Trabalho 1: Compactação e descompactação de arquivos texto

turma A

Descrição:

Um primeiro algoritmo de compactação poderia ser implementado da seguinte forma:

Utilizando a tabela ASC para os caracteres maiúsculos e os caracteres “.”, “,”, “;”, “espaço”, “?” e “! (aqui representado como EOF)”, trocar caracter a caracter o código ASC estendido, de 8 bits para o correspondente em 5 bits fixos (varia de ‘00000’ a ‘11111’) de acordo com uma tabela onde:

O caractere “A” corresponderia ao código de 5 bits ‘00000’,
O caractere “B” corresponderia ao código de 5 bits ‘00001’,
O caractere “C” corresponderia ao código de 5 bits ‘00010’,
e assim sucessivamente, ...
O caractere “?” corresponderia ao código de 5 bits ‘11110’,
O caractere “fim de arquivo” corresponderia ao código de 5 bits ‘11111’.

Desta forma, este algoritmo de compactação iria ganhar 3 bits para cada caractere de 8 bits, onde um arquivo original de 1K bytes, seria comprimido para 640 bytes.

Assim, o texto “ABRACADABRA!” teria a representação em bits como:
01000001010000100101001001000001010000110100000101000100010000010100001000101
A B R A C A D A B
0010010000001001000 01 - totalizando 12 x 8 = 96 bits
R A !

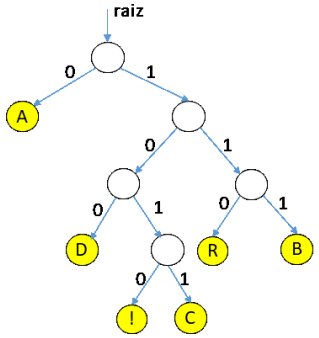
E o arquivo comprimido teria a seguinte representação em bits:
00000000011000100000000100000000110000000010001000001111 1-totalizando 60 bits
A B R A C A D A B R A !

Um segundo algoritmo, mais eficiente na compactação, foi proposto por Huffman. Este algoritmo permite comprimir informação, sem perda, também através da recodificação dos seus bytes. O algoritmo consiste de uma série de passos nos quais é possível construir a codificação (fluxo de bits comprimido) que substitui o fluxo de bits original. O princípio do algoritmo é de gerar **códigos mais curtos para caracteres mais frequentes e códigos maiores para os caracteres menos frequentes** (pesquisar sobre árvores Trie ou árvore de prefixos).

De forma geral, o algoritmo consiste em construir um histograma dos caracteres utilizados no texto em ordem de frequência. Vamos supor que o texto seja “ABRACADABRA!” e o histograma indique que a ordem de frequência no texto seja o caractere “A” o mais frequente, em seguida o caractere “B”, depois o “R”, em seguida o “C”, depois o “D” e finalmente o “! (indicando o EOF)”. Assim, o histograma seria:
Caractere “A” – ocorre 5 vezes
Caractere “R” – ocorre 2 vezes
Caractere “B” – ocorre 2 vezes
Caractere “D” – ocorre 1 vez
Caractere “!” – ocorre 1 vez
Caractere “C” – ocorre 1 vez

Huffman propôs usar uma árvore de prefixos, onde os prefixos da representação em bits são únicos.

Exemplo da árvore de prefixos e respectiva representação dos caracteres em bits:



A - 0
D -100
R -110
B -111
! - 1010
C -1011

Desta forma, a representação interna do arquivo comprimido fica:

0111110010110100011111001010 - totalizando 28 bits
A B RA CA DA B RA !

O trabalho de EDA consiste em **implementar um compactador e um descompactador de arquivos de texto com base no algoritmo de Huffman**.

Para teste do algoritmo, pode ser usado um arquivo contendo a seguinte frase: “as estruturas de dados sao fundamentais para a organizacao e a manipulacao eficiente das informacoes”. Exiba a tabela obtida. Esta saída deve fazer parte do relatório.

Para compactar e descompacatar o arquivo implemente uma tabela contendo o caractere e o número de ocorrências do caractere no texto e outra contendo o caractere, o código binário correspondente e o seu tamanho em bits.

Siga as orientações de entrega de laboratórios descrita no EAD.

Data de Entrega: até 9:00 hs do dia 13/05/2024.

Atenção: Trabalhos entregues com atraso sofrerão perda de 10% de sua nota por cada dia após o prazo de entrega.

O arquivo a ser compactado e descompactado para verificação da correção do seu programa está no EAD.