Ternary Disk and Huffman Tree

曹雨森

2020 年 9 月 23 日

Problem

Ternary Disk

Trimedia Disks Inc. has developed "ternary" hard disks. Each cell on a disk can now store values 0, 1, or 2 (instead of just 0 or 1).

To take advantage of this new technology, provide a modified Huffman algorithm for constructing an optimal variable-length prefix-free code for characters from an alphabet of size n, where the characters occur with known frequencies $f_1, f_2, \cdots, f_n$.

Ternary Huffman Tree

- 每个节点的后代数仅可能为 3 或为 0（叶节点）
- 内节点的左中右儿子分别编码为 0，1，2

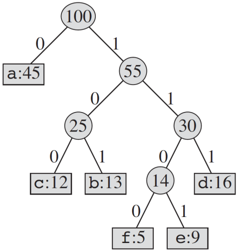


图: Binary Huffman Code

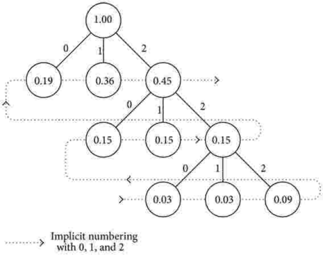
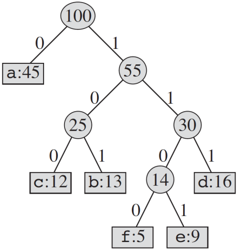


图: Ternary Huffman Code

Ternary Huffman Tree

- 每个节点的后代数仅可能为 3 或为 0（叶节点）
- 内节点的左中右儿子分别编码为 0，1，2




```
HUFFMAN(C)
n = |C|
Q = C
for i = 1 to n − 1
    allocate a new node z
    z.left = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q, z)
return EXTRACT-MIN(Q)        // return the root of the tree
```

图: Binary Huffman Code

Ternary Huffman Tree

- 每个节点的后代数仅可能为 3 或为 0（叶节点）
- 内节点的左中右儿子分别编码为 0，1，2

Ternary Huffman encoding algorithm

1. 找到三个频率最低的对象，将其合并
2. 合并这三个对象时，所得新对象的频率设置为他们的和
3. 重复上述两个操作直到剩下的对象数为 1

Ternary Huffman encoding algorithm

1. 找到三个频率最低的对象，将其合并
2. 合并这三个对象时，所得新对象的频率设置为他们的和
3. 重复上述两个操作直到剩下的对象数为 1

每次操作减少三个对象，新增一个对象，等价于每次的对象数减少了 2

- 若初始对象数为奇数，最后可以获得 1 个对象，合题意
- 若初始对象数为偶数，最后会剩下 2 个对象无法进行合并

我们可以添加一个 dummy node, 它的频率值设置为 0

贪心选择性质

Lemma 16.2

Let C be an alphabet in which each character $c \in C$ has frequency $c.freq$. Let x and y be two characters in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x and y have the same length and differ only in the last bit.

贪心选择性质

Lemma 16.2

Let C be an alphabet in which each character $c \in C$ has frequency $c.freq$. Let x and y be two characters in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x and y have the same length and differ only in the last bit.

而我们在这里需要证明的是:

Lemma 16.2*

Let C be an alphabet in which each character $c \in C$ has frequency $c.freq$. Let x , y and z be three characters in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x , y and z have the same length and differ only in the last bit.

贪心选择性质

令 a,b,c 是 T 中深度最大的兄弟叶结点:

同时不妨假定 $a.freq \leqslant b.freq \leqslant c.freq$ 且 $x.freq \leqslant y.freq \leqslant z.freq$

由于 x, y, z 三者的频率是最低的，所以我们自然有 $x.freq \leqslant a.freq$,
$y.freq \leqslant b.freq$, $z.freq \leqslant c.freq$

我们也排除六者频率同时相等的可能性（此时引理是显然成立的)。

我们在 T 中交换 x 和 a 生成一棵新树 T', 在 T' 中交换 b 和 y 生成一棵新树 T'', 再在 T'' 交换 c 和 z 生成 T''', 则在 T''' 中 x, y, z 是最深的三个兄弟叶结点。

贪心选择性质

计算 T 和 T' 的代价差 (16.4):

$$B(T) - B(T') = \sum_{c \in C} c.freq \cdot d_T(c) - \sum_{c \in C} c.freq \cdot d_{T'}(c)$$

$$= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_{T'}(x) - a.freq \cdot d_{T'}(a)$$

又因为交换了 x 和 a, 所以 $d_T(a) = d_{T'}(x), d_T(x) = d_{T'}(a)$

$$= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_T(a) - a.freq \cdot d_T(x)$$

$$= (a.freq - x.freq)(d_T(a) - d_T(x))$$

贪心选择性质

计算 T 和 T' 的代价差 (16.4)：

$$B(T) - B(T') = \sum_{c \in C} c.freq \cdot d_T(c) - \sum_{c \in C} c.freq \cdot d_{T'}(c)$$

$$= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_{T'}(x) - a.freq \cdot d_{T'}(a)$$

又因为交换了 x 和 a, 所以 $d_T(a) = d_{T'}(x), d_T(x) = d_{T'}(a)$

$$= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_T(a) - a.freq \cdot d_T(x)$$

$$= (a.freq - x.freq)(d_T(a) - d_T(x)) \geqslant 0$$

$$\geqslant 0 \qquad\qquad \geqslant 0$$

贪心选择性质

类似的我们可以得到：在 T' 中交换 b,y 和在 T'' 中交换 c,z 都不能增加代价，即 $B(T'') \leqslant B(T')$，$B(T''') \leqslant B(T'')$。

所以我们有 $B(T) \geqslant B(T') \geqslant B(T'') \geqslant B(T''')$

又因为已知的 T 已经对应了一个最优前缀码，所以 $B(T) \leqslant B(T''')$

综上可知 $B(T) = B(T'''), T'''$ 也是最优树，且 x, y, z 是最深的三个兄弟叶结点。引理得证。

最优子结构

Lemma 16.3

Let C be a given alphabet with frequency c. freq defined for each character $c \in C$. Let x and y be two characters in C with minimum frequency. Let C' be the alphabet C with the characters x and y removed and a new character z added, so that $C' = C - \{x, y\} \cup \{z\}$. Define f for C' as for C, except that $z.freq = x.freq + y.freq$. Let T' be any tree representing an optimal prefix code for the alphabet C'. Then the tree T, obtained from T' by replacing the leaf node for z with an internal node having x and y as children, represents an optimal prefix code for the alphabet C.

最优子结构

Lemma 16.3*

Let C be a given alphabet with frequency c. freq defined for each character $c \in C$. Let x, y and z be three characters in C with minimum frequency. Let C' be the alphabet C with the characters x, y and z removed and a new character t added, so that $C' = C - \{x, y, z\} \cup \{t\}$. Define f for C' as for C, except that $t.freq = x.freq + y.freq + z.freq$. Let T' be any tree representing an optimal prefix code for the alphabet C'. Then the tree T, obtained from T' by replacing the leaf node for t with an internal node having x, y and z as children, represents an optimal prefix code for the alphabet C.

最优子结构

对于每个字符 $c \in C - \{x, y, z\}$, 我们有 $d_T(c) = d_{T'}(c)$
$c.freq \cdot d_T(c) = c.freq \cdot d_{T'}(c)$。
因为 $d_T(x) = d_T(y) = d_T(z) = d_{T'}(t) + 1$

$$x.freq \cdot d_T(x) + y.freq \cdot d_T(y) + z.freq \cdot d_T(z)$$

$$= (x.freq + y.freq + z.freq)(d_{T'}(t) + 1)$$

$$= t.freq \cdot d_{T'}(t) + (x.freq + y.freq + z.freq)$$

$$\Rightarrow t.freq \cdot d_{T'}(t) - (x.freq \cdot d_T(x) + y.freq \cdot d_T(y) + z.freq \cdot d_T(z))$$

$$= x.freq + y.freq + z.freq$$

最优子结构

而又因为 $C' = C \cup \{t\} - \{x, y, z\}$，所以

$$B(T') = B(T) + t.freq \cdot d_{T'}(t) - (x.freq \cdot d_T(x) + y.freq \cdot d_T(y) + z.freq \cdot d_T(z))$$

$$\Rightarrow B(T') = B(T) - (x.freq + y.freq + z.freq)$$

最优子结构

$$B(T') = B(T) - (x.freq + y.freq + z.freq)$$

下面我们使用反证：假设 T 对应的前缀码不是 C 的最优前缀码。那么存在某个最优前缀码 T'' 满足 $B(T'') < B(T)$。
不妨取 T'' 中的三个兄弟结点 x, y, z，令 T''' 为将 T'' 中 x, y, z 及它们的父结点替换为叶结点 t 得到的树，其中 $t.freq = x.freq + y.freq + z.freq$。所以由上述公式：

$$B(T''') = B(T'') - (x.freq + y.freq + z.freq)$$

$$< B(T) - (x.freq + y.freq + z.freq) = B(T')$$

显然，这与存在这样一个 T' 对应着最优前缀码的假设矛盾。
因而 T 必然表示字母表 C 的一个最优前缀码。

Lemma 16.2*(贪心选择性质)

Let C be an alphabet in which each character $c \in C$ has frequency $c.freq$. Let x , y and z be three characters in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x , y and z have the same length and differ only in the last bit.

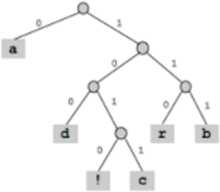
Lemma 16.3*(最优子结构)

Let C be a given alphabet with frequency c. freq defined for each character $c \in C$. Let x, y and z be three characters in C with minimum frequency. Let C' be the alphabet C with the characters x, y and z removed and a new character t added, so that $C' = C - \{x, y, z\} \cup \{t\}$. Define f for C' as for C, except that $t.freq = x.freq + y.freq + z.freq$. Let T' be any tree representing an optimal prefix code for the alphabet C'. Then the tree T, obtained from T' by replacing the leaf node for t with an internal node having x, y and z as children, represents an optimal prefix code for the alphabet C.

Ternary Huffman Algorithm 贪心算法的正确性得证

Huffman 树

Huffman 树即最优二叉树，是一种带权路径长最短的树

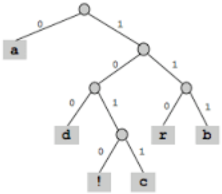


char	encoding
a	0
b	111
c	1011
d	100
r	110
!	1010

Huffman 树

Huffman 树即最优二叉树，是一种带权路径长最短的树

而最优前缀码的构建本质上也即是一棵最优树的构建



char	encoding
a	0
b	111
c	1011
d	100
r	110
!	1010

Huffman 树

构建 Huffman 树的时间复杂度:

The solution to this recurrence, by case 2 of the master theorem (Theorem 4.1), is $T(n) = O(\lg n)$. Alternatively, we can characterize the running time of MAX-HEAPIFY on a node of height h as $O(h)$.

```
HUFFMAN(C)
n = |C|
Q = C
for i = 1 to n - 1
    allocate a new node z
    z.left = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q, z)
return EXTRACT-MIN(Q)        // return the root of the tree
```

Huffman 树

构建 Huffman 树的时间复杂度:

我们用堆来保存权值，根据 HEAPIFY 函数的复杂度已知找出堆中最小元素的复杂度为 $O(lgn)$，而进行的迭代次数是 $O(n)$ 的，所以可推知构建 Huffman 树的时间复杂度是 $O(nlgn)$ 的。

```
HUFFMAN(C)
n = |C|
Q = C
for i = 1 to n - 1
    allocate a new node z
    z.left = x = EXTRACT-MIN(Q)
    z.right = y = EXTRACT-MIN(Q)
    z.freq = x.freq + y.freq
    INSERT(Q, z)
return EXTRACT-MIN(Q)      // return the root of the tree
```

Huffman 树

Ternary Disk

Trimedia Disks Inc. has developed “ternary” hard disks. Each cell on a disk can now store values 0, 1, or 2 (instead of just 0 or 1).

To take advantage of this new technology, provide a modified Huffman algorithm for constructing an optimal variable-length prefix-free code for characters from an alphabet of size n, where the characters occur with known frequencies $f_1, f_2, \cdots, f_n$.

而 Ternary Disk 这个问题本质上就是三叉哈夫曼树

Huffman 树

Ternary Disk

Trimedia Disks Inc. has developed “ternary” hard disks. Each cell on a disk can now store values 0, 1, or 2 (instead of just 0 or 1).

To take advantage of this new technology, provide a modified Huffman algorithm for constructing an optimal variable-length prefix-free code for characters from an alphabet of size n, where the characters occur with known frequencies $f_1, f_2, \cdots, f_n$.

而 Ternary Disk 这个问题本质上就是三叉哈夫曼树

事实上解决 k 叉哈夫曼树的问题是完全类似的，只要添加足够的 dummy node 来保证最后能够归结到一个节点即可

Huffman 编码和算术编码

我们以信源信号为 $\{A, B, C, D\}$ 为例，按频率将 $[0, 1)$ 分解成若干区间
假设信号输入为 CAD:

A	B	C	D
0.1	0.4	0.2	0.3
[0,0.1)	[0.1,0.5)	[0.5,0.7)	[0.7,1)

算术编码的计算方式

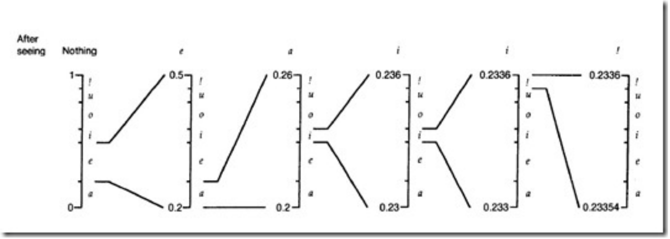
- 首先读入 C, 编码区间变为 $[0.5, 0.7)$
- 再读入 A,A 的原始区间为 $[0, 0.1)$，4，所以编码区间变为 $[0.5, 0.52)$
- 对于 D，因为其原始区间为 $[0.7, 1)$, 所以投影的编码区间为 $[0.514, 0.52)$
- 在最终编码区间中任取一个数作为编码输出即可

Huffman 编码和算术编码

我们以信源信号为 $\{A, B, C, D\}$ 为例，按频率将 $[0, 1)$ 分解成若干区间
假设信号输入为 CAD:

算术编码的计算方式

- 首先读入 C, 编码区间变为 $[0.5, 0.7)$
- 再读入 A,A 的原始区间为 $[0, 0.1)$，4，所以编码区间变为 $[0.5, 0.52)$
- 对于 D，因为其原始区间为 $[0.7, 1)$, 所以投影的编码区间为 $[0.514, 0.52)$
- 在最终编码区间中任取一个数作为编码输出即可



Huffman 编码和算术编码

事实上，Huffman 编码的效率并不会随着其位数的增加而上升，反而会下降

Bits Comparison

The binary Huffman tree uses on average 4.41 bits per letter.
The trinary Huffman tree uses on average 2.81 trits per letter

随着位数的增加，每个字母使用的位数将会减少，这造成了资源的冗余

Huffman 编码和算术编码

事实上，Huffman 编码的效率并不会随着其树的内节点的度数的增加而上升，反而会下降

Bits Comparison

The binary Huffman tree uses on average 4.41 bits per letter.
The trinary Huffman tree uses on average 2.81 trits per letter

随着度数的增加，每个字母使用的位数将会减少，这造成了资源的冗余

同时因为 Huffman 编码采取整数位的方法，其压缩率不一定是最佳的

比如字母列由两个 a,b 两个字母组成，$p(a) = 0.75, p(b) = 0.25$

按照 Huffman 编码的方式无法区分二者的概率

Huffman 编码和算术编码

- Huffman and Arithmetic coding - Performance analysis
- Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards

Acknowledgement

- [TC] Introduction to Algorithms
- Huffman coding

Thanks for your listening!