4
Códigos de Huffman

4.1
Árvores de código

Numa árvore qualquer, os nós que não têm filhos são chamados de **folhas** e os demais nós são chamados de **nós internos**.

Se um nó de uma árvore é filho de outro, então há um arco ligando os dois nós. Um caminho P ligando dois nós da árvore é dado por uma seqüência de arcos conexos. O tamanho de P é calculado pelo número de arcos do seu caminho.

O **nível** de um nó em uma árvore é igual ao tamanho do caminho que conecta a raiz da árvore a esse nó.

A **altura** de uma árvore é dada pelo nível do nó de maior nível da árvore. Assim, as árvores da figura 4.1 têm alturas iguais a 2, 3 e 4, respectivamente.

Árvore binária é aquela cujos nós têm, no máximo, dois filhos. Se além disso, a ordem dos filhos de cada nó também é levada em conta, então temos uma **árvore binária ordenada**.

Os códigos de prefixo são usualmente representados por árvores binárias ordenadas. As árvores da figura 4.1 são árvores binárias ordenadas.

Existe uma relação de um para um entre código de prefixo com n palavras de código e árvores binárias ordenadas com n folhas. Todo código de prefixo C pode ser representado por uma árvore binária ordenada e esta árvore é única para esse código. Cada folha da árvore está associada a uma palavra de código c_i de C. Esta palavra de código é determinada pelo caminho na árvore que a conecta à raiz da mesma. A regra geral para formar os bits de uma palavra de código é, partindo da raiz, percorrer o caminho até a folha correspondente, de tal modo que, cada arco percorrido para a direita equivale, por exemplo, a um bit '1' na palavra de código e cada arco percorrido para a esquerda equivale a um bit '0'.

A figura 4.1 mostra as árvores binárias que representam os códigos de prefixo 1, 4 e 5 da tabela 2.2. Os códigos 2 e 3 não têm representações em árvores binárias com palavras de código nas folhas.

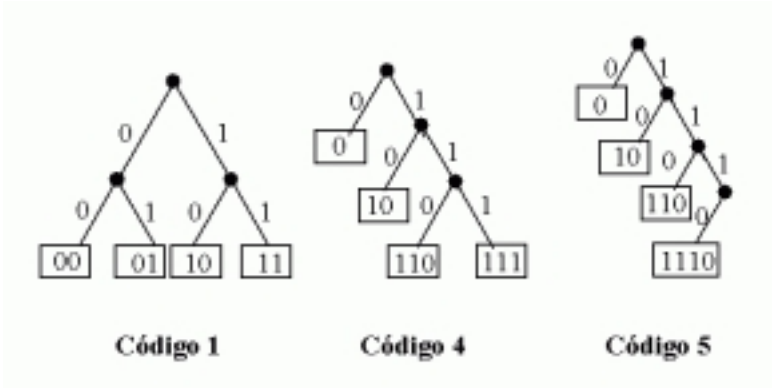

Figura 4.1: Representação de códigos binários por árvores binárias.

Uma árvore é estritamente binária quando todos os seus nós internos têm exatamente dois filhos. As árvores representativas dos códigos 1 e 4 da figura 4.1 são estritamente binárias enquanto que aquela que representa o código 5 não.

Toda árvore estritamente binária com n nós internos tem $n + 1$ folhas. Um código livre de prefixo associado a uma árvore estritamente binária é dito um código minimal ou completo. Todo código ótimo é minimal.

Seja um texto $T = t_1.t_2...tm$, obtido de uma fonte independente e codificado por $C = c_1.c_2....c_m$. Supondo que cada palavra de código c_i seja suficiente para carregar a quantidade de informação contida no seu símbolo correspondente em T, a entropia fornece um limite inferior para o comprimento médio das palavras de um código utilizado para a codificação de T. Pode-se entender esse limite inferior do comprimento médio como o número esperado de bits por símbolo em um código C.

4.2 Códigos de Huffman

Uma das melhores técnicas de compressão conhecidas é devida a Huffman (7). Para uma dada distribuição de probabilidades gera um código ótimo, livre de prefixo e de redundância mínima, além de produzir uma sequência de bits aleatórios. Utiliza códigos de tamanho variável para representar os símbolos do texto, que podem ser caracteres ou cadeias de caracteres (digramas, trigramas, n-gramas ou palavras). A idéia básica do algoritmo é atribuir códigos de bits menores para os símbolos mais freqüentes no texto, e códigos mais longos para os mais raros.

O algoritmo de Huffman original baseia-se no método guloso e constrói um código ótimo com esforço computacional $O(n.logn)$. Existem novas versões que constroem o código de Huffman com esforço computacional $O(n)$, mas

Tabela 4.1: Frequências por símbolo.

Símbolo	Frequência
0	0,20
1	0,25
2	0,15
3	0,08
4	0,07
5	0,06
6	0,05
7	0,05
8	0,05
9	0,04

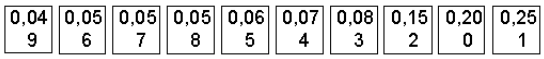

Figura 4.2: (a) Passo 1 - Início da construção da árvore de Huffman.

é necessário que as frequências já estejam ordenadas. Como o problema de ordenação tem cota inferior $\Omega(n.logn)$, isto é, não existe um algoritmo que consiga ordenar as frequências com esforço computacional menor, o esforço computacional para construir o código de Huffman é $\Theta(n.logn)$.

4.2.1 Construção da árvore de Huffman

O algoritmo constrói uma árvore binária completa com n nós externos e $n-1$ nós internos, onde os nós externos são rotulados com as frequências.

Como exemplo do funcionamento do algoritmo de Huffman utilizaremos os dados fornecidos na tabela 4.1.

Inicialmente, existe um conjunto de árvores sem filhos, uma para cada frequência na lista, ordenadas pela frequência, conforme a figura 4.2.

A seguir, as duas menores freqüências são colocadas em uma nova árvore cuja frequência é a soma destas duas e cuja raiz tem dois filhos que são as duas menores freqüências. A nova árvore é inserida na lista e as duas frequências das quais foi criada são retiradas, conforme a figura 4.3.

Na sequência, novamente as duas menores freqüências são retiradas e é criada uma nova árvore e inserida na lista, conforme a figura 4.4.

E o processo se repete como na figura 4.5, até que a árvore final de Huffman é gerada, conforme pode ser visualizado na figura 4.6.

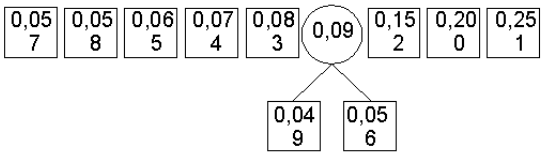

Figura 4.3: (b) Passo 2.

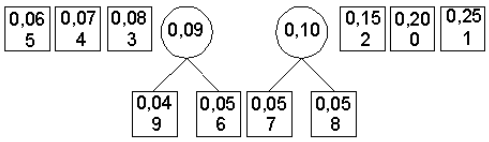

Figura 4.4: (c) Passo 3.

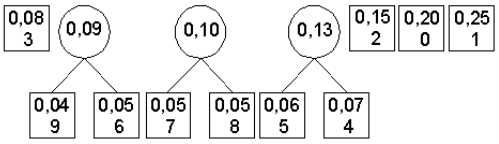

Figura 4.5: (d) Passo 4.

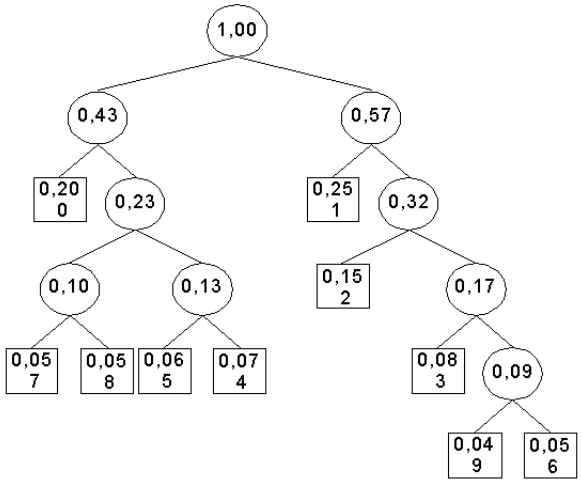

Figura 4.6: (e) Passo 5 - Árvore de Huffman final.

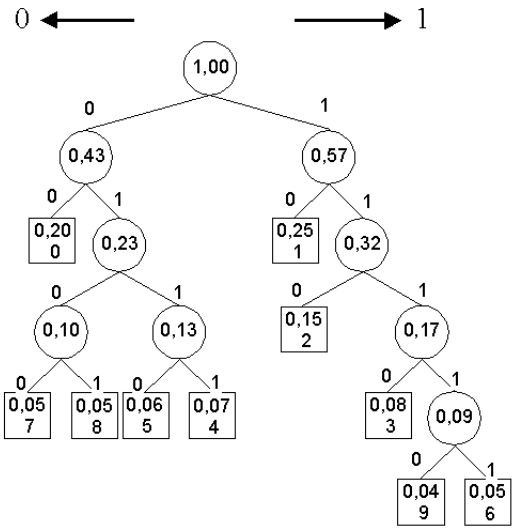

Figura 4.7: Navegação na árvore de Huffman.

Tabela 4.2: Códigos de Huffman por símbolo.

Símbolo	Código de Hufffman
0	00
1	10
2	110
3	1110
4	0111
5	0110
6	11111
7	0100
8	0101
9	11110

Para achar o código de cada símbolo, basta navegar na árvore de Huffman da raiz até a folha, concatenando os valores encontrados para cada nó da árvore: bit 0 (zero) ao navegar na subárvore da esquerda, e bit 1 (um) ao navegar na subárvore da direita, conforme a figura 4.7. A tabela 4.2 lista os códigos de Huffman resultantes da concatenação dos bits pela navegação na árvore.

Observando a tabela 4.2 notamos que os códigos mais curtos, 00 e 10, são atribuídos aos símbolos mais frequentes, enquanto os códigos mais longos são atribuídos aos símbolos relativamente mais raros.

Dependendo do processo de construção da árvore de Huffman e da atribuição dos bits 0 e 1, podemos obter códigos diferentes dos listados na tabela 4.2. Entretanto, isto não afeta a entropia do código.

O algoritmo de Huffman produz um código de comprimento médio

mínimo. Consequentemente, este algoritmo resolve o problema da minimização do comprimento médio do código.

Seja l_i o comprimento do caminho da raiz da árvore até a folha onde se encontra o i-ésimo caracter. Logo, l_i é o tamanho do código do símbolo i.

Calculando-se o **comprimento médio do código** para o exemplo desenvolvido temos

$$\bar{l} = \sum_{i=1}^{n} f_i.l_i = 3,04$$

Desta forma, temos uma média de 3,04 bits por símbolo, isto é, para um texto contendo 1000 símbolos são necessários 3040 bits para codificá-lo, enquanto que são necessários 8000 bits ao se utilizar códigos de tamanho fixo (EBCDIC, ASCII, etc.).

Como o código produzido pelo algoritmo de Huffman corresponde a folhas da árvore, fica claro que esse código é isento de prefixo, isto é, nenhum código é prefixo de algum outro. Consequentemente, nenhum texto é decodificado de modo ambíguo.

A decodificação implica em examinar cada bit recebido e verificar se forma um código válido. Caso contrário, examina-se o bit seguinte e verifica-se se a sequência é um código válido, e assim sucessivamente.

Por exemplo, para a mensagem 01100011010, temos a seguinte decodificação: os primeiros 4 bits (0110) são o símbolo 5; os dois bits seguintes (00) são o símbolo 0; os próximos três bits (110) são o símbolo 2; e os últimos 2 bits (10) são o símbolo 1. Desta forma, o texto 01100011010 é decodificado para 5021.

Na construção da árvore de Huffman não especificamos como proceder com os empates. Na verdade, qualquer que seja a escolha levará a construção de uma árvore de Huffman ótima. Entretanto, se em todo empate envolvendo uma folha e um nó interno, a folha é escolhida para participar da combinação, então a árvore de Huffman tem **altura mínima**. Em contrapartida, se sempre um nó interno for escolhido, a árvore de Huffman tem **altura máxima**.

Os códigos de Huffman que utilizam caracteres como símbolos do alfabeto são chamados de **Huffman de caracteres** ou Huffman simples. Para melhores taxas de compressão, utiliza-se o **Huffman de palavras**, onde cada símbolo é uma cadeia de caracteres delimitada por separadores (vírgula, ponto, ponto-e-vírgula, etc.). Existe ainda o Huffman de n-gramas, onde os símbolos são n-uplas de caracteres.

4.3 Códigos de Huffman canônicos

Existe uma representação ligeiramente diferente do código de Huffman que é muito eficiente para a decodificação, mesmo que os modelos sejam extremamente grandes, o que pode acontecer em sistemas de recuperação de informação com grande volume de texto. Esta representação é chamada de **código de Huffman canônico**. Resulta nos mesmos tamanhos do código de Huffman, mas necessita de uma representação particular para os códigos.

Um código canônico é estruturado para permitir uma decodificação extremamente rápida, necessitando apenas de alguns bits por símbolo do alfabeto. O código é chamado de canônico porque muito do não determinismo dos códigos de Huffman convencionais é evitado. Por exemplo, na construção normal da árvore de Huffman, foi assumida a convenção de usar o bit 0 para indicar o ramo esquerdo e o bit 1 para indicar o direito (codificação *0-left*), mas uma escolha diferente conduz a diferentes códigos, mas igualmente válidos.

4.3.1 Construção de códigos de Huffman canônicos

Os códigos de Huffman canônicos são gerados através de um algoritmo que leva em consideração apenas o tamanho do código de Huffman para o símbolo, ou seja, a altura do símbolo na árvore.

Na tabela 4.3, temos os códigos de Huffman Canônico para a codificação de Huffman original da tabela 4.2.

A construção da árvore de Huffman canônica é detalhada por Moffat et al. (22).

Visualmente, a árvore canônica é a mesma árvore de Huffman, onde todos os nós estão deslocados para a esquerda (ou direita) dentro de cada nível da árvore. A figura 4.8 representa a árvore canônica correspondente aos códigos de Huffman Canônicos da tabela 4.3 .

Para construir a árvore, basta conhecer o tamanho do código de Huffman para cada símbolo, e a partir daí posicionam-se as folhas nos níveis adequados, da esquerda para a direita, ordenados ascendentemente pela freqüência. Depois, completa-se a árvore criando os nós internos. O algoritmo de Huffman tradicional pode ser utilizado para determinar os tamanhos dos códigos. O processo de construção dos códigos de Huffman Canônicos é rápido e econômico. Não é necessária a construção de uma árvore realmente, o algoritmo utiliza apenas o comprimento do código de Huffman para derivar o código Canônico, e a grande vantagem é um ganho substancial no processo de decodificação.

Tabela 4.3: Códigos de Huffman canônico.

Símbolo	Huffman	Tamanho do código	Huffman canônico
0	00	2	10
1	10	2	11
2	110	3	011
3	1110	4	0101
4	0111	4	0100
5	0110	4	0011
6	11111	5	00001
7	0100	4	0001
8	1101	4	0010
9	11110	5	00000

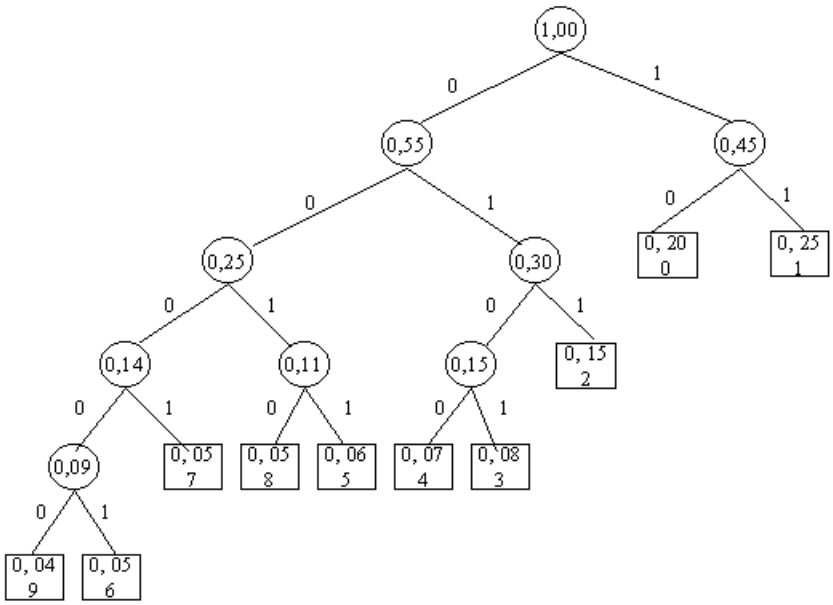

Figura 4.8: Árvore de Huffman canônica.