ESTRUTURAS DE DADOS II

MSC. DANIELE CARVALHO OLIVEIRA

DOUTORANDA EM CIÊNCIA DA COMPUTAÇÃO - USP

MESTRE EM CIÊNCIA DA COMPUTAÇÃO – UFU

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO - UFJF

2

COMPRESSÃO DE DADOS

3 COMPRESSÃO DE DADOS

- Objetivos
  - Reduzir espaço de armazenagem
  - Reduzir tempo de transmissão
- Muito importante
  - Informação (e dados) tende a crescer de forma exponencial
- Exemplos:
  - Compressão de arquivos em geral
    - ZIP, GZIP, BZIP, BOA.
  - Sistemas de arquivos: NTFS.
  - Multimedia.
    - Imagens: GIF, JPEG
    - Som: MP3.
    - Vídeo: MPEG, DivX™, HDTV

4 COMPRESSÃO X COMPACTAÇÃO

- Compressão
  - Mudança na representação de algum dado para reduzir seu tamanho
- Compactação
  - União de dados que não estão unidos

- Exemplo:
  - Compressão do HD: reduzir o tamanho dos arquivos
  - Compactação do HD: juntar partes disjuntas (desfragmentar)

5 SISTEMAS DE RECUPERAÇÃO DE INFORMAÇÃO

- Métodos recentes de compressão têm permitido:
  - Pesquisar diretamente o texto comprimido mais rapidamente do que o texto original.
  - Obter maior compressão em relação a métodos tradicionais, gerando maior economia de espaço.
  - Acessar diretamente qualquer parte do texto comprimido sem necessidade de descomprimir todo o texto desde o início

6 CODIFICAÇÃO E DECODIFICAÇÃO

- Seja M a mensagem que desejamos armazenar/transmitir
  - Codificação: Gerar uma representação "comprimida" C(M) que, espera-se, utilize menos bits
  - Decodificação: Reconstrução da mensagem original ou alguma aproximação M'
- Razão de Compressão: bits de C(M) / bits de M
- Compressão sem perda: M = M'
  - Texto, programas fonte, executáveis etc.
  - RC: 50-75% ou menos
- Compressão com perda: M ~ M'
  - Imagens, som, vídeo
  - Uma ideia é descartar informação não perceptível pelos sentidos
  - RC: 10% ou mais, dependendo do fator de qualidade

7 CODIFICAÇÃO POR SEQÜÊNCIAS REPETIDAS

- Conhecida como Run-length encoding (RLE).
- Ideia é explorar longas sequências de caracteres repetidos
  - Inicialmente, escolhe-se um valor de byte especial, que não ocorre no arquivo. Esse byte será usado para indicar que, a seguir, vem um código RLE.
  - A seguir, executa-se o algoritmo RLE, como segue:
    - leia a sequência, copiando sequencialmente os valores para um arquivo destino, exceto quando o mesmo valor ocorre mais de uma vez sucessivamente.
    - quando dois ou mais valores iguais ocorrem em sequência, substitua-os pelos seguintes 3 bytes:
      - o byte especial que indica RLE
      - o valor que se repete
      - o número de vezes que o valor repete

8 EXEMPLO COMPRESSÃO RLE

- Como ficaria a sequência abaixo, se o byte especial é ff ?
  - 22 23 24 24 24 24 24 24 24 25 26 26 26 26 26 26 25
    - Resposta: ?????????????

- Outro Exemplo:
  - AAAABBBAABBBBBCCCCCCCCDABCBAAABBBBCCCD
    - Resposta: ?????????????

- Problema: Como representar o número de repetições?
  - Alguns esquemas + ou – complicados
  - Pode levar à inflação de um arquivo se as sequências são curtas

9 EXEMPLO COMPRESSÃO RLE

- Como ficaria a sequência abaixo, se o byte especial é ff ?
  - 22 23 24 24 24 24 24 24 24 25 26 26 26 26 26 26 25
    - Resposta: 22 23 ff 24 07 25 ff 26 06 25

- Outro Exemplo:
  - AAAABBBAABBBBBCCCCCCCCDABCBAAABBBBCCCD
    - Resposta: 4A3BAA5B8CDABCB3A4B3CD

- Problema: Como representar o número de repetições?
  - Alguns esquemas + ou – complicados
  - Pode levar à inflação de um arquivo se as sequências são curtas

10 CODIFICAÇÃO DE HUFFMAN

- É um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser compactado para determinar códigos de tamanho variável para cada símbolo.

- Considere a string "bom esse bombom". Usando ASCII teríamos:

Char	ASCII	Binário
b	98	0110 0010
o	111	0110 1111
m	109	0110 1101
e	101	0110 0101
s	115	0111 0011
Espaço	32	0010 0000

- A string "bom esse bombom" seria escrita numericamente assim:
  - 98 111 109 32 101 115 115 101 32 98 111 109 98 111 109
- Em binário, seria assim:
  - 0110 0010 0110 1111 0110 1101 0010 0000 0110 0101 0111 0011 0111 0011 0110 0101 0010 0000 0110 0010 0110 1111 0110 1101 0110 0010 0110 1111 0110 1101

11 CODIFICAÇÃO DE HUFFMAN

- Procurando comprimir este texto, podemos reduzir a codificação para:

Char		Binário
b	0	000
o	1	001
m	2	010
e	3	011
s	4	100
Espaço	5	101

- A string seria numericamente escrita assim:
  - 0 1 2 5 3 4 4 3 5 0 1 2 0 1 2
- e em binário:
  - 000 001 010 101 011 100 100 011 101 000 001 010 000 001 010

12 CODIFICAÇÃO DE HUFFMAN

- Usando 3-bits por caractere, a string "bom esse bombom" usa um total de 48 bits ao invés de 128.
- A "economia" de bits poderia ser ainda maior se usarmos menos de 3 bits para codificar caracteres que aparecem mais vezes (b, o, m) e mais bits para caracteres que aparecem menos vezes (e, s).

- Essa é a ideia básica da codificação de Huffman: usar menor número de bits para representar caracteres com maior frequência.
- Essa técnica pode ser implementada usando uma árvore binária que armazena caracteres nas folhas, e cujos caminhos da raiz até as folhas provêm a sequência de bits que são usados para codificar os respectivos caracteres.

ÁRVORE DE CODIFICAÇÃO

- Usando uma árvore binária, podemos representar todos os caracteres como folhas de uma árvore completa
  - Arcos da esquerda são numerados com 0 enquanto arcos da direita com 1.
  - O código para qualquer caractere/folha é obtido seguindo o caminho da raiz à folha e concatenando os 0's e 1's.

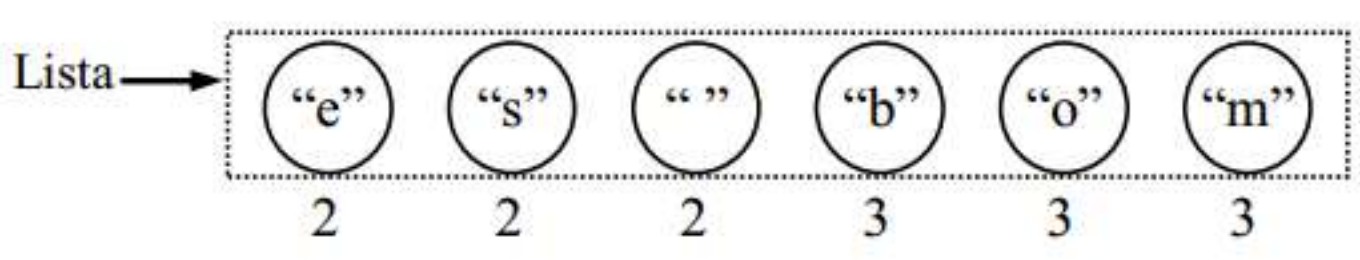
- Para maximar a compressão, precisamos encontrar uma árvore ótima, que apresente um número mínimo de codificação por caractere.
- O algoritmo para encontrar essa árvore ótima é chamado de codificação de Huffman

14 ALGORITMO DE HUFFMAN

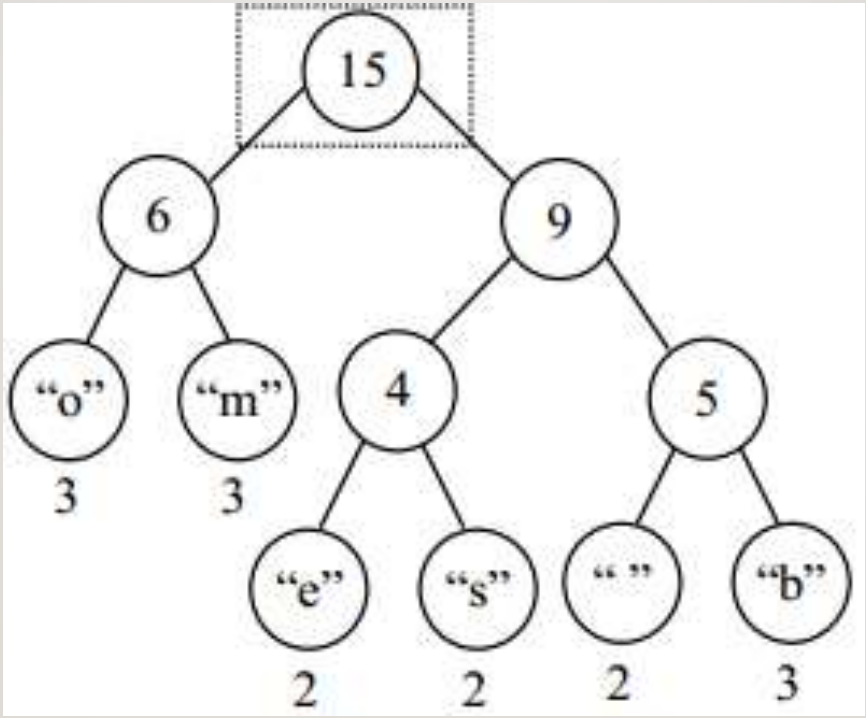
- Assuma que cada caractere em um texto está associado a um peso, que é definido pelo número de vezes que o caractere aparece em um arquivo.
  - Na string "bom esse bombom", os caracteres "b", "o" e "m" têm peso 3, enquanto os caracteres "e", "s" e espaço têm peso 2.
  - Para usar o algoritmo de Huffman, é necessário calcular esses pesos

15 ALGORITMO DE HUFFMAN

- Aplique o Algoritmo de Huffman:



16 ALGORITMO DE HUFFMAN



- A codificação de caracteres induzida pela última árvore é a seguinte (considerando 0 para arcos da esquerda e 1 para arcos da direita):

Char	Binário
b	111
o	00
m	01
e	100
s	101
Espaço	110

- Usando essa tabela de codificação, a string "bom esse bombom" ficaria assim:
  - 111 00 01 110 100 101 101 100 110 111 00 01 111 00 01

- Desta forma usamos 39 bits para codificar a string "bom esse bombom", ao invés de 48 bits como mostramos usando a codificação de 3-bits.

18 ALGORITMO DE HUFFMAN

- O algoritmo de Huffman assume que uma árvore será construída a partir de um grupo de árvores.
- Inicialmente essas árvores têm um único nó com um caractere e o peso desse caractere.
- À cada iteração do algoritmo, duas árvores são juntadas criando uma nova árvore. Isso faz com que o número de árvores diminua a cada passo.
- Este é o algoritmo:
  - Comece com uma lista de árvores. Inicialmente, todas as árvores são compostas de um nó apenas, com o peso da árvore igual ao peso do caractere do nó. Caracteres que ocorrerem mais frequentemente têm o maior peso. Caracteres que ocorrerem menos frequentemente têm o menor peso. Ordene a lista de árvores de forma crescente, fazendo com que o nó raiz da primeira árvore seja o caractere de menor peso e o nó raiz da última árvore seja o caractere de maior peso.
  - Repita os passos a seguir até que sobre apenas uma única árvore:
    - Pegue as duas primeiras árvores da lista e as chame de T1 e T2. Crie uma nova árvore Tr cuja raiz tenha o peso igual à soma dos pesos de T1 e T2 e cuja subárvore esquerda seja T1 e subárvore direita seja T2.
    - Exclua T1 e T2 da lista (mantendo T1 e T2 na memória) e inclua Tr na lista, de maneira que a lista seja mantida ordenada.
    - A árvore final será a árvore ótima de codificação.

19 IMPLEMENTAÇÃO

- Há duas partes distintas na implementação: um programa que faz a compactação e um programa que faz a descompactação.
- Para compactar um arquivo, é necessário a tabela de compressão. Como vimos, esta tabela é construída com uma árvore binária de compactação.
- Assumindo que um número fixo de bits é escrito em um arquivo, um arquivo compactado é criado seguindo os seguintes passos:
  - Construa a tabela de codificação;
  - Leia o arquivo a ser compactado e processe um caractere de cada vez. Para processar o caractere e chegar à sequencia de bits que representa o caractere, use a tabela de caracteres que foi criada no passo anterior. Escreva essa sequencia de bits no arquivo compactado.

IMPLEMENTAÇÃO

- O arquivo compactado deve conter as informações necessárias para se chegar corretamente ao arquivo original a partir do arquivo comprimido.
- As seguintes informações são necessárias:
  - um cabeçalho no arquivo compactado que deve conter a árvore de codificação;
  - alguma marca de final de arquivo para indicar que a sequencia de bits chegou ao fim.
- O programa descompressor deve criar a árvore de codificação que está no cabeçalho do arquivo, ler os bits e navegar na árvore até encontrar os nós folhas correspondentes àquela sequencia de bits lidos.
  - O custo deste algoritmo de busca é proporcional à altura da árvore.

CABEÇALHO DO ARQUIVO COMPRIMIDO

- A árvore de codificação pode ser armazenada no início do arquivo compactado.
- Existem várias maneiras de armazenar a árvore.
  - Por exemplo, use o percurso em pré-ordem para escrever cada nó visitado pela travessia.
  - Nós folha devem ser diferenciados de nós não-folhas.
  - Uma maneira de fazer a diferenciação é escrever um único bit para cada nó, por exemplo 1 para nós folhas e 0 para nós não-folha. Para nós-folha, é também necessário escrever o caractere armazenado. Para nós não-folha é necessário apenas o bit indicando de que se trata de um nó não-folha.

CARACTERE DE FINAL DA SEQUENCIA DE BITS

- É preciso introduzir um pseudo-caractere de final de arquivo, introduzido explicitamente pelo programador ao final da sequencia de bits válidos.
- Este caractere também deve entrar na árvore/tabela de codificação para que seja corretamente descomprimido pelo programa descompressor.
- Quando um arquivo compactado é escrito, os últimos bits escritos devem ser os bits que representam o pseudo-caractere.

Trabalho

- Implemente o Algoritmo de Huffman

Extra

- URI
  - 1673

FIM DA AULA 7

Próxima aula:

Recursão