Compressão Sem Perdas: Codificações Huffman e Aritmética

Adelar da Silva Queiróz
Marcelo Teixeira
Thiago da Silva Sodré

Compressão Sem Perdas (Lossless Data Compression)

Refere-se a métodos de compressão de dados aplicados através de algoritmos em que a informação obtida após a descompressão é idêntica à informação original (antes de ser comprimida);
Transferência de texto, arquivos binários, etc.

Técnicas de compressão sem perdas

- Modelos Estatísticos:
  - Huffman
  - Shannon-Fano
  - Codificação Aritmética
- Técnicas baseadas em dicionários:
  - Lempel-Ziv (LZ 77)
  - Lempel-Ziv (LZ 78)
  - Lempel-Ziv-Welch

Codificação Huffman

- Idéia é usar caracteres (símbolos) com número variável de bits
  - Dada uma mensagem, encontrar a melhor (menor) codificação para cada caractere
- Construir a Árvore de Prefixos de Huffman
  - Maior freqüência -> Códigos pequenos
  - Menor freqüência -> Códigos grandes

Características

- Árvore de Prefixos de Huffman
- Folhas = caracteres
- Aresta esquerda = 0
- Aresta direita = 1
- Complexidade $O(M + N \log N)$
  - M é o comprimento da mensagem
  - N é o número de caracteres distintos

Algoritmo Guloso de Huffman

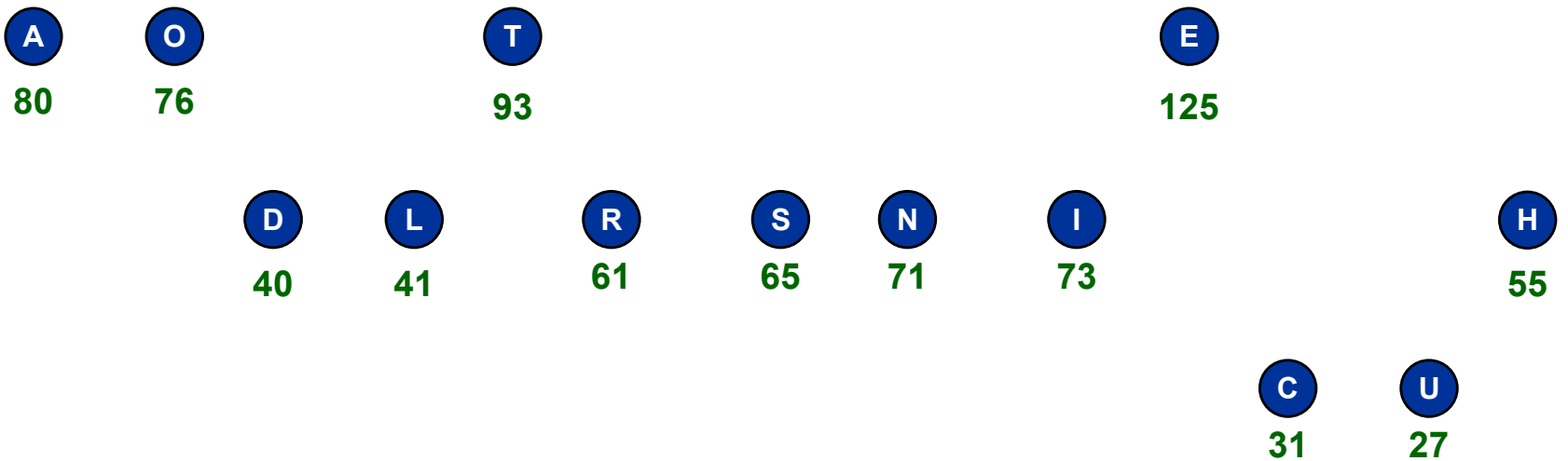
- Tabular freqüências dos símbolos
- Montar uma floresta de árvores unitárias com todos os símbolos e suas freqüências
- Repetir
  - Localizar os dois elementos com menor freqüência F_i e F_j
  - Uní-los numa única árvore com freqüência $F_i + F_j$

Exemplo de Codificação de Huffman

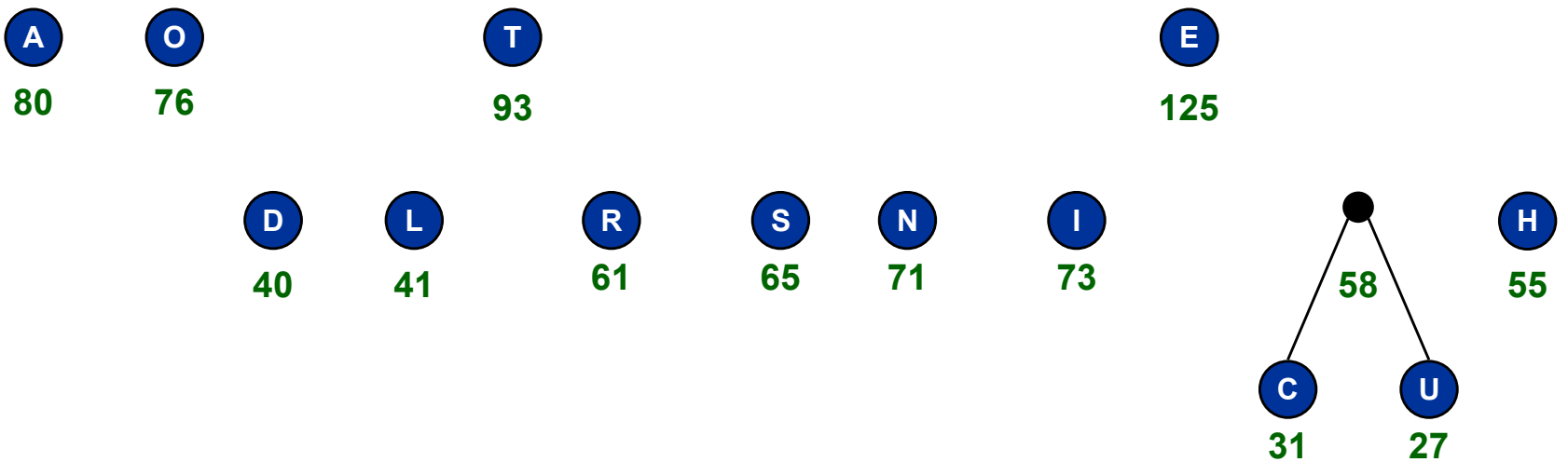
- Freqüências dos caracteres

Char	Freq
E	125
T	93
A	80
O	76
I	72
N	71
S	65
R	61
H	55
L	41
D	40
C	31
U	27

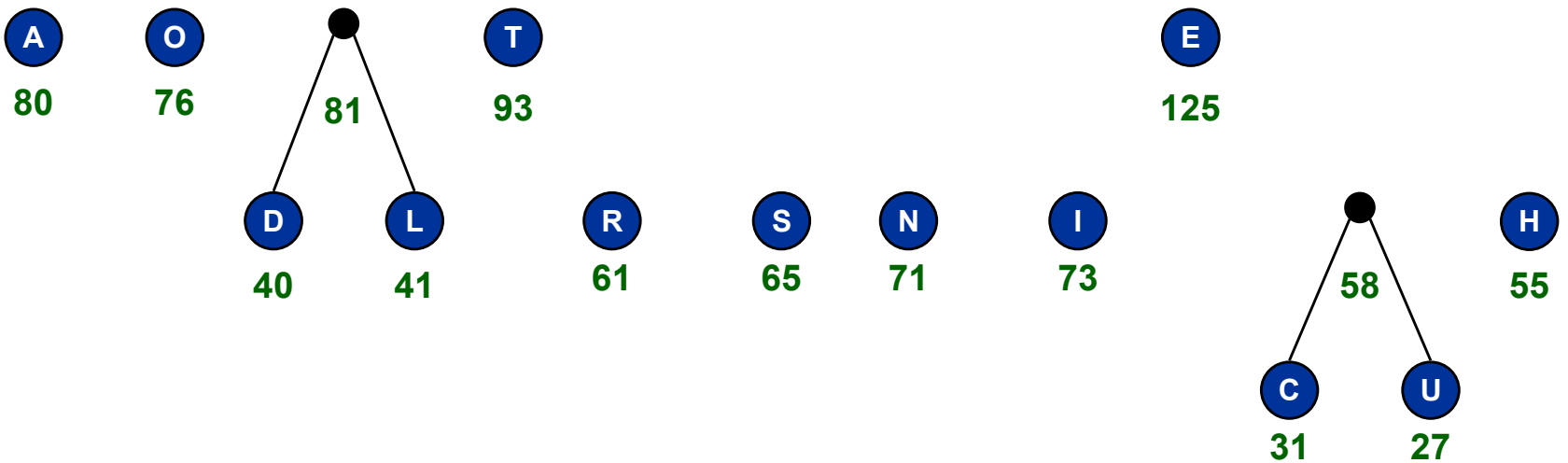
Exemplo de Codificação de Huffman



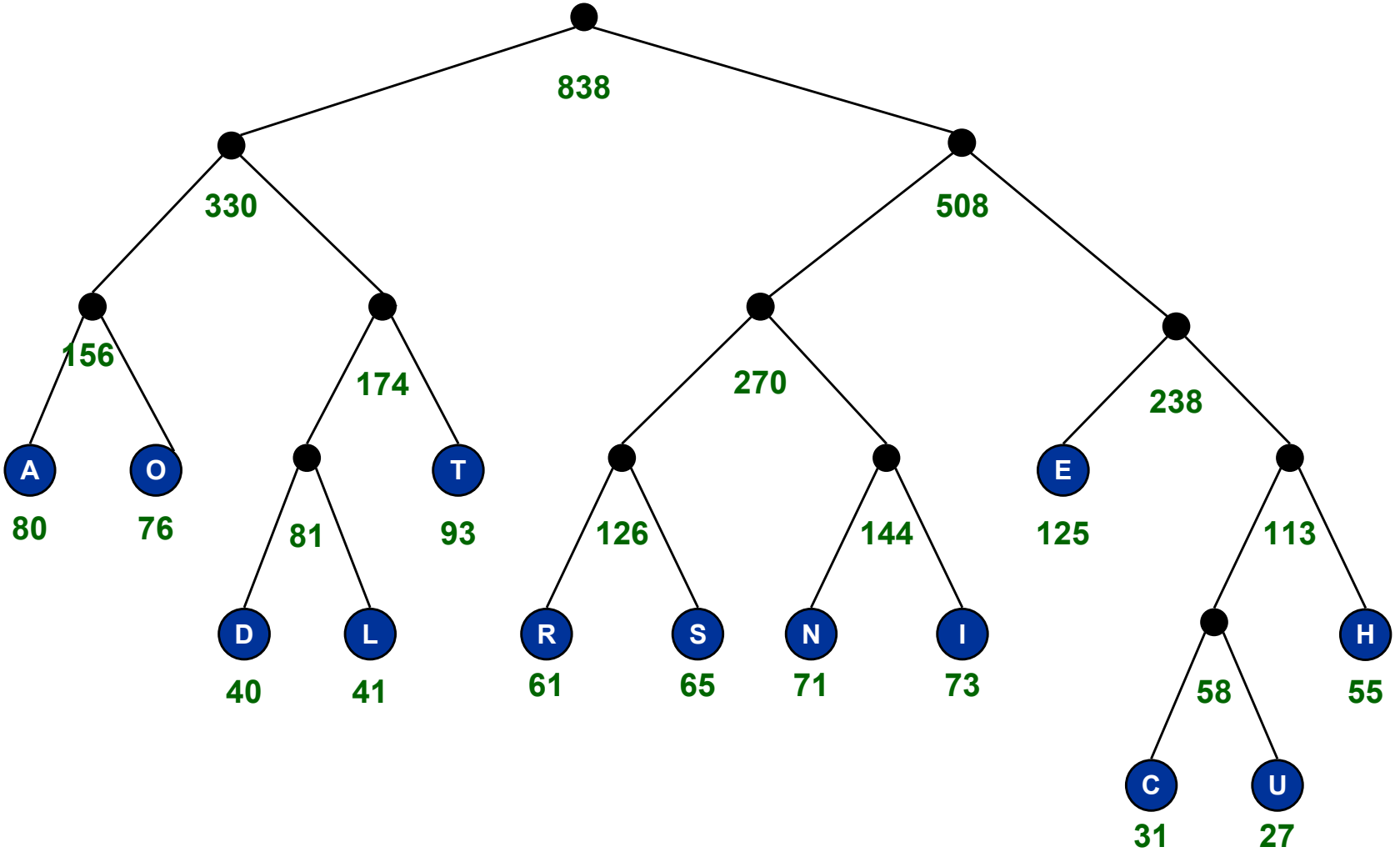
Exemplo de Codificação de Huffman



Exemplo de Codificação de Huffman



Exemplo de Codificação de Huffman



Exemplo de Codificação de Huffman

Char	Freq	Fixo	Huff
E	125	0000	110
T	93	0001	000
A	80	0010	000
O	76	0011	011
I	73	0100	1011
N	71	0101	1010
S	65	0110	1001
R	61	0111	1000
H	55	1000	1111
L	41	1001	0101
D	40	1010	0100
C	31	1011	11100
U	27	1100	11101
	838	4.00	3.62

Compressão de imagens

- O arquivo comprimido deve conter toda a seqüência de códigos junto com a árvore;
- Pode se obter as freqüências através do cálculo do histograma da imagem;
- O método não utiliza a relação existente entre os pixels vizinhos abordada em outras técnicas de compressão;

Vantagem

- Redução do código para armazenar caracteres de maior freqüência
- A principal vantagem:
  - Processo de descodificação torna-se bastante simples
  - Percorrer a árvore: 0 esquerda, 1 direita;

Dificuldades e Problemas

- Construir a árvore
- Construir freqüência dos caracteres ou pixels
- Não é ótimo
  - Número de bits por pixel é inteiro
  - Codificação aritmética: permite número “fracionário”de bits por pixel

Codificação Aritmética

Conceitos

- Taxa de codificação: é a média dos números de bits usados para representar um símbolo de uma fonte.
- Entropia: Para uma dada probabilidade, é a menor taxa no qual a fonte pode ser codificada.

Definição

- Método para compressão de dados, não baseado em tabelas de símbolos;
- O codificador aritmético elimina a associação entre símbolos individuais e palavras (códigos de comprimento inteiro) e, com isto, é capaz de praticamente igualar a entropia da fonte em todos os casos;
- A partir de um modelo estatístico, constrói-se uma tabela onde são listadas as probabilidades do próximo símbolo lido ser cada um dos possíveis símbolos;
- Em geral esta probabilidade é simplesmente a contagem de todas as ocorrências do símbolo no arquivo, dividida pelo tamanho do :

Definição (continuação)

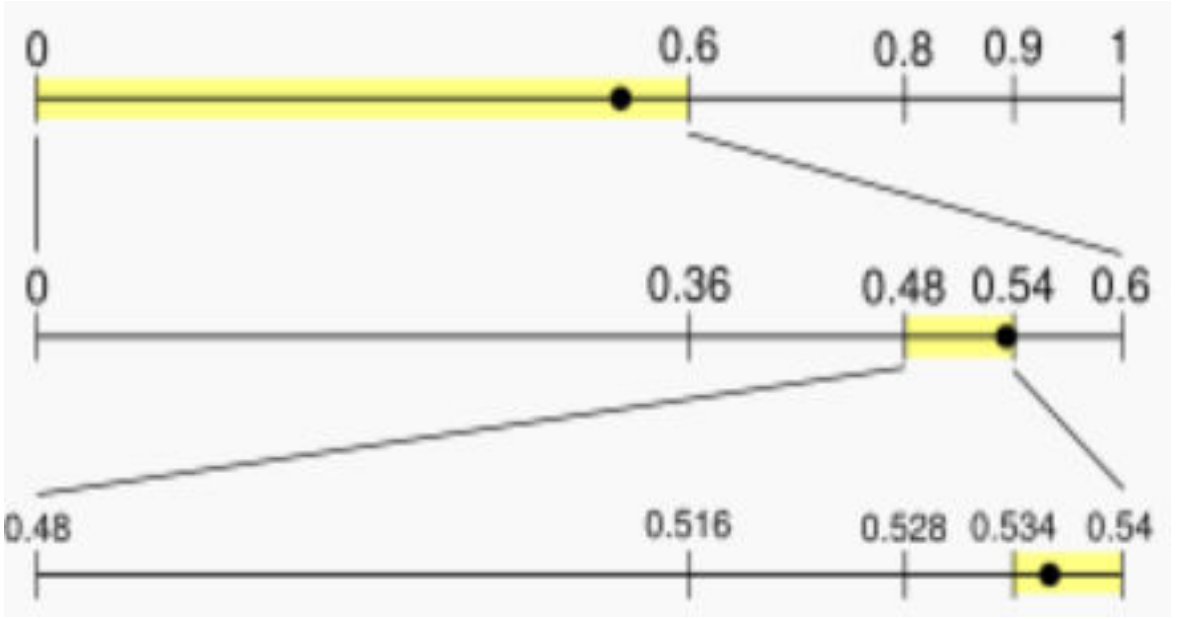
$$P(\sigma) = \frac{n_\sigma}{N}$$

- onde P(σ) é a probabilidade de ocorrência do símbolo σ,
- nσ é o número de ocorrências desse símbolo,
- e N é o tamanho do arquivo.

Codificação Aritmética

- O algoritmo de codificação aritmética consiste em representar a probabilidade da ocorrência de cada caractere de acordo com intervalos cumulativos;
- Parte-se do intervalo [0,1[e nele identifica-se o sub-intervalo ao qual corresponde o primeiro símbolo lido do arquivo;
- Para cada símbolo subsequente, subdivide-se o intervalo atual em sub-intervalos proporcionais às probabilidades da tabela de intervalos, e encontra-se novamente o intervalo que corresponde ao próximo símbolo;
- No final do processo, temos um intervalo que corresponde à probabilidade da ocorrência de todos os símbolos na ordem correta.

Codificação Aritmética



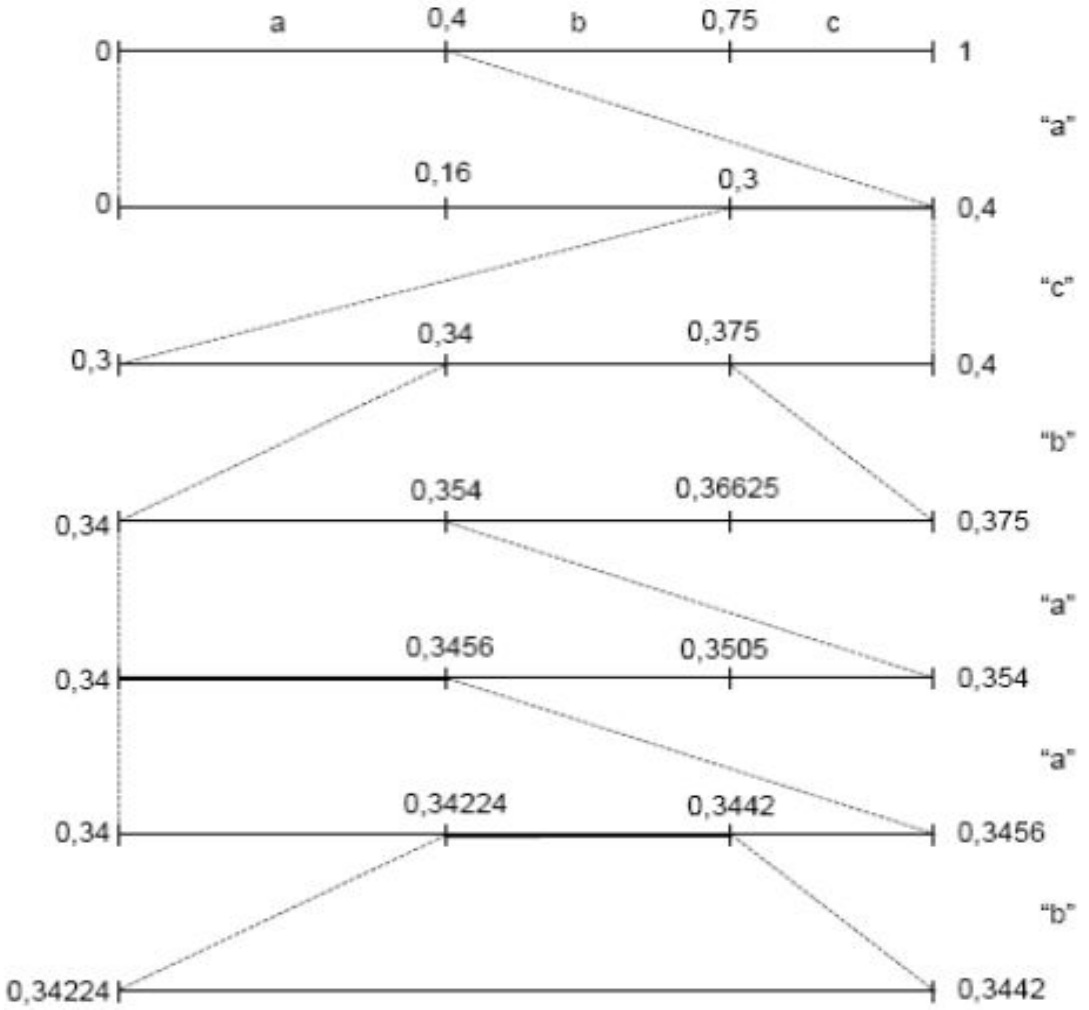
Codificação Aritmética

- A saída do algoritmo é um valor que esteja contido no intervalo [0,1[e possa ser representado com o menor número possível de dígitos;
- Na prática não se tenta encontrar o menor número de dígitos, mas apenas um número razoável de dígitos.

Codificação Aritmética

1. Cria-se um intervalo iniciado com [L,H[o valor [0,1[;
2. Para cada elemento da mensagem:
   - Particiona-se o intervalo corrente em sub-intervalos, um para cada letra do alfabeto. O tamanho do sub-intervalo associado a uma dada letra é proporcional à probabilidade de que essa letra seja o próximo elemento da mensagem, de acordo com o modelo assumido.
   - O sub-intervalo correspondente à letra que é realmente o próximo elemento é seleccionado como o novo intervalo corrente.
3. Codifica-se a mensagem com o menor número de bits necessário para distinguir o intervalo corrente final de todos os outros possíveis intervalos correntes finais.

Codificação Aritmética



P(a) = 0,4
P(b) = 0,35
P(c) = 0,25

Sequência a codificar:
acbaab

O código deve ser um nº real incluído no intervalo
[0.34224, 0,3442[

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

Se nos basearmos diretamente na definição da codificação aritmética iremos “esbarrar” em dois problemas práticos:

- Nenhuma codificação é produzida antes que toda a mensagem tenha sido processada;
- O cálculo dos limites do intervalo corrente para mensagens genéricas exige aritmética de altíssima precisão;

A solução para o problema é relativamente simples.
Um codificador aritmético prático usa apenas a aritmética inteira para "simular" a aritmética de números reais.

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

Definem-se dois valores designados de high e low que representam o intervalo atual.

Inicialmente, o intervalo é entre [0,1[. Estamos a trabalhar apenas com inteiros, de precisão finita.

Então vamos considerar que high e low são apenas os primeiros dígitos após da vírgula no nosso intervalo. Sabemos também que 0,999... é equivalente a 1. Então podemos representar (considerando base decimal e precisão de 4 dígitos):

low (valor inferior) = 0000
high (valor superior) = 9999

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

Representando o nosso intervalo inicial. Para cada caractere lido, devemos estreitar esse intervalo proporcionalmente à probabilidade do caractere.

Assim teremos para cada passo:

```
new_low = low + (high-low+1) * prob_inicial
new_high = low + (high-low+1) * prob_final - 1
```

new_low e new_high são os novos valores para low e high prob_inicial e prob_final são respectivamente o início e o fim do intervalo das probabilidades cumulativas da ocorrência do caractere.

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

A cada caractere lido torna a aplicar-se essa fórmula até que tenham sido lidos todos os caracteres.

Entretanto isto ainda não resolve o problema da precisão finita do nosso cálculo: caso o resultado desejado tenha mais de 4 dígitos depois da vírgula, não seremos capazes de calcular esse valor.

O passo seguinte soluciona os dois problemas apresentados inicialmente neste ponto:

- Caso o primeiro dígito (mais a esquerda) dos dois valores, low e high venha a tornar-se idêntico, sabemos que ele não mais mudará e com isso podemos eliminá-lo dos nossos cálculos e escrevê-lo na saída do nosso programa.

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

Assim, caso os dígitos mais significativos do intervalo se igualem, escrevemos o dígito na saída (resolvendo o problema 1) e atualizamos o nosso intervalo para ignorar esse dígito (i.e. Os valores low e high passam a ser os dígitos da segunda casa após a vírgula em diante).

Os novos valores para low e high nesse caso serão:

```
ultimo_digito = 1000 * (high / 1000)
high = (high - ultimo_digito) * 10 + 9
low = (l       ow - ultimo_digito) * 10
```

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

No caso de high somamos 9 pois o valor original de high representava 0,999..., uma dízima periódica cujo próximo dígito que vamos “buscar” sempre será 9. Em ambos os casos multiplicamos por 10 para poder “ganhar” um dígito na nossa precisão.

No final deste processo, emitimos o valor de low na saída, que representa os últimos dígitos do nosso código aritmético (os outros dígitos já foram emitidos durante o processo).

Codificação Aritmética

EXEMPLO

Pretende-se utilizar a codificação aritmética da cadeia seguinte:

A_ASA_DA_CASA

As probabilidades dessa cadeia são:

Símbolo	Ocorrências	Probabilidade	Acumulada
A	6	0.4615	0.4615
_	3	0.2308	0.6923
S	2	0.1538	0.8462
C	1	0.0769	0.9231
A	1	0.0769	1.0000

Codificação Aritmética

EXEMPLO

Entrada	Low	High	Saída gerada
	0000	9999	-
A	0000	4614	-
_	2130	3194	-
A	2130	2620	2
	1300	6209	-
S	4699	5453	-
A	4699	5046	-
_	4859	4938	4
	8590	9389	-
D	9328	9389	9

	3280	3899	3
	2800	8999	-
A	2800	5660	-
_	4120	4779	4
-	1200	7799	-
C	6784	7291	-
A	6784	7017	-
S	6946	6981	6
	9460	9819	9
	4600	8199	-
A	4600	6260	-

Codificação Aritmética

CÁLCULO COM PRECISÃO FINITA

Obtemos como saída os dígitos 2493469, que acrescidos dos dígitos de low (ignoram-se os zeros no final) fica 249346946.

Este é nosso código aritmético para a frase inicial.

A_ASA_DA_CASA = 249346946

Este número pode ser expresso em 28 bits. A frase inicial tem 13 caracteres, que podem ser expressos com 3 bits cada, totalizando 39 bits.

Com a compressão aritmética economizamos 11 bits.

Comparação entre a Codificação Huffman e Aritmética

- Codificação Aritmética é mais complicada que a Huffman porém permite que sejam codificados seqüências de símbolos.
- Para a codificação aritmética, quanto maior for a sequência, mais o código se aproxima da entropia. Porém devemos escolher o tamanho da seqüência de forma que torne-se viável.
- A codificação Huffman é melhor que a aritmética quando se tem grandes alfabetos.
- Uma grande vantagem da codificação aritmética é a facilidade de se implementar sistemas com múltiplos codificadores aritméticos.

Referências Bibliográficas

Algoritmos e Estruturas de Dados. Disponível em: orion.lcg.ufrj.br/algoritmos/Topicos%20Extra.ppt. Acesso em: 17 out. 2008.

Codificação Huffman e Aritmética. Disponível em: www.cic.unb.br/~lamar/te729/Aulas/cod_aula2.pdf. Acesso em: 15 out. 2008.

Processamento e Comunicação Multimédia. Disponível em: www.di.ubi.pt/cursos/mestrados/mei/disciplinas/5052/fichs/Apres_Topico_2.pdf. Acesso em: 17 out. 2008.

FIM