UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

Ciência da Computação

ESTRUTURA DE DADOS II

RESUMO COMPRESSÃO DE TEXTOS

Gabriel Mace do Santos Ferreira

Marcus Vinícius Souza Fernandes

Ouro Preto

2021

Introdução a compressão de textos

A compressão de textos é um método empregado por meio de um conjunto de técnicas que visam a redução do tamanho de um arquivo texto, sendo necessária a conversão dos símbolos textuais por estruturas que ocupam um espaço menor, como bytes ou bits.

Vale lembrar que em algum algoritmo de compressão ideal, a velocidade de descompressão é mais importante do que a velocidade de compressão, isso se dá, pois na busca de um elemento é necessário que o mesmo seja retornado rapidamente, portanto a descompressão é muito importante.

Vantagens:

- Os itens comprimidos ocupam um menor espaço de armazenamento, acarretando em uma pesquisa mais rápida, bem como operações efetuadas no arquivo com tempo menor.
- É possível realizar o casamento de cadeias em um texto comprimido, além disso, a busca sequencial em um texto comprimido pode ser mais eficiente do que em um arquivo descomprimido, dado que seu tamanho não é menor.
- Acesso direto a qualquer parte do texto, ademais, é possível realizar a descompressão do texto a partir de qualquer parte desejada.

  Obs: O segundo e terceiro itens dependem da implementação do algoritmo de compressão, no entanto, a forma ideal seria utilizando estas propriedades.

Desvantagens:

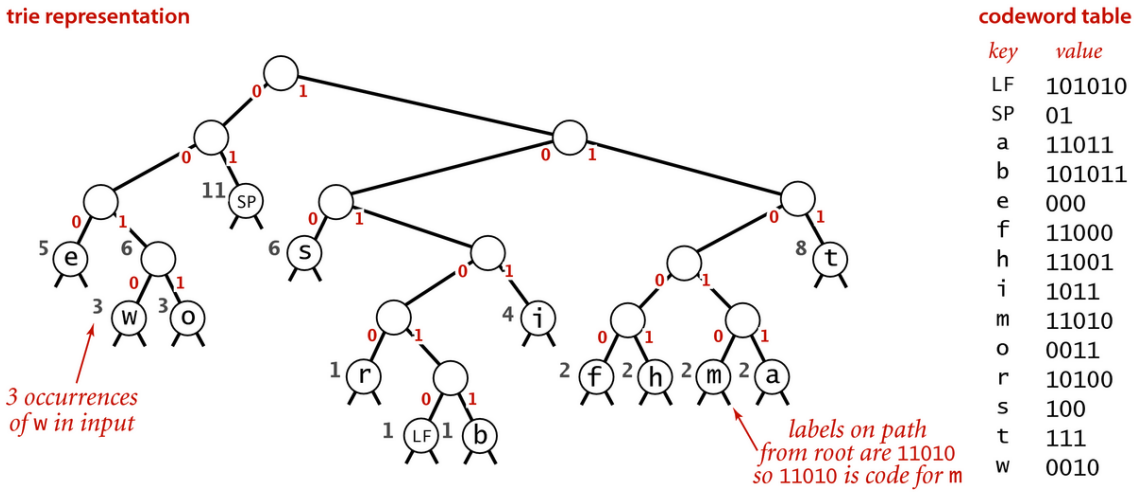
- Custo computacional para decodificar e codificar o texto.

É possível calcular a eficácia de diferentes métodos de compressão de texto por meio do ganho de espaço obtido (razão de compressão), ou seja, a porcentagem do tamanho do arquivo comprimido em relação ao arquivo original em questão.

Compressão de Huffman

A compressão de Huffman é considerado um dos melhores algoritmos de compressão atualmente. É atribuído um código único de tamanho variável aos diferentes símbolos que se encontram no texto. Os símbolos de um determinado texto são equivalentes às palavras do mesmo. Códigos menores são atribuídos aos símbolos que possuem alta frequência em um texto.

Obs: Nas implementações iniciais de Huffman os caracteres eram considerados como símbolo, já atualmente as palavras são consideradas como tal. Houve essa mudança, devido ao fato de que a razão de compressão seria menor.



Huffman code for the character stream "it was the best of times it was the worst of times LF"

Etapas do algoritmo da compressão de Huffman:

As duas etapas correspondem a passadas distintas do algoritmo sobre o texto em questão.

1. Inicialmente são contadas as diferentes palavras do texto, a fim de estabelecer sua frequência, bem como atribuir um código a essas. Vale ressaltar que a tabela de símbolos do codificador é idêntica ao vocabulário do texto, o que permite a integração entre o método de compressão e arquivo invertido.

2. Durante a segunda a passada são substituídas as palavras por seus códigos de Huffman, correspondendo à etapa de compressão em si.

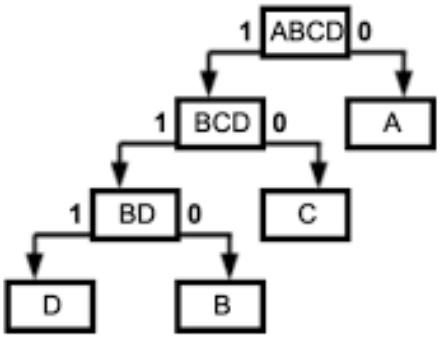
   Obs: É importante ressaltar que dentro de um texto existem caracteres separadores(Exemplo: “,” ; ”!”; “ ”) e palavras. Dessa forma, é necessário tratar esses separadores, por meio da conversão de separadores em códigos, no entanto será considerado que o espaço em branco simples deve ser inserido ao final de cada uma das palavras.

Árvore de codificação

O algoritmo de codificação de Huffman se baseia na árvore de codificação, inicialmente é composto o vocabulário do texto, bem como a frequência das palavras contidas nele, essas palavras e suas frequências irão compor os **n** nós folhas da árvore de codificação.

A partir das folhas será criado o resto da árvore, por meio da união dos nós com menor frequência em uma subárvore que possui o nodo mãe composto pela combinação das frequências de suas folhas. O processo será repetido até a última palavra ser inserida na árvore, é importante ressaltar que durante as iterações, caso existam duas subárvores o algoritmo irá combiná-las entre si .

Ao final do processo a árvore se encontrará na forma canônica, ou seja, a subárvore à direita deve ser maior do que a esquerda.



Símbolo	Código
A	0
B	110
C	10
D	111

Vantagens:

- Facilita a visualização e sugere métodos triviais de compressão e descompressão, tais eles:
    - Codificação: Na compressão a árvore emite bits ao longo de suas arestas enquanto é percorrida.
    - Decodificação: Na descompressão os bits de entrada são usados para selecionar as arestas.

Desvantagens:

- A abordagem é ineficiente em termos de espaço e tempo.

Algoritmo de Moffat e Katajainen

Esse algoritmo é baseado na codificação canônica, no entanto apresenta comportamento linear em tempo e espaço.

O funcionamento do algoritmo ocorre através do cálculo do comprimento do código dos itens, baseado nas palavras encontradas no vocabulário do texto e a

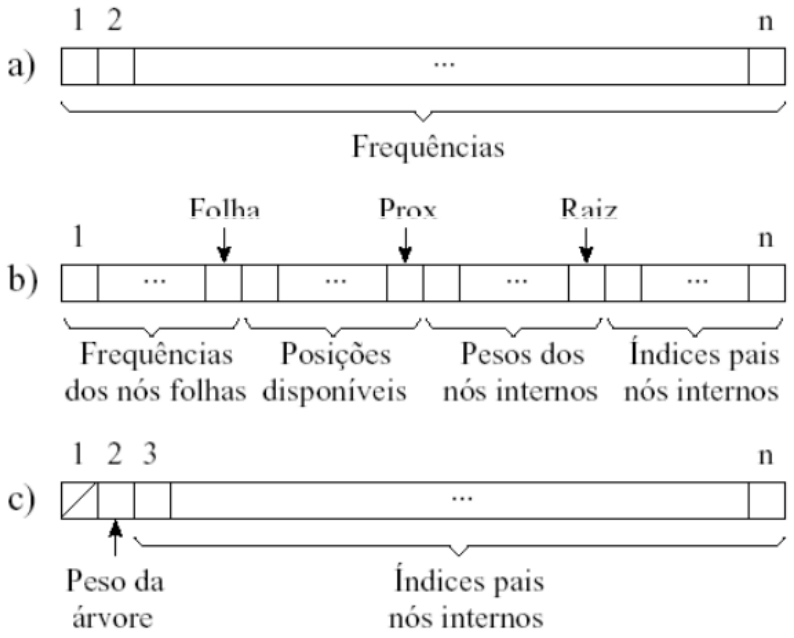
frequência dessas, ademais, a compressão atingida é semelhante independente do código utilizado no processo.

Após o cálculo do comprimento das palavras são efetuados os métodos de compressão e descompressão de forma eficiente.

Inicialmente o algoritmo recebe um vetor **A**, ordenado decrescentemente, contendo as frequências das palavras. Durante a execução são utilizados vetores logicamente distintos porém com a mesma coexistência do vetor **A**.

A seguir são realizadas as etapas:

1. **Combinação dos nós:**

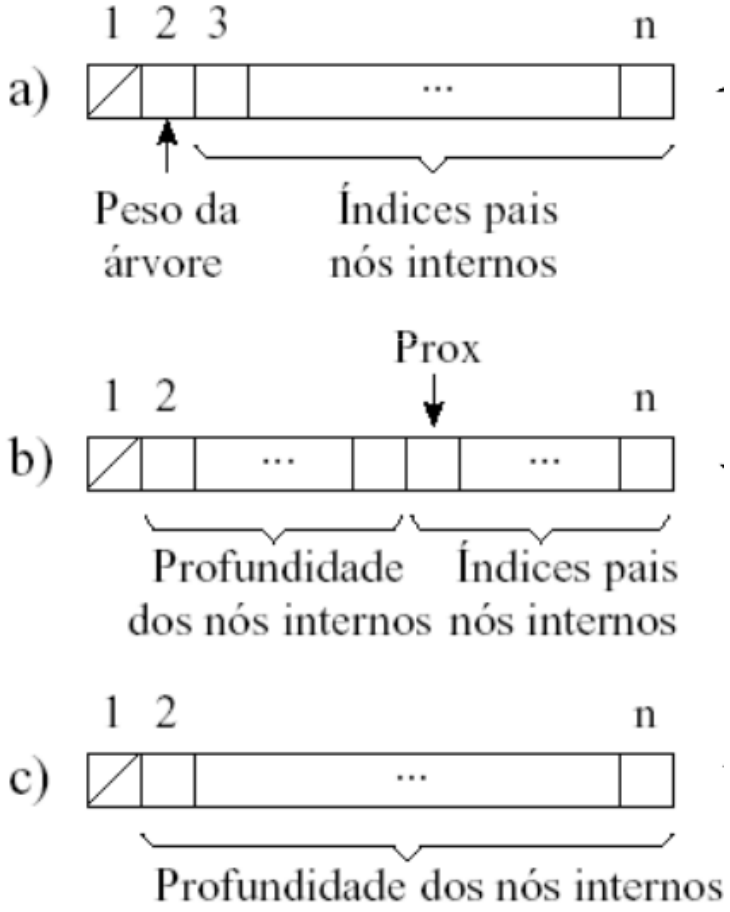


a) Na medida que as frequências são combinadas, elas se transformam em “pesos”, sendo cada um desses pesos a soma da combinação das frequências e/ou pesos.

b) O vetor **A** é percorrido no sentido direita-esquerda, sendo manipuladas 4 listas. A **Raiz** é o próximo nó a ser processado, o **Prox** é a próxima posição a ser disponibilizada para um nó interno e a **Folha** é o próximo nó folha a ser processado.

c) A situação alcançada ao final do processamento da 1º fase: Peso da árvore (**A[2]**) e os índices dos pais dos nós internos. A posição **A[1]** não é usada devido uma árvore com **n** nós folhas necessitar de (**n-1**) nós internos.

2. Determinação da profundidade dos nós:

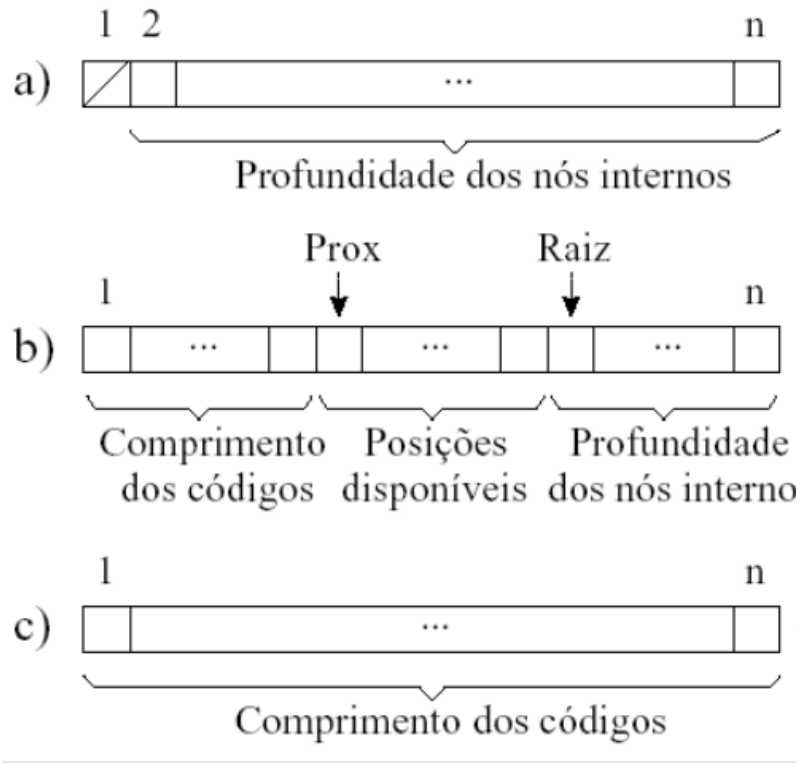


a) Referente ao resultado da primeira fase. O vetor **A** convertido, esquerda-direita, na profundidade dos nós internos.

b) **Prox** é o próximo índice de pai dos nodos internos a ser processado. **A[2]** representa neste instante a raiz da árvore. Fazendo **A[2]=0** e **A[Prox]=A[A[Prox]]+1** chegamos ao que se deseja (profundidade dos nós internos), ressaltando que **Prox** varia de 3 a **n**.

c) A situação alcançada ao final do processamento da 2º fase: A profundidade dos nós internos, a posição **A[1]** não é utilizada devido ao fato de que uma árvore com **n** nós folhas, são necessários (**n-1**) nós internos.

3. **Determinação da profundidade dos nós folhas (comprimento dos códigos):**



a) Resultado da 2º fase. A partir deste ponto são calculados as profundidades dos nós folhas, os quais representam os comprimentos dos códigos.

b) O vetor **A** é percorrido no sentido esquerda-direita, sendo manipuladas 3 listas. A **Raiz** é o próximo nó interno a ser processado, o **Prox** é a posição na qual o próximo comprimento de código deve ser alocado.

c) A situação alcançada ao final do processamento da 3º fase: Profundidade dos nós folhas (comprimento dos códigos).

Obs: Para exemplificar os passos foram utilizadas as imagens do slide referente ao material do professor Guilherme Tavares de Assis, uma vez que ilustrações de qualidade sobre o tema são escassas na internet.

Obtenção de Códigos Canônicos

Algumas das propriedades dos códigos canônicos são:

- O comprimento dos códigos seguem o algoritmo de Huffman.
- Os códigos de mesmo comprimento são inteiros consecutivos.

A partir dos comprimentos obtidos pelo algoritmo de Moffat e Katajainen, o cálculo dos códigos se torna simples:

- Inicialmente o primeiro código é composto por apenas **zeros**.
- Para os demais, se adiciona 1 ao código anterior e é realizado um deslocamento à esquerda para obter o comprimento adequado e necessário.

Codificação e Decodificação

Os algoritmos de codificação e decodificação são baseados na ideia de que códigos de mesmo comprimento são inteiros consecutivos.

A codificação e decodificação funcionam por meio do uso de dois vetores de comprimento equivalente ao maior código existente (**MaxCompCod**). Esses vetores são conhecidos como:

Base: Indica o valor inteiro do primeiro com comprimento **c,** sendo esse valor decidido anteriormente.

Base[c] = 0, se **c** = 1
Base[c] = 2 * (**Base[c-1] + w**$_{c-1}$), caso contrário.
w = Número de códigos com comprimento (**c - 1**)

Offset: Indica o índice do vocabulário da primeira palavra com comprimento **c,** dado que esse comprimento é estabelecido previamente.

Compressão

O processo de compressão é realizado em 3 etapas, sendo elas:

1. O arquivo texto é percorrido e o vocabulário é gerado juntamente com a frequência referente a cada palavra. Uma tabela hash com tratamento de colisão é utilizada para operações de pesquisa e inserção no vetor de vocabulário, assegurando o custo O(1).

2. O vetor de vocabulário é ordenado pela frequência de suas palavras, em seguida, é calculado o comprimento dos códigos, os vetores **Base**, **Offset** e **Vocabulário** são gerados e gravados no início do arquivo comprimido.

3. O arquivo texto é percorrido novamente, as palavras são extraídas e codificadas e por fim, os códigos correspondentes são escritos no arquivo comprimido.

Decompressão

O processo de descompressão se comparado a compressão é mais simples. Sendo composto por duas etapas:

1. Leitura dos vetores gravados no começo do arquivo comprimido: **Base**, **Offset** e **Vocabulário.**

2. Leitura dos códigos do arquivo comprimido, de forma a decodificá-los e substituí-los pelas palavras equivalentes no arquivo texto.