Codificação de Huffman

Guilherme Seidyo Imai Aldeia, Prof Dr. Maycon Sambinelli

Universidade Federal do ABC
Centro de Matemática, Computação e Cognição
Graduação em Ciência da Computação

22 de agosto de 2022

Tópicos

Introdução

Problema de compressão de dados

Compressão de dados: diminuir a quantidade de espaço utilizado para armazenar uma informação.

Um *codificador* faz o papel de comprimir os dados, e um *decodificador* recupera a informação original.

Código de comprimento fixo

Suponha que cada caractere é representado por uma cadeia fixa de bits. Para representar o alfabeto, precisamos de k bits, tais que $2^k \geq 26$. Ou seja, ao menos 5 bits.

Teríamos:

- a: 00000
- b: 00001
- c: 00010
- $\ldots$
- x: 01100
- y: 01101
- z: 01110

Código de comprimento fixo

Utilizando um código de comprimento fixo, cada cada caractere é representado por uma palavra de k bits.

Podemos melhorar?

Código de comprimento variável

Um código de comprimento variável associa palavras de menor tamanho para caracteres mais frequentes; enquanto palavras longas tem maior tamanho.

Código de comprimento variável

Suponha o exemplo abaixo com um alfabeto com os caracteres $[a - f]$:

	a	b	c	d	e	f
Frequência (em milhares)	45	13	12	16	9	5
Palavra em código de comprimento fixo	000	001	010	011	100	101
Palavra em código de comprimento variável	0	101	100	111	1101	1100

Temos 100.000 caracteres. A compressão utilizaria:

- 300.000 bits com comprimento fixo;
- $(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1.000 = 224.000$ bits com comprimento variável.

Código de prefixo

Código de prefixo

Um caso particular do código de comprimento variável, onde nenhuma das palavras é prefixo de outra palavra.

Sempre conseguirá a compressão de dados ótima em qualquer código de caracteres.

Codificação

Codificação: concatenamos cada palavra que representa um caractere.

Como nenhum palavra é prefixo de outra, a palavra de código que inicia o arquivo não é ambígua. Então podemos decodificar a primeira letra, e então repetir o processo com o restante dos dados codificados.

Decodificação

Decodificação: utilizamos uma árvore binária, onde as folhas representam os caracteres codificados.

Decodificação

O caminho da raiz até uma folha representa a decodificação da palavra formada.

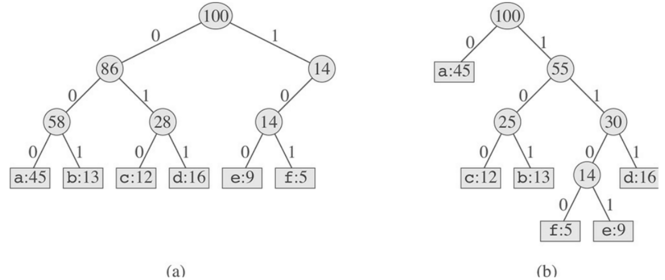


Figura 16.4 Árvores correspondentes aos esquemas de codificação na Figura 16.3. Cada folha é identificada com um caractere e sua frequência de ocorrência. Cada nó interno é identificado com a soma das frequências das folhas em sua subárvore. (**a**) A árvore correspondente ao código de comprimento fixo a = 000, ..., f = 101. (**b**) A árvore correspondente ao código de prefixo ótimo a = 0, b = 101, ..., f = 1100.

Decodificação

Se C é o alfabeto de onde os caracteres são extraídos, e todas as frequências de caracteres são não-negativas, então uma árvore para um código de prefixo ótimo tem $|C|$ folhas, uma para cada letra do alfabeto, e $|C| - 1$ nós internos.

Custo de uma árvore T

Dado uma árvore T correspondente a um código de prefixo, a quantidade de bits exigidos para armazenar os dados codificados é:

$$B(T) = \sum_{c \in C} c.freq \cdot d_T(c), \tag{1}$$

onde $c.freq$ é a frequência do caractere c, e d_T a profundidade da folha de c na árvore.

Dizemos que **o custo da árvore** é igual à quantidade de bits exigidos dada por (1).

O código de Huffman

Ideia geral do algoritmo

O código de Huffman é um algoritmo guloso, que produz um código de prefixo ótimo, se baseando em uma tabela de frequência de ocorrencia de cada caractere.

Ideia geral do algoritmo

Suponha que C seja um conjunto de n caracteres, e que cada caractere $c \in C$ tenha um atributo $c.freq$ que representa sua frequência. Ainda, seja Q uma fila de prioridade baseada no atributo $freq$, onde a prioridade é maior quanto menor a frequência.

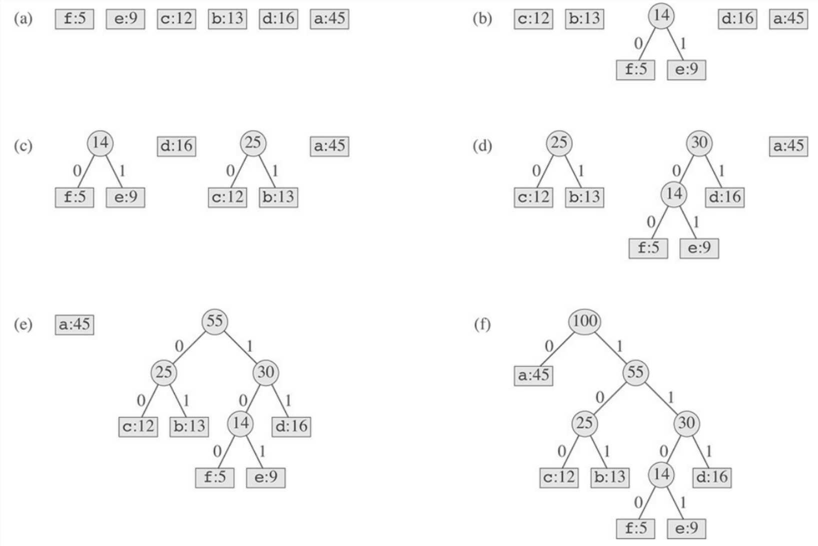
O algoritmo irá construir a árvore T de baixo para cima, começando com um conjunto de $|C|$ árvores unitárias e, por meio de $|C| - 1$ intercalações de árvores, produzirá a árvore final T.

Ao fazer uma intercalação, as árvores com as 2 menores frequências são intercaladas, criando uma nova árvore cuja frequência é a soma da frequência dos dois objetos intercalados.

Pseudo-código

```
HUFFMAN(C)
n = |C|
Q = C
for i = 1 to n − 1
        alocar um novo nó z
        z.esquerda = x = EXTRACT-MIN(Q)
        z.direita = y = EXTRACT-MIN(Q)
        z.freq = x.freq + y.freq
        INSERT(Q, z)
return EXTRACT-MIN(Q)                    // retorna a raiz da árvore.
```

Simulando o funcionamento do algoritmo



Complexidade

Suponha que Q seja implementada com uma *heap de mínimo binário*. Sua inicialização leva $O(n)$.

O laço das linhas $3-8$ é executado n vezes. Como cada operação de *heap* requer $O(\log n)$, o laço contribui com $O(n \log n)$.

A complexidade final é $O(n \log n)$.

Correção do algoritmo

Por onde começar a demonstração de correção?

1. Primeiro mostramos que o problema de determinar um código de prefixo ótimo tem:
    - Propriedades de escolha gulosa;
    - Subestrutura ótima;
2. Depois precisamos demonstrar que o algoritmo guloso em questão resolve o problema encontrando a melhor solução possível.

Lema 1 - Propriedade de escolha gulosa: Seja C um alfabeto no qual cada caractere $c \in C$ tem frequência $c.freq$. Sejam x e y dois caracteres com as frequências mais baixas. Então, existe um código de prefixo ótimo para C no qual as palavras de código para x e y tem o mesmo comprimento e diferem em apenas 1 bit.

Esse lema implica que o processo de construir uma árvore ótima por intercalações pode, sem prejuízo de generalidade, começar com a escolha gulosa de intercalar os dois caracteres de frequência mais baixa.

Ideia: tomar uma árvore T que representa o código de prefixo ótimo e modificá-la para virar T', onde x e y aparecem como folhas irmãs com profundidade máxima.

Lema 1 - Demonstração

Demonstração. Sejam a e b duas folhas de profundidade máxima.

Vamos supor que

$$a.freq \leq b.freq, \tag{2}$$

$$x.freq \leq y.freq, \tag{3}$$

sem prejuízo de generalidade.

Como x e y tem as menores frequências, e a e b tem ffrequências arbitrárias, podemos dizer que:

$$x.freq \leq a.freq, y.freq \leq b.freq, \tag{4}$$

pois se $a.freq = x.freq$ e $b.freq = y.freq$, o problema seria trivialmente verdadeiro. Precisamos supor que $x \neq b$.

Lema 1 - Demonstração

- Podemos permutar a e x em T e produzir a árvore T';
- Podemos permutar b e y em T' e produzir a árvore T''.

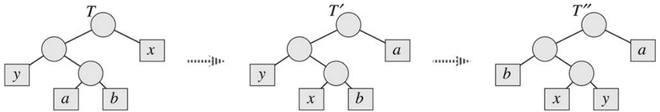


Figura 16.6 Uma ilustração da etapa fundamental na prova do Lema 16.2. Na árvore ótima *T*, as folhas *a* e *b* são duas irmãs de profundidade máxima. As folhas *x* e *y* são os dois caracteres que têm as frequências mais baixas; eles aparecem em posições arbitrárias em *T*. Considerando que x ≠ b, permutar as folhas *a* e *x* produz a árvore T', e permutar as folhas b e y produz a árvore T''. Visto que cada permuta não aumenta o custo, a árvore resultante *T''* é também uma árvore ótima.

Lema 1 - Demonstração

Vejamos a diferença de custo $B(T) - B(T')$. Precisamos que seja não-negativa, para mostrar que a escolha gulosa produz uma árvore ótima, já que assumimos T ótima.

$$\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} c.freq \cdot d_T(c) - \sum_{c \in C} c.freq \cdot d_{T'}(c) \\
&= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_{T'}(x) - a.freq \cdot d_{T'}(a) \\
&= x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_T(a) - a.freq \cdot d_T(x) \\
&= (a.freq - x.freq)(d_T(a) - d_T(x)) \geq 0,
\end{aligned} \tag{5}$$

pois os dois termos são sempre não-negativos.

De forma análoga, podemos permutar y e b em T' para gerar T'', que não aumenta o custo, e $B(T') - B(T'')$ é não-negativo.

Como $B(T'') \leq B(T)$, e assumindo T ótima, chegamos a conclusão de que T'' é ótima com x e y em folhas de profundidade máxima, da qual o lema ocorre. $\square$

Lema 2 - Relação de custos: Seja C um dado alfabeto com frequência $c.freq$ definida para cada caractere $c \in C$. Sejam x e y os dois caracteres de C com frequência mínima. Seja C' o alfabeto C do qual os caracteres x e y foram removidos e um novo caractere z foi acrescentado, de modo que $C' = C - \{x, y\} \cup \{z\}$. Defina $freq$ para C' exatamente como feito para C, mas com $z.freq = x.freq + y.freq$. Seja T' qualquer árvore que represente um código de prefixo ótimo para o alfabeto C'. Então A relação entre os custos é:

$$B(T) = B(T') + x.freq + y.freq, \tag{6}$$

que equivale a:

$$B(T) = B(T') + z.freq, \tag{7}$$

Lema 2 - Relação de custos

Ideia: Queremos mostrar a equivalência entre duas árvores que foram construídas sobre alfabetos que diferem apenas em relação a 3 caracteres. Precisamos mostrar que é possível expressar o custo de ambas as árvores por meio de valores que possam ser calculados para as duas, e assim igualar os custos em uma expressão.

Demonstração. Para descrever a relação entre os custos de T e T', primeiro note que, para cada $c \in C \cap C'$, vale que $d_T(c) = d_{T'}(c)$ e, portanto, o custo de T e T' difere apenas em termos de x, y e z.

Note também que:

$$d_T(x) = d_T(y) = d_{T'}(z) + 1. \tag{8}$$

O custo, para a árvore T é calculado por:

$$\begin{aligned} B(T) &= \sum_{c \in C} c.freq \cdot d_T(c) \\ &= \sum_{c \in C \setminus \{x,y\}} c.freq \cdot d_T(c) + x.freq \cdot d_T(x) + y.freq \cdot d_T(y), \end{aligned} \tag{9}$$

E, para a árvore T':

$$\begin{aligned} B(T') &= \sum_{c \in C'} c.freq \cdot d_{T'}(c) \\ &= \sum_{c \in C' \setminus \{z\}} c.freq \cdot d_{T'}(c) + z.freq \cdot d_{T'}(z) \\ &= \sum_{c \in C \setminus \{x,y\}} c.freq \cdot d_T(c) + z.freq \cdot d_{T'}(z). \end{aligned} \tag{10}$$

Lema 2 - Demonstração

Isolando em 9 e 10 o somatório em comum:

$$\sum_{c \in C \setminus \{x,y\}} c.freq \cdot d_T(c) = B(T) - x.freq \cdot d_T(x) - y.freq \cdot d_T(y), \tag{11}$$

$$\sum_{c \in C \setminus \{x,y\}} c.freq \cdot d_T(c) = B(T') - z.freq \cdot d_{T'}(z), \tag{12}$$

Igualando 11 e 12:

$$B(T) - x.freq \cdot d_T(x) - y.freq \cdot d_T(y) = B(T') - z.freq \cdot d_{T'}(z), \tag{13}$$

$$B(T) = B(T') - z.freq \cdot d_{T'}(z) + x.freq \cdot d_T(x) + y.freq \cdot d_T(y). \tag{14}$$

Lema 2 - Demonstração

Por 8, temos

$$B(T) = B(T') - z.freq \cdot (d_T(x) - 1) + x.freq \cdot d_T(x) + y.freq \cdot d_T(x), \tag{15}$$

$$B(T) = B(T') - z.freq \cdot (d_T(x) - 1) + d_T(x) \cdot (x.freq + y.freq), \tag{16}$$

Lema 2 - Demonstração

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

$$B(T) = B(T') - (x.freq + y.freq) \cdot (d_T(x) - 1) + d_T(x) \cdot (x.freq + y.freq), \tag{17}$$

$$B(T) = B(T') + (x.freq + y.freq) \cdot (d_T(x) - d_T(x) + 1). \tag{18}$$

Então, temos:

$$B(T) = B(T') + x.freq + y.freq. \tag{19}$$

Como queríamos demonstrar. $\square$

Lema 3 - Subestrutura ótima: Seja C um dado alfabeto com frequência $c.freq$ definida para cada caractere $c \in C$. Sejam x e y os dois caracteres de C com frequência mínima. Seja C' o alfabeto C do qual os caracteres x e y foram removidos e um novo caractere z foi acrescentado, de modo que $C' = C - \{x, y\} \cup \{z\}$. Defina $freq$ para C' exatamente como feito para C, mas com $z.freq = x.freq + y.freq$. Seja T' qualquer árvore que represente um código de prefixo ótimo para o alfabeto C'. Então a árvore T, obtida de T' pela substituição do nó folha com z por um nó interno com filhos x e y, representa um código de prefixo ótimo para o alfabeto C.

Lema 3 - Sub-estrutura ótima

O lema nos diz que, dado uma árvore T' onde o caractere z, obtido combinando os caracteres de menor frequência x e y, então a árvore T obtida substituíndo z por um nó intermediário com x e y como filhos é ótima.

Ideia: vimos como expressar o custo $B(T)$ em termos de $B(T')$. Vamos supor, por contradição, que T não é ótima, e então concluir que tal suposição seria um absurdo.

Lema 3 - Demonstração

Agora, mostraremos por contradição que T é ótimo. Suponha, por contradição, que $\exists T^*$ tal que $B(T^*) < B(T)$. Pelo Lema 1, podemos escolher T^* tendo x e y nas folhas mais profundas. Seja T'' a árvore obtida de T^* pela remoção de x e y e a substituição de seu pai por z.

Ou seja, T está para T' assim como T^* está para T''.

Lema 3 - Demonstração

Pelo Lema 2, sabemos expressar $B(T^*)$ em termos de $B(T'')$, assim como $B(T')$ em termos de $B(T)$:

$$B(T^*) = B(T'') + x.freq + y.freq \tag{20}$$

$$B(T) = B(T') + x.freq + y.freq \tag{21}$$

Como supomos que T' é ótima, sabemos que $B(T'') \leq B(T')$. Então vale:

$$\begin{aligned} B(T^*) &= B(T'') + x.freq + y.freq \\ &\geq B(T') + x.freq + y.freq, \end{aligned} \tag{22}$$

Por 21, podemos expressar o custo de $B(T')$ em relação a $B(T)$:

$$\begin{aligned} B(T^*) &= B(T'') + x.freq + y.freq \\ &\geq B(T') + x.freq + y.freq = B(T), \end{aligned} \tag{23}$$

uma contradição, pois assumimos que T não é ótima. Portanto, vale que T é ótima. $\square$

Teorema - Correção do algoritmo de Huffman

Segue diretamente dos Lemas 1 e 2.