$ 中重复选择合并过程

叶子频数						
合并频数					100	



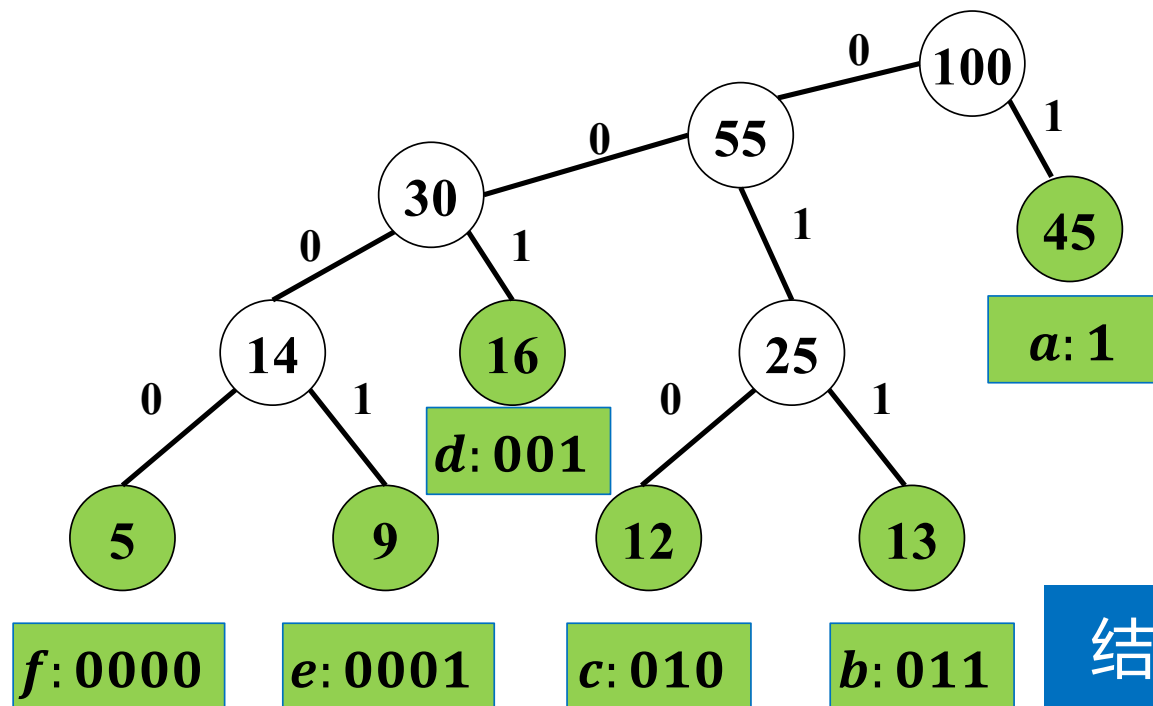
霍夫曼编码



贪心策略2

- 优先处理低频字符

- $45 + 55 + 25 + 30 + 12 + 13 + 14 + 16 + 9 + 5 = 224$ (千次)



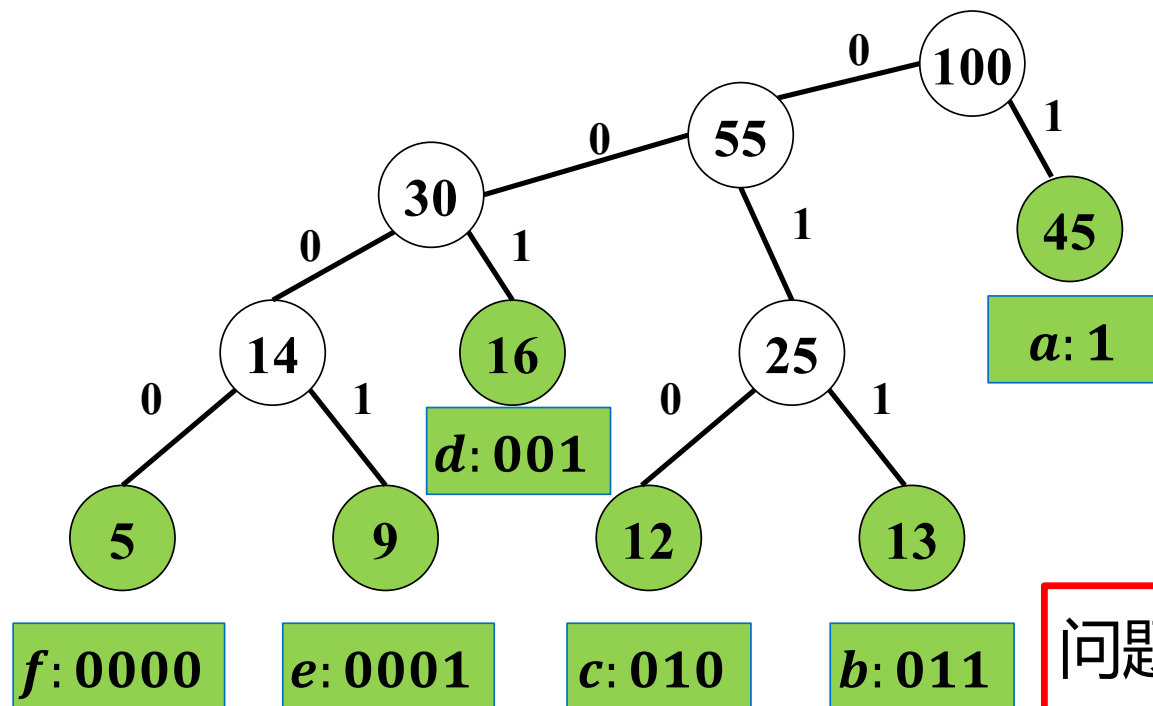
结点的孩子可为子树或叶子



贪心策略2

- 优先处理低频字符

- $45 + 55 + 25 + 30 + 12 + 13 + 14 + 16 + 9 + 5 = 224(\text{千次})$



问题：是否为最优编码方案？

贪心策略：一般步骤



提出贪心策略

观察问题特征，构造贪心选择

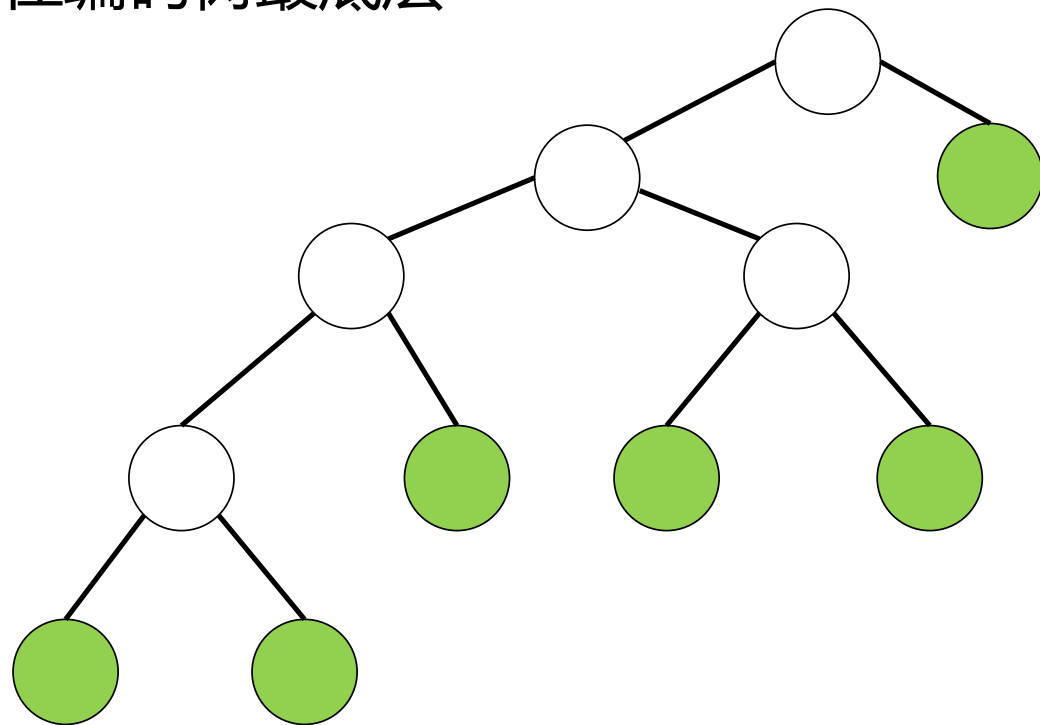


证明策略正确

假设最优方案，通过替换证明

正确性证明

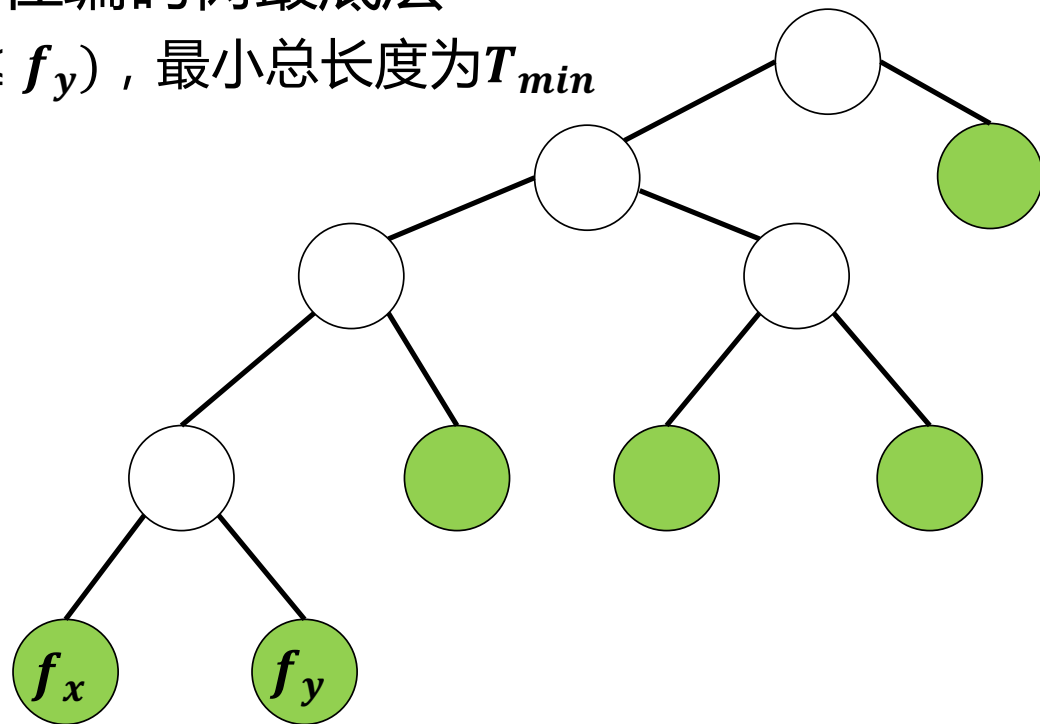
- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层





正确性证明

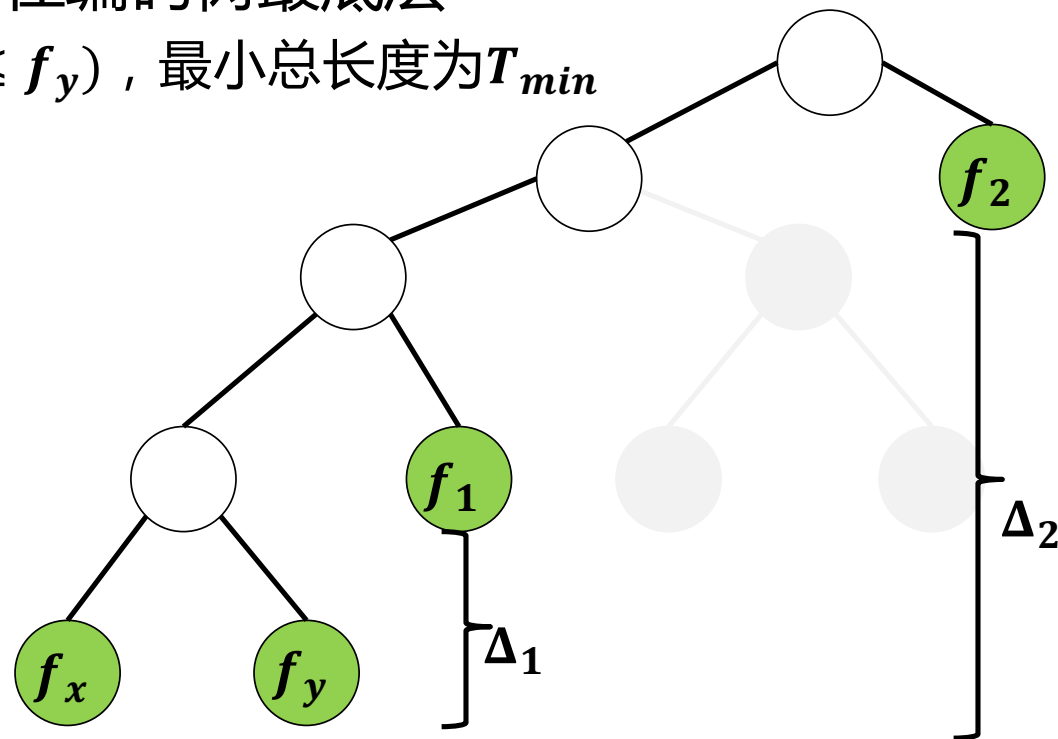
- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层
 - 最优方案最底层两结点是 f_x, f_y ($f_x \leq f_y$)，最小总长度为 T_{min}



最优编码树 T_{min}

正确性证明

- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层
 - 最优方案最底层两结点是 f_x, f_y ($f_x \leq f_y$)，最小总长度为 T_{min}
 - 根据原始条件有
 - $f_1 \leq f_x, f_2 \leq f_y$
 - $\Delta_1 = d_T(f_x) - d_T(f_1) \geq 0$
 - $\Delta_2 = d_T(f_y) - d_T(f_2) \geq 0$



最优编码树 T_{min}

正确性证明

- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层

- 最优方案最底层两结点是 f_x, f_y ($f_x \leq f_y$)，最小总长度为 T_{min}

- 根据原始条件有

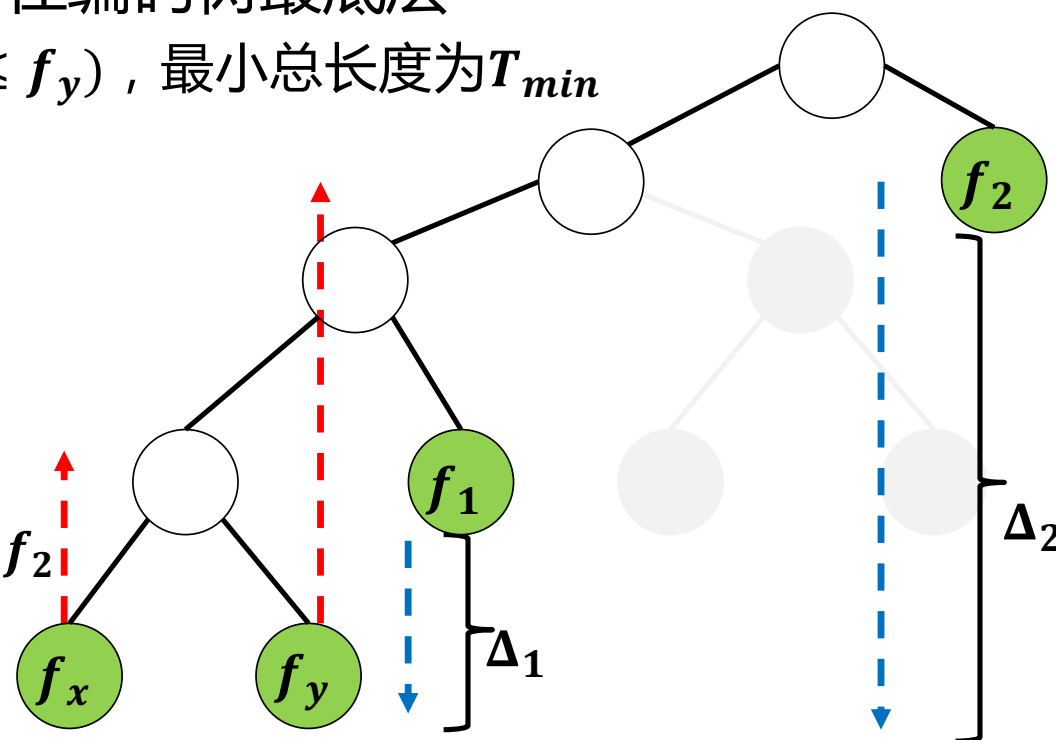
- $f_1 \leq f_x, f_2 \leq f_y$

- $\Delta_1 = d_T(f_x) - d_T(f_1) \geq 0$

- $\Delta_2 = d_T(f_y) - d_T(f_2) \geq 0$

- 交换 f_1 与 f_x ， f_2 与 f_y 可得

- $T = T_{min} - \Delta_1 f_x - \Delta_2 f_y + \Delta_1 f_1 + \Delta_2 f_2$



最优编码树 T_{min}

正确性证明

- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层

- 最优方案最底层两结点是 f_x, f_y ($f_x \leq f_y$)，最小总长度为 T_{min}

- 根据原始条件有

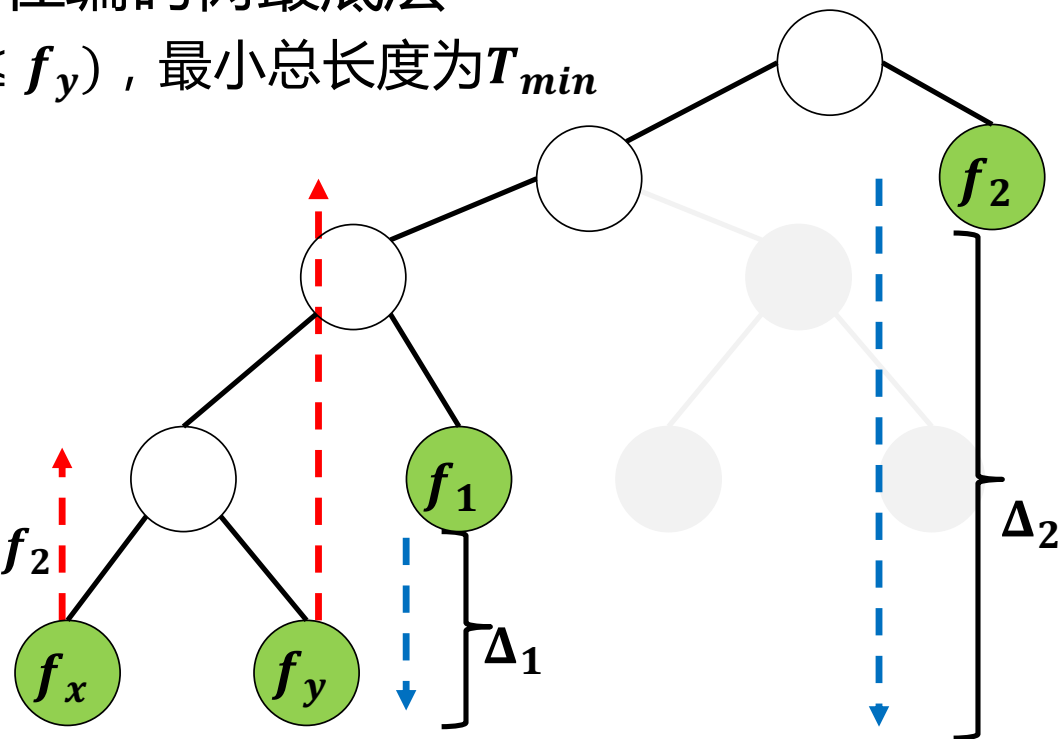
- $f_1 \leq f_x, f_2 \leq f_y$

- $\Delta_1 = d_T(f_x) - d_T(f_1) \geq 0$

- $\Delta_2 = d_T(f_y) - d_T(f_2) \geq 0$

- 交换 f_1 与 f_x ， f_2 与 f_y 可得

- $$\begin{aligned} T &= T_{min} - \Delta_1 f_x - \Delta_2 f_y + \Delta_1 f_1 + \Delta_2 f_2 \\ &= T_{min} - \Delta_1 (f_x - f_1) - \Delta_2 (f_y - f_2) \\ &\leq T_{min} \end{aligned}$$



最优编码树 T_{min}

正确性证明

- 从小到大排序后，频数 $F = \langle f_1, f_2, \dots, f_n \rangle$ ($f_1 \leq f_2 \leq \dots \leq f_n$)
- 证明：存在最优方案，使得 f_1, f_2 在编码树最底层

- 最优方案最底层两结点是 f_x, f_y ($f_x \leq f_y$)，最小总长度为 T_{min}

- 根据原始条件有

- $f_1 \leq f_x, f_2 \leq f_y$

- $\Delta_1 = d_T(f_x) - d_T(f_1) \geq 0$

- $\Delta_2 = d_T(f_y) - d_T(f_2) \geq 0$

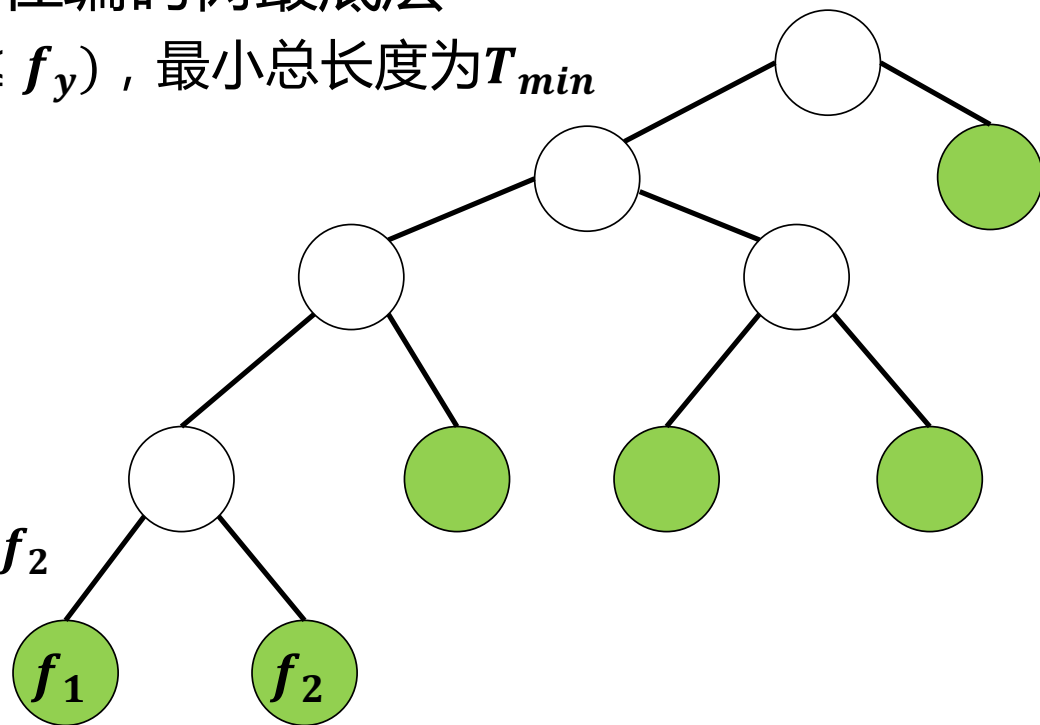
- 交换 f_1 与 f_x ， f_2 与 f_y 可得

- $T = T_{min} - \Delta_1 f_x - \Delta_2 f_y + \Delta_1 f_1 + \Delta_2 f_2$

- $= T_{min} - \Delta_1 (f_x - f_1) - \Delta_2 (f_y - f_2)$

- $\leq T_{min}$

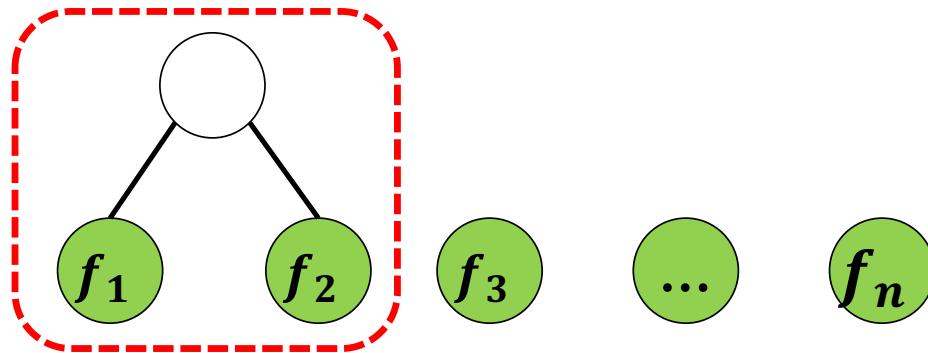
- 存在最优方案，使得 f_1, f_2 在编码树最底层



最优编码树 T_{min}

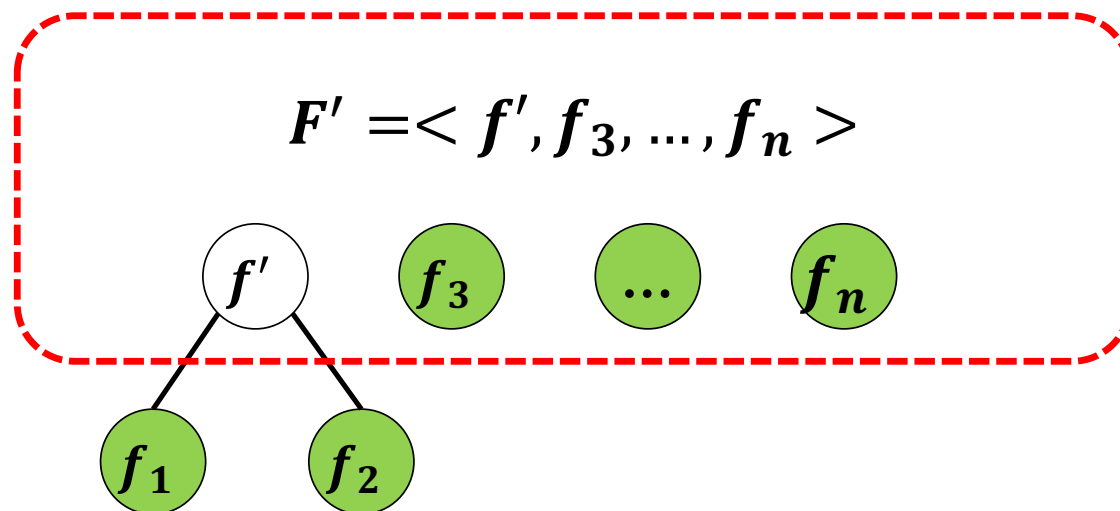
正确性证明

- $F = \langle f_1, f_2, \dots, f_n \rangle$ 存在最优方案, 使 f_1, f_2 在编码树最底层



正确性证明

- $F = \langle f_1, f_2, \dots, f_n \rangle$ 存在最优方案, 使 f_1, f_2 在编码树最底层
- $F' = \langle f', f_3, \dots, f_n \rangle$ ($f' = f_1 + f_2$) 中, 再寻最优方案

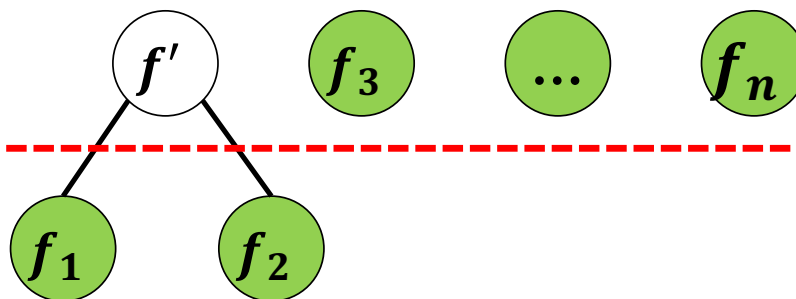


最优子结构性质

正确性证明

- $F = \langle f_1, f_2, \dots, f_n \rangle$ 存在最优方案，使 f_1, f_2 在编码树最底层
- $F' = \langle f', f_3, \dots, f_n \rangle$ ($f' = f_1 + f_2$) 中，再寻最优方案

$F' = \langle f', f_3, \dots, f_n \rangle$ 中选择最小两频数



此过程即为霍夫曼编码，贪心正确性得证

伪代码



• Huffman(F, n)

输入: 字符数 n , 各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

字符频数按从小到大排序

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

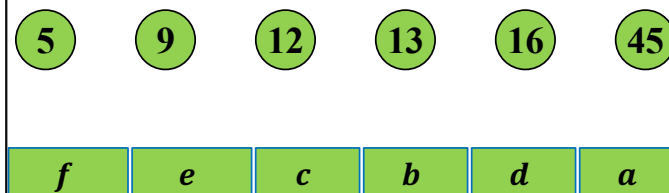
$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$





伪代码

• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n-1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

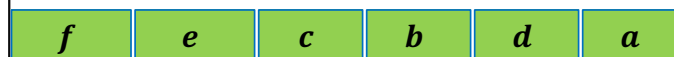
$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

初始化结点数组

叶子频数	5	9	12	13	16	45
合并频数						





伪代码

● Huffman(F, n)

输入: 字符数 n , 各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

共需合并 $n - 1$ 次

伪代码



• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

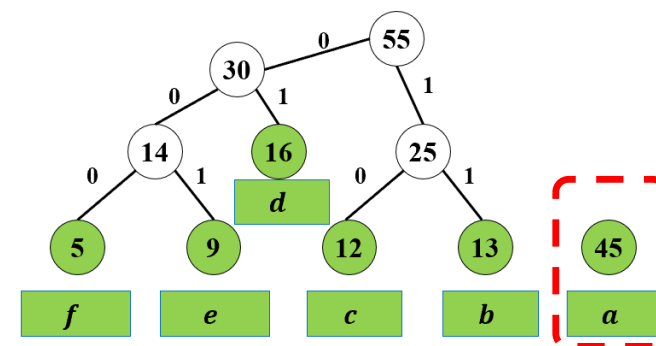
$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

选择频数最小的结点

叶子频数						45
合并频数				55		



伪代码



• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

~~$x \leftarrow \text{ExtractMin}(P, Q)$~~

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

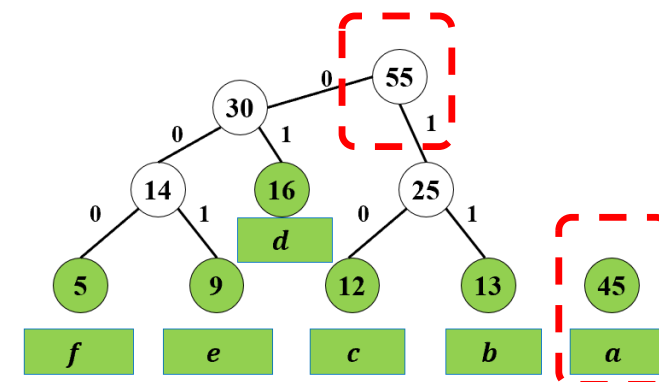
return $\text{ExtractMin}(P, Q)$

选择频数最小的结点

叶子频数

合并频数

55



伪代码



• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

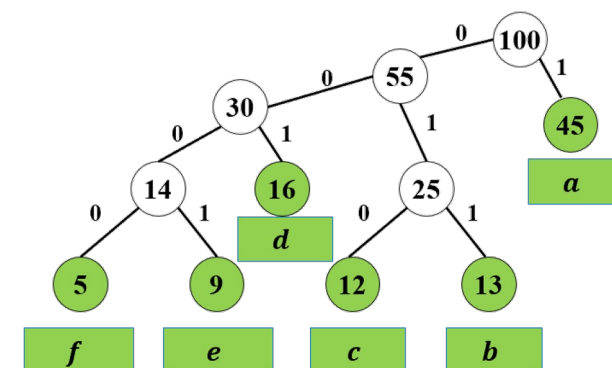
end

return $\text{ExtractMin}(P, Q)$

合并两个选择的结点

叶子频数

合并频数



伪代码



• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

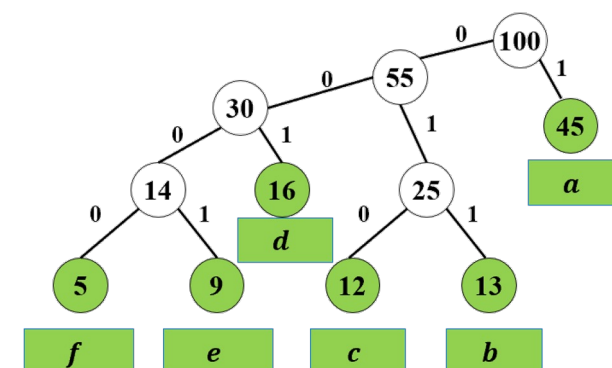
end

return $\text{ExtractMin}(P, Q)$

叶子频数

合并频数

100



存储合并后的结点

伪代码



• Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

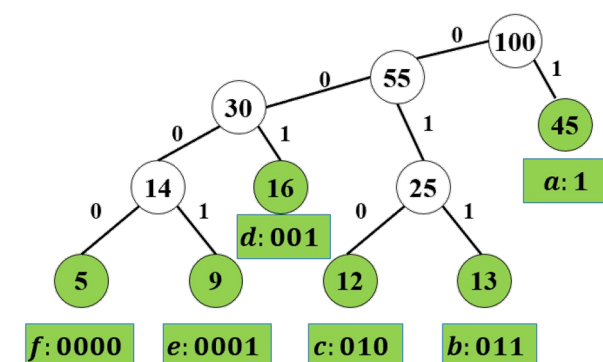
$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

返回编码树的根结点

叶子频数						
合并频数					100	





复杂度分析

● Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序 — — — — — $O(n \log n)$

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$



复杂度分析

● Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序 - - - - - $O(n \log n)$

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

} $O(n)$



复杂度分析

● Huffman(F, n)

输入: 字符数 n , 各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序 — — — — — $O(n \log n)$

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

} $O(n)$

$O(1)$

叶子频数					16	45
合并频数	14	25			100	

数组开头查找最小值 $O(1)$



复杂度分析

● Huffman(F, n)

输入: 字符数 n ,各字符频数 F

输出: 霍夫曼编码树

//预处理

将 F 递增排序 — — — — — $O(n \log n)$

新建结点数组 $P[1..n]$ 和 $Q[1..n]$

for $i \leftarrow 1$ to n do

$P[i].freq \leftarrow F[i]$

$P[i].left \leftarrow NULL$

$P[i].right \leftarrow NULL$

end

$Q \leftarrow \emptyset$

for $i \leftarrow 1$ to $n - 1$ do

 新建结点 z

$x \leftarrow \text{ExtractMin}(P, Q)$

$y \leftarrow \text{ExtractMin}(P, Q)$

$z.freq \leftarrow x.freq + y.freq$

$z.left \leftarrow x$

$z.right \leftarrow y$

$Q.Add(z)$

end

return $\text{ExtractMin}(P, Q)$

$O(n)$

$O(n)$

时间复杂度 : $O(n \log n)$



谢谢

