**Compressão de texto**

A compressão de texto consiste em representar o texto original de documentos em menos espaço. Deve-se substituir os símbolos do texto por outros que ocupam um número menor de *bits* ou *bytes*.

Ganho obtido: o texto comprimido ocupa menos espaço de armazenamento, levando menos tempo para ser pesquisado e para ser lido do disco ou transmitido por um canal de comunicação.

Preço a pagar: custo computacional para codificar e decodificar o texto.

Além da economia de espaço, outros aspectos relevantes são:

Velocidade de compressão e descompressão (Em muitas situações, a velocidade de descompressão é mais importante que a compressão);

Possibilidade de realizar casamento de cadeias diretamente no texto comprimido (A busca sequencial da cadeia comprimida pode ser bem mais eficiente do que descomprimir o texto a ser pesquisado);

Acesso direto a qualquer parte do texto comprimido, possibilitando o início da descompressão a partir da parte acessada (Um sistema de recuperação de informações para grandes coleções de documentos que estejam comprimidos necessita acesso direto a qualquer ponto do texto comprimido);

Razão de compressão corresponde à porcentagem que o arquivo comprimido representa em relação ao tamanho do arquivo não comprimido, sendo utilizada para medir o ganho em espaço obtido por um método de compressão. Ex.: se o arquivo não comprimido possui 100 *bytes* e o arquivo comprimido possui 30 *bytes*, a razão é de 30%.

**Huffman**

Um método de codificação bem conhecido e utilizado é o de Huffman, proposto em 1952. Um código único, de tamanho variável, é atribuído a cada símbolo diferente do texto, códigos mais curtos são atribuídos a símbolos com frequência mais altas. \*As implementações tradicionais do método de Huffman consideram caracteres como símbolos.

Para atender as necessidades dos sistemas de RI, deve-se considerar palavras como símbolos a serem codificados. \*Métodos de Huffman baseados em caracteres e em palavras comprimem o texto para cerca de 60% e 25%, respectivamente.

**Compressão de Huffman Usando palavras**

Corresponde à técnica de compressão mais eficaz para textos em linguagem natural. Inicialmente, considera cada palavra diferente do texto como um símbolo, contando suas frequências e gerando um código de Huffman para as mesmas, a seguir, comprime o texto substituindo cada palavra pelo seu código correspondente. \*A tabela de símbolos do codificador é exatamente o vocabulário do texto, o que permite uma integração natural entre o método de compressão e arquivo invertido (sistema de RI).

A compressão é realizada em duas passadas sobre o texto:

Obtenção da frequência de cada palavra diferente.

Realização da compressão.

Um texto em linguagem natural é constituído de palavras e de separadores (caracteres que aparece, entre palavras, como espaço, vírgula, ponto, etc).

Uma forma eficiente de lidar com palavras e separadores é representar o espaço simples de forma implícita no texto comprimido. Se uma palavra é seguida de um espaço, somente a palavra é codificada; caso contrário, a palavra e o separador são codificados separadamente. No momento da decodificação, supõe-se que um espaço simples segue cada palavra, a não ser que o próximo símbolo corresponda a um separador.

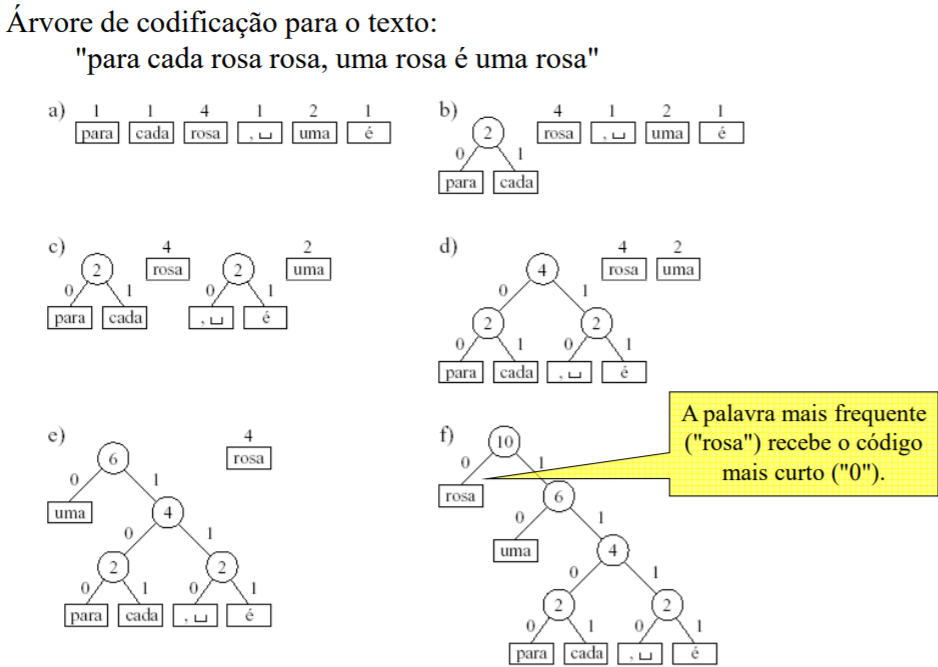
**Árvore de codificação**

O algoritmo de Huffman constrói uma árvore de codificação, partindo-se de baixo para cima.

Inicialmente, há um conjunto de N folhas representando as palavras do vocabulário e suas respectivas frequências;

A cada intercalação, as duas árvores com as menores frequências são combinadas em uma única árvore e a soma de suas frequências é associada ao nó raiz da árvore gerada;

Ao final de (n-1) iterações, obtém-se a árvore de codificação, na qual o código associado a uma palavra é representado pela sequência dos rótulos das arestas da raiz à folha que a representa.

 Ex.:

O método de Huffman produz a árvore de codificação que minimiza o comprimento do arquivo.

Existem várias árvores que produzem a mesma compressão, trocar o filho à esquerda de um nó por um filho a direita leva a uma árvore de codificação alternativa com a mesma razão de compressão.

A escolha preferencial é a árvore canônica. \*Uma árvore de Huffman é canônica quando a altura da subárvore à direita de qualquer nó nunca é menor que a altura da subárvore à esquerda.

A representação do código por meio de uma árvore canônica de codificação facilita a visualização e sugere métodos triviais de codificação e decodificação.

Codificação: a árvore é percorrida emitindo *bits* ao longo de suas arestas;

Decodificação: os *bits* de entrada são usados para selecionar as arestas.

Essa abordagem é ineficiente tanto em termos de espaço quanto em termos de tempo.

**Algoritmo de Moffat e Katajainen**

O algoritmo criado em 1995, baseado na codificação canônica, apresenta comportamento linear em tempo e em espaço.

O algoritmo calcula os comprimentos dos códigos em lugar dos códigos propriamente ditos. \*A compressão atingia é a mesma, independentemente dos códigos utilizados

Após o cálculo dos comprimentos, há uma forma elegante e eficiente para a codificação e a decodificação.

A entrada do algoritmo é um vetor A contendo as frequências das palavras em ordem decrescente.

Para o texto “Para cada rosa rosa, uma rosa é uma rosa”, o vetor A é:



4 – Rosa;

2 - Uma;

1 – Para;

1 – Cada;

1 – **‘,’**;

1 – é.

Durante a execução, são usados vetores logicamente distintos, mas que coexistem no mesmo vetor A.

O algoritmo divide-se em três fases distintas:

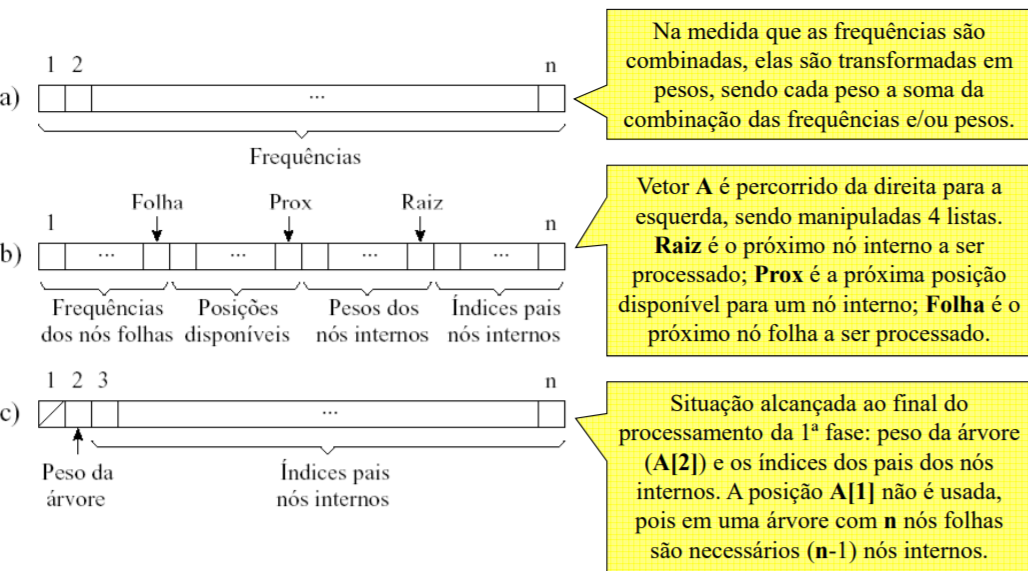
1 – Combinação dos nós;

2 – Determinação das profundidades dos nós internos;

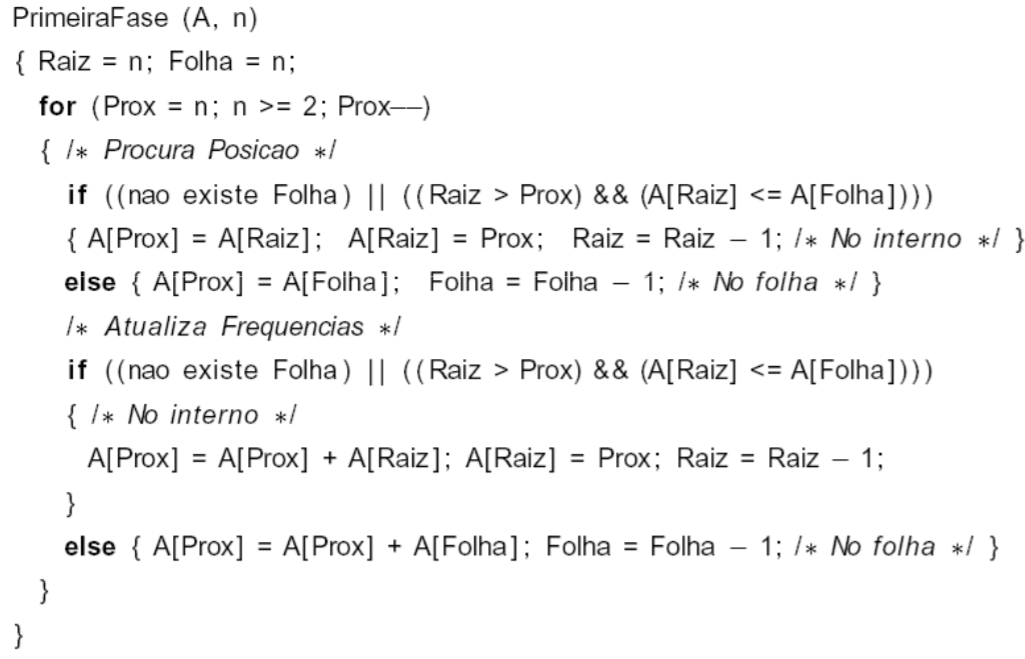
3 – Determinação das profundidades dos nós folhas (comprimento dos códigos).

**Exemplificação.**

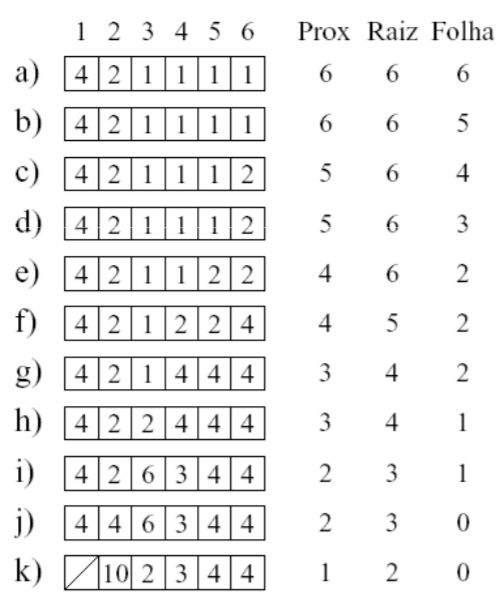
Primeira fase do algoritmo: Combinação dos nós



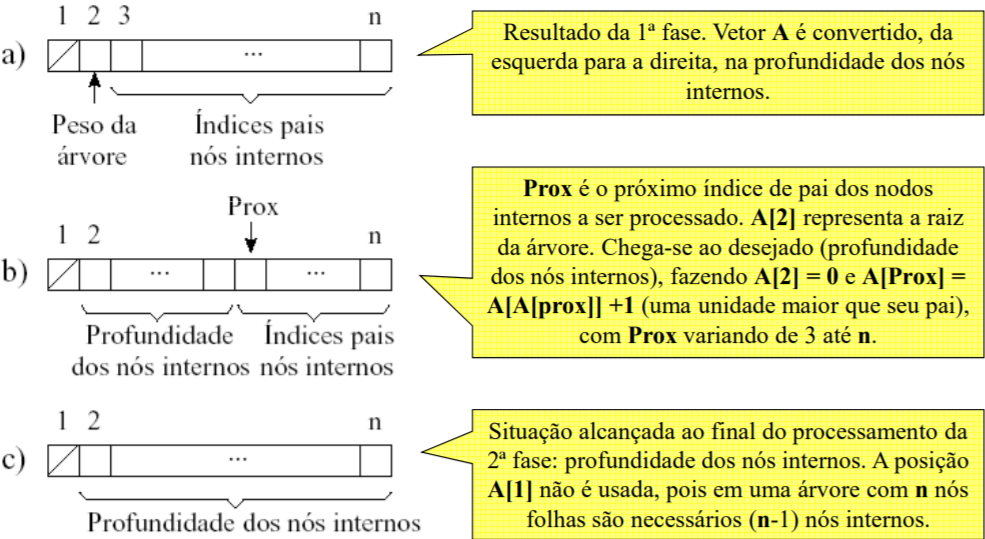
Código:



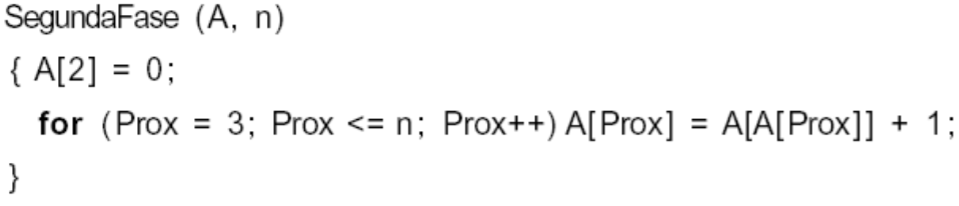
Exemplo da primeira fase do algoritmo:



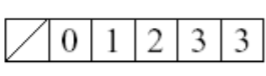
Segunda fase: profundidade dos nós internos



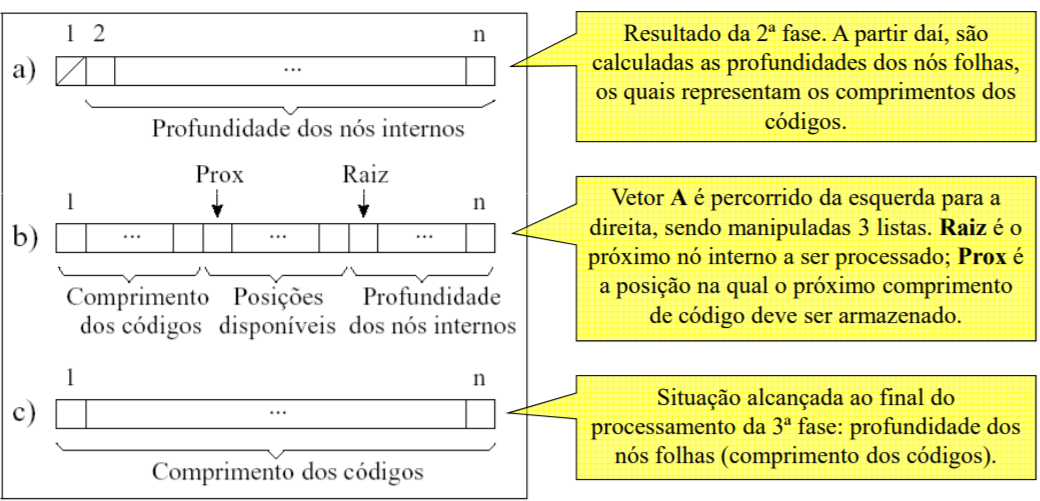
Código:



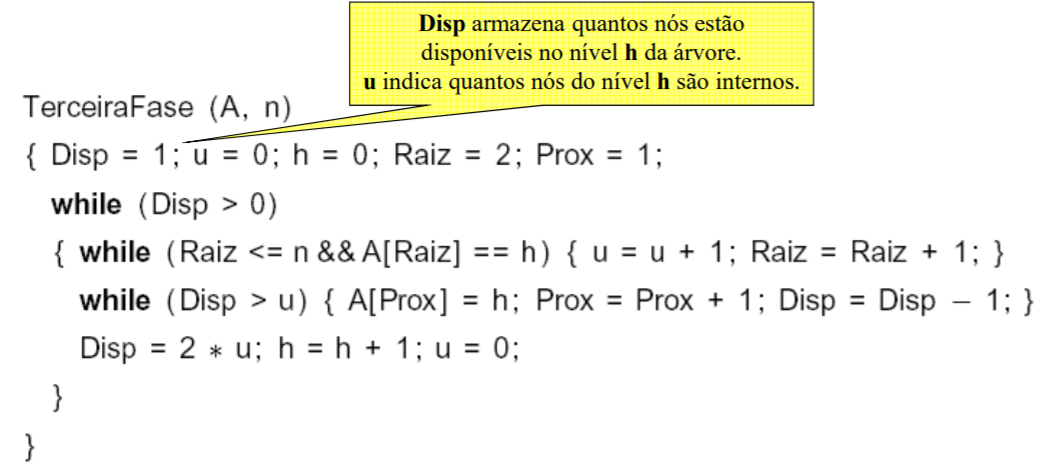
Resultado:



Terceira fase: Profundidade dos nós folhas



Código:



Resultado:



Programa completo para calcular o comprimento dos códigos a partir de um vetor de frequências:

**CalculaCompCodigo(A, n){**

**A = PrimeiraFase (A, n);**

**A = SegundaFase (A, n);**

**A = TerceiraFase (A, n);**

**}**

**Obtenção dos Códigos Canónicos**

As propriedades dos códigos canônicos são:

Os comprimentos dos códigos seguem o algoritmo de Huffman;

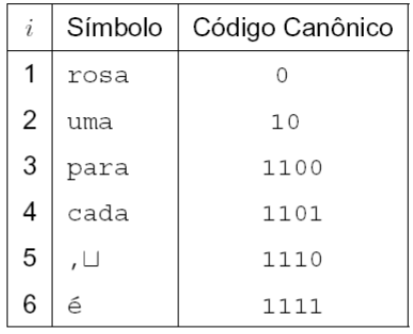
Códigos de mesmo comprimento são inteiros consecutivos.

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

O primeiro código é composto apenas por zeros;

Para os demais, adiciona-se 1 ao código anterior e fase um deslocamento à esquerda para obter-se o comprimento adequado quando necessário;

Ex.:

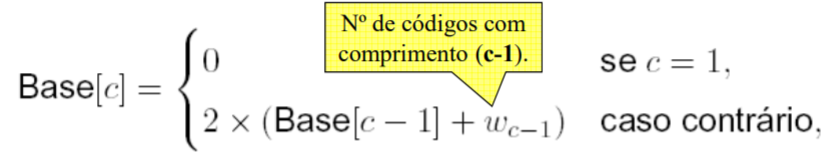


**Codificação e Decodificação**

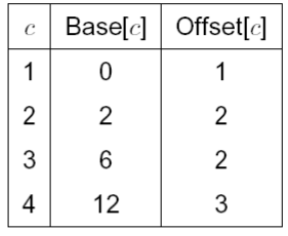
Os algoritmos são baseados no fato: Códigos de mesmo comprimento são inteiros consecutivos.

Os algoritmos usam dois vetores com MaxCompCod (o comprimento do maior código) elementos:

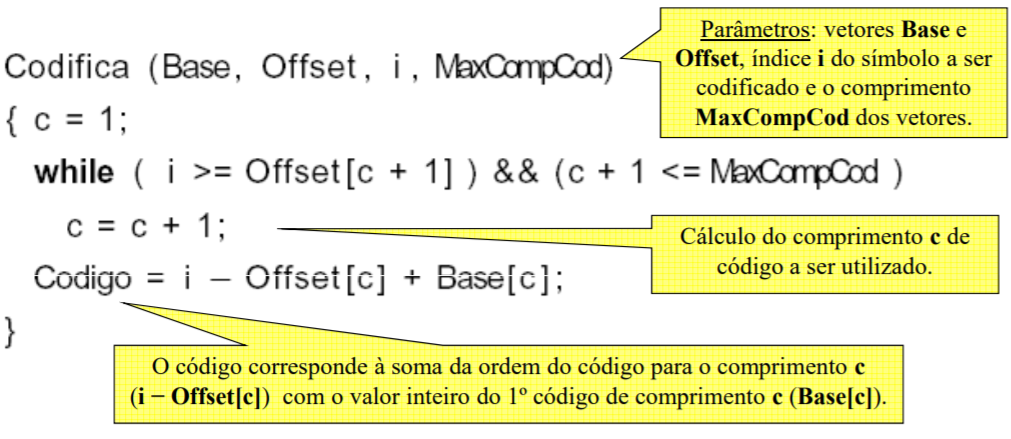
*Base*: Indica, para um dado comprimento *c*, o valor inteiro do 1º código com tal comprimento;



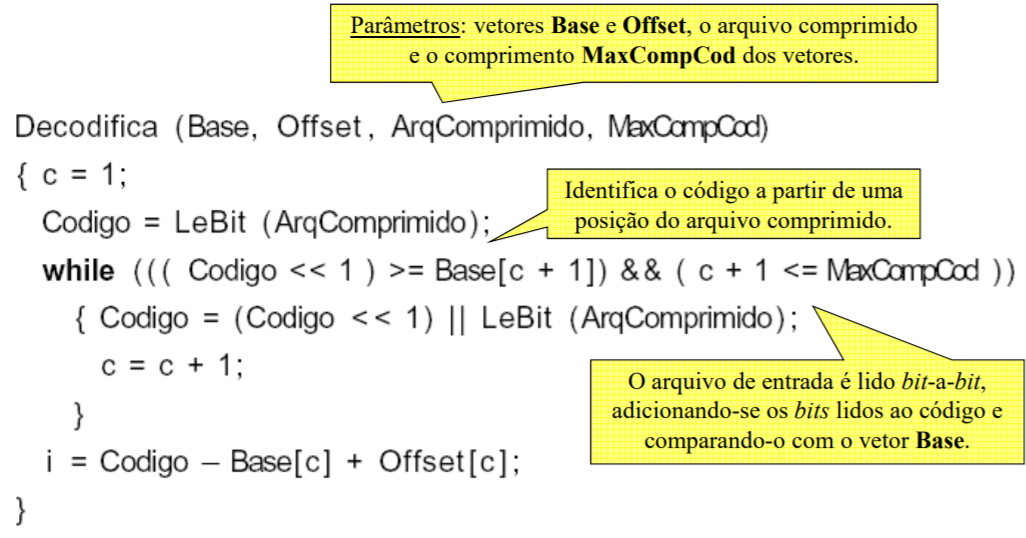
*Offse*t: indica, para um dado comprimento *c*, o índice no vocabulário da 1ª palavra de tal comprimento.



Código:



Para i = 4 (“cada”), calcula-se que seu código possui comprimento 4 e verifica-se que é o 2º código de tal comprimento. Assim, seu código é 13(4 – *offset* [4] + *base* [4]): 1101



Código de compressão:

1ª Etapa: o arquivo texto é percorrido e o vocabulário é gerado juntamente com a frequência de cada palavra.

Uma tabela *hash* com tratamento de colisão é utilizada para que as operações de inserção e pesquisa no vetor de vocabulário sejam realizadas com custo O (1).

2ª Etapa:

O vetor vocabulário é ordenado pelas frequências de suas palavras;

Calcula-se o comprimento dos códigos (algoritmo de Moffat e Katajainen);

Os vetores *Base, Offset* e *Vocabulário* são construídos e gravados no início do arquivo comprimido;

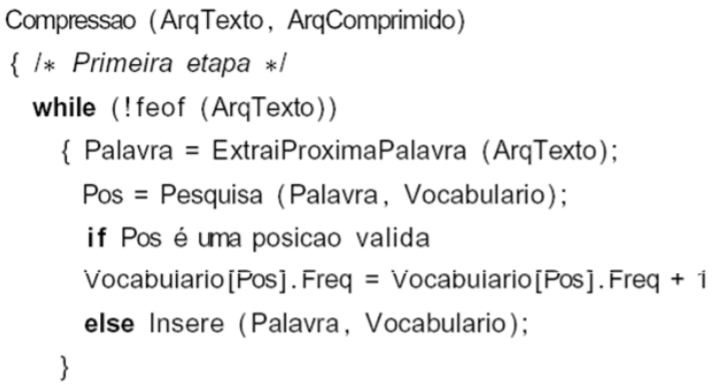
A tabela *hash* é reconstruída a partir do vocabulário no disco, como preparação para a 3ª Etapa.

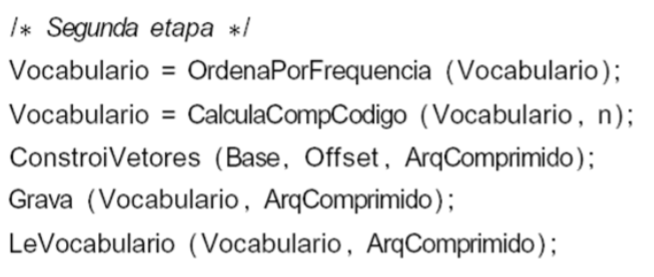
3ª Etapa:

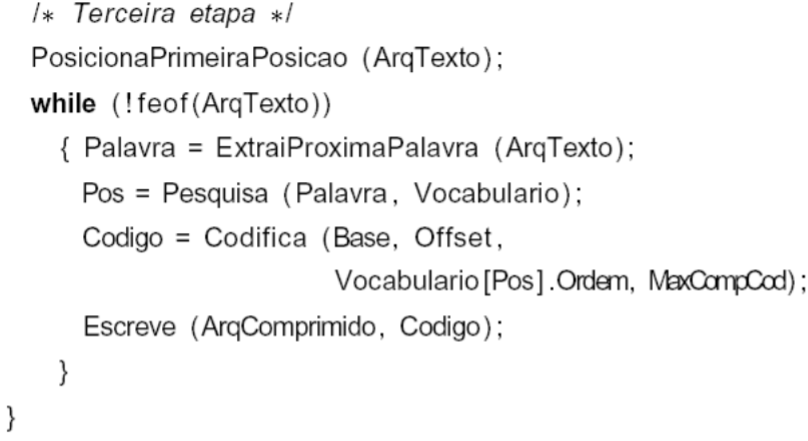
O arquivo texto é novamente percorrido;

As palavras são extraídas e codificadas;

Os códigos correspondentes são gravados no arquivo comprimido.







Código de descompressão:

O processo de descompressão é mais simples do que o de compressão:

Leitura dos vetores *Base, Offset* e *Vocabulário* gravados no início do arquivo comprimido;

Leitura dos códigos do arquivo comprimido, descodificando-se e gravando as palavras correspondentes no arquivo texto.

