Algoritmia Aplicada

Ano lectivo 2011-2012

Aula 3 – Processamento de cadeias de caracteres: Compressão

Sumário

- Processamento de cadeias de caracteres
  - Compressão
    - Codificação segundo comprimentos de séries
      - Caso de ficheiros de texto
      - Caso de ficheiros binários
    - Codificação segundo comprimentos variáveis (método de Huffman)

Processamento de cadeias de caracteres - Compressão

- **Objectivo (algoritmos de compressão):**

  Reduzir espaço ocupado pelos dados sem grande preocupação com o tempo gasto no processo

  Perspectiva contrária aquela que normalmente preside à elaboração dos algoritmos:

  "*Minimizar o tempo de execução conservando, se possível, o espaço*"

Processamento de cadeias de caracteres - Compressão

- **Motivação (original):**

  Ironicamente, os métodos que vamos estudar foram desenvolvidos para **minimizar a quantidade de informação** necessária em Sistemas de Comunicação.

  Deste modo, o seu objectivo original era **poupar tempo**.

Processamento de cadeias de caracteres - Compressão

- **Abordagem:**

  Aproveitar a redundância de dados:

  - Em **ficheiros de texto** ("text files"), onde há caracteres que se repetem com muita frequência
  - Em **ficheiros de figuras** ("master files"), onde há grandes áreas homogéneas
  - Em **ficheiros** para a representação digital do **som** e outros sinais analógicos, com padrões muito repetidos

Processamento de cadeias de caracteres - Compressão

- **Resultados (típicos):**

  - Economias de **20% a 50%** em **ficheiros de texto**
  - Economias de **50% a 90%** em **ficheiros binários** que não tenham uma distribuição aleatória dos bits
    (caso em que os ganhos são poucos!)

Processamento de cadeias de caracteres - Compressão

- **Absurdo:** "***Dispor de um método tão bom tão bom que, uma vez aplicado a qualquer ficheiro, reduz o espaço ocupado***"

  Se assim fosse, a aplicação do método ao ficheiro original, iria provocar uma primeira redução, criando um novo ficheiro comprimido

  Uma 2ª aplicação do método a este último, iria provocar nova redução, e assim sucessivamente

  Ou seja, seria possível produzir um ficheiro tão pequeno quanto se quisesse!

Processamento de cadeias de caracteres - Compressão

- **Conclusão:**

  Qualquer método de compressão, por melhor que seja, irá originar, em certos casos, ficheiros maiores do que o ficheiro original!

Processamento de cadeias de caracteres - Compressão

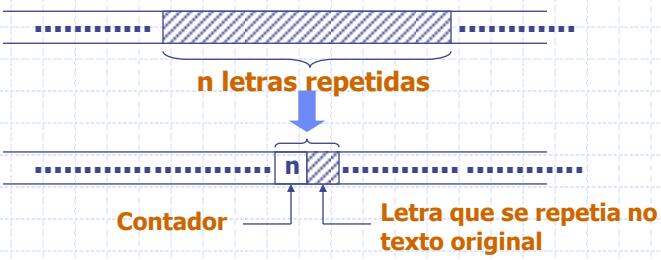
- **Métodos:**
  - **COMPRIMENTOS DE SÉRIES**
  - **COMPRIMENTOS VARIÁVEIS (Método de Huffman)**

Comprimentos de Séries

- **Codificação segundo comprimentos de séries**
  - **Este processo de compactação tem interesse quando há sequências de caracteres repetidos no ficheiro a comprimir**
  - **A ideia do método é substituir cada sequência pelo caracter que se repete e por um contador**
  - **Por exemplo, podemos dizer que o texto**
    DDDDBBCCC2222AA....
    é composto por 4 D's, seguidos de 2 B's, de 3 C's, de 4 2's, etc.

Comprimentos de Séries

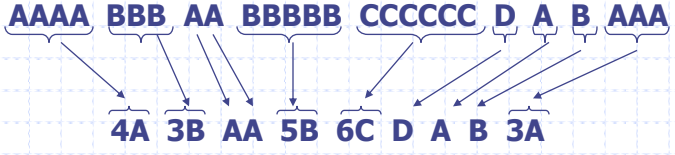
- **Esta ideia pode ser realizada de diferentes formas, dependendo, por exemplo, do tipo de caracteres que constituem o texto**
  - **a) Ficheiros de texto contendo apenas letras**



Nota: Não interessa codificar as sequências com n<=2

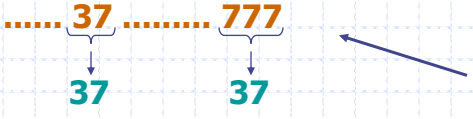
Comprimentos de Séries

- **Exemplo:**



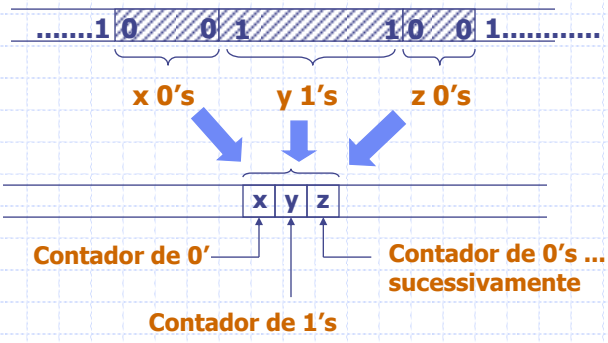
Nota1: O texto original, de 26 caracteres, ficou comprimido num texto (código) de apenas 15 caracteres

Nota2: Se o texto contivesse dígitos, este processo não funcionava, porque os dígitos são usados como contadores, originando ambiguidades.



Comprimentos de Séries

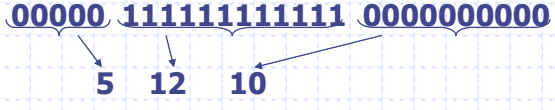
➢ b) Ficheiros Binários



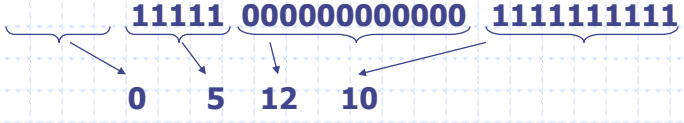
Nota: Apenas se memorizam os contadores, visto que as sequências são alternadamente de 0's e de 1's

Comprimentos de Séries

- **Exemplo 1:**



- **Exemplo 2:**

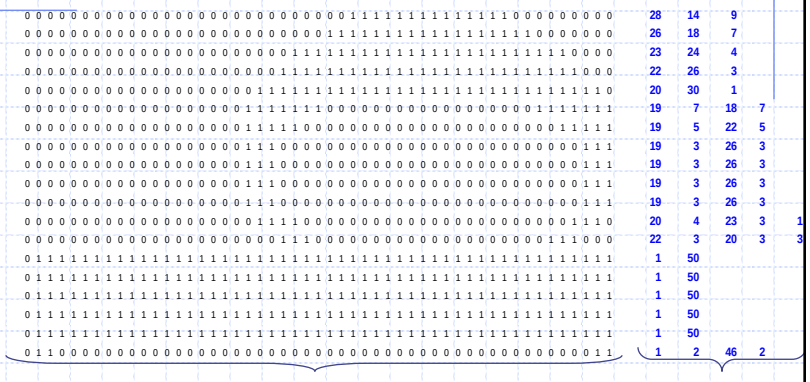


No início há uma sequência de 1's, ou seja, há 0 0's no início do ficheiro

Comprimentos de Séries

- **Exemplo 3 (Imagem de bits da letra q):**

Com um pouco de atenção, vemos aqui a imagem de um q deitado



Tipo de informação que poderá ter de ser processada por um sistema de formatação de texto

Lista de números que poderá ser usada para armazenar a imagem comprimida de q (63 números ← 63 contadores ← 63 sequências)

Comprimentos de Séries

- **Exemplo 3 (continuação):**

⇒ **Para guardar a representação explícita da letra q, é necessário um ficheiro binário com um total de 975 bits (969 + 6)**

A imagem tem 19 x 51 = 969 bits
linhas colunas

Para reconstruir a imagem é preciso memorizar também o nº de bits por linha, no caso 51 bits/linha: $51=110011_{(2)}$

6 bits

Comprimentos de Séries

- **Exemplo 3 (continuação):**

⇒ **Este ficheiro de 975 bits pode ser comprimido num ficheiro de apenas 384 bits (378 + 6)**

63 contadores x 6 bits para representar cada contador em binário = 378 bits

Memorizar 51 em binário pela mesma razão anterior → 6 bits

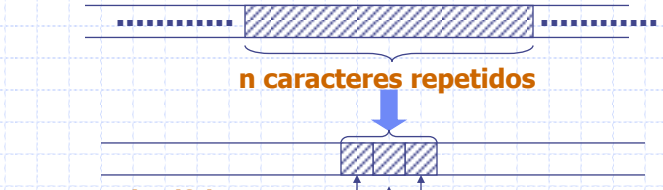
Comprimentos de Séries

➢ **c) Ficheiros de texto (caso geral)**

- **O método a apresentar permite codificar qualquer texto, contendo caracteres de um alfabeto fixo, usando somente caracteres desse alfabeto.**
- **Note-se que o 1º método (a) codificava textos apenas de letras, usando para o efeito, dígitos. Logo, não satisfaz o princípio antes enunciado.**

Comprimentos de Séries

- **c) Ficheiros de texto (caso geral) – Método:**



Caracter do alfabeto usado que raramente aparece (pouco esperado)....

Contador, representando o valor n

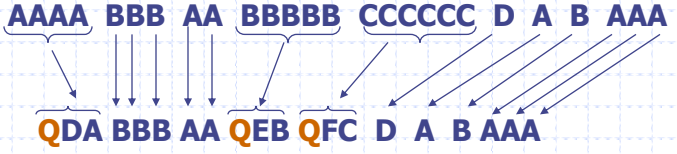
Caracter que se repetia no texto original

O aparecimento deste caracter indica que os dois caracteres seguintes são um par (contador,caracter)

Nota: Não interessa codificar as sequências com n<=3

Comprimentos de Séries

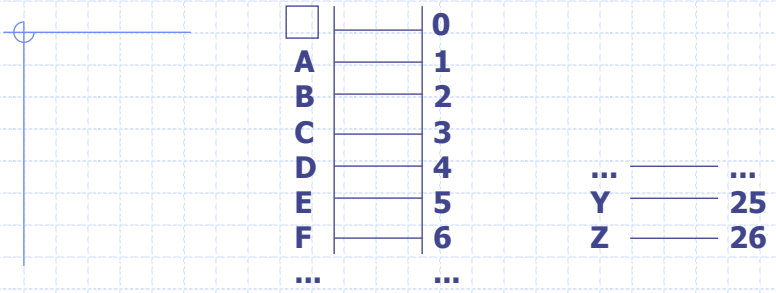
- **Exemplo 1 (idem ao já apresentado atrás):**



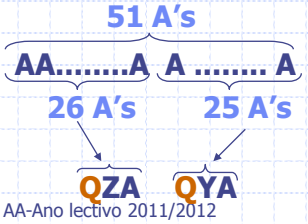
Q é o caracter usado para indicar que os dois caracteres seguintes são um par (contador,caracter)

Admitiu-se que tanto o texto original como o texto codificado só utilizam as 26 letras do alfabeto e o Espaço (□), de modo que os caracteres representam os seguintes valores quando são usados como contadores:

Comprimentos de Séries



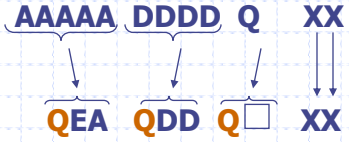
- **Exemplo 2 (mesmas regras do exemplo anterior):**



Uma alternativa, para sequências muito compridas, seria usar mais do que um caracter para codificar os contadores

Comprimentos de Séries

- **Exemplo 3 (mesmas regras dos exemplos anteriores):**



Note-se que o espaço (□) representa o valor zero quando é usado como contador

- **Conclusão:**

A codificação segundo comprimentos de séries não é particularmente interessante para ficheiros de texto, visto que o único caracter que se espera que ocorra muitas vezes repetido é o espaço!

Terá muito mais interesse utilizar uma técnica como aquela que vai ser apresentada de seguida.

Comprimentos variáveis – Método de Huffman

- **Codificação segundo comprimentos variáveis (Método de Huffman)**

- **Ideia:** **Abandonar o modo como os ficheiros de texto são usualmente gravados;**
  Em vez de se usar os 8 bits para cada caracter, usar comprimentos variáveis:
  - **poucos bits para os caracteres mais usuais**
  - **mais bits para os caracteres raramente usados**

Comprimentos variáveis – Método de Huffman

- **Vantagem:** **Este procedimento pode economizar grande espaço em ficheiros de texto e mesmo noutros tipos de ficheiros**

- **Exemplo:** **Para descrever o método de Huffman, vamos recorrer a um exemplo que consiste em codificar/comprimir o texto seguinte:**

  " A SIMPLE STRING TO BE ENCODED USING A MINIMAL NUMBER OF BITS"

Comprimentos variáveis – Método de Huffman

- **1º Passo: Determinar a frequência de cada caracter no texto (= nº de vezes que este ocorre):**

Espaço	:	11	L	:	2
A	:	3	M	:	4
B	:	3	N	:	5
C	:	1	O	:	3
D	:	2	P	:	1
E	:	5	R	:	2
F	:	1	S	:	4
G	:	2	T	:	3
I	:	6	U	:	2

($\sum$ Frequências= Nº caracteres do texto)

Comprimentos variáveis – Método de Huffman

- **2º Passo: Construir uma árvore de codificação (árvore de Huffman) do fundo para o topo (raiz) usando o seguinte processo:**

  0- Criar tantos nós quantos os caracteres identificados no texto.
  Cada nó estará associado a um dado caracter, e conterá a frequência desse caracter no texto

11	3	3	1	2	5	1	2	6	2	4	5	3	1	2	4	3
ESPAÇO	A	B	C	D	E	F	G	I	L	M	N	O	P	R	S	T

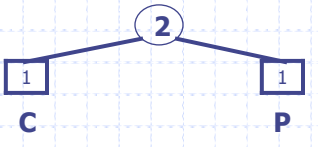
Comprimentos variáveis – Método de Huffman

1- De entre as sub-árvores existentes (no ínicio há tantas sub-árvores quantos os nós criados) seleccionar as duas sub-árvores cujas raízes tenham as menores frequências

No início, há 3 nós nestas condições (frequência=1) e podemos por isso seleccionar quaisquer dois, por exemplo:

Comprimentos variáveis – Método de Huffman

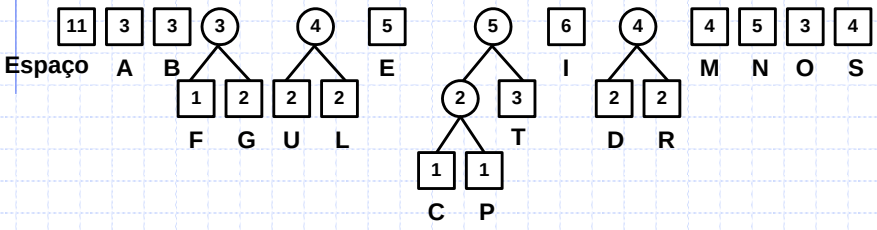
2- Criar um novo nó cujos filhos sejam os nós (raízes) seleccionados em 1.
Atribuir ao novo nó uma frequência que é a soma das frequências dos seus dois filhos



3- Voltar a 1-, ou terminar (se só existir uma árvore)

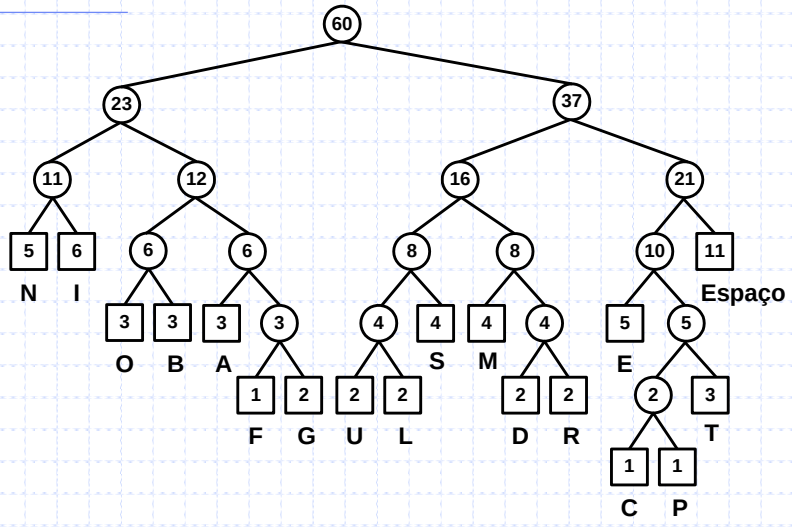
Comprimentos variáveis – Método de Huffman

- **2º Passo (continuação): O processo descrito conduz à seguinte floresta de árvores, após terem sido seleccionados todos os nós com frequência 2:**



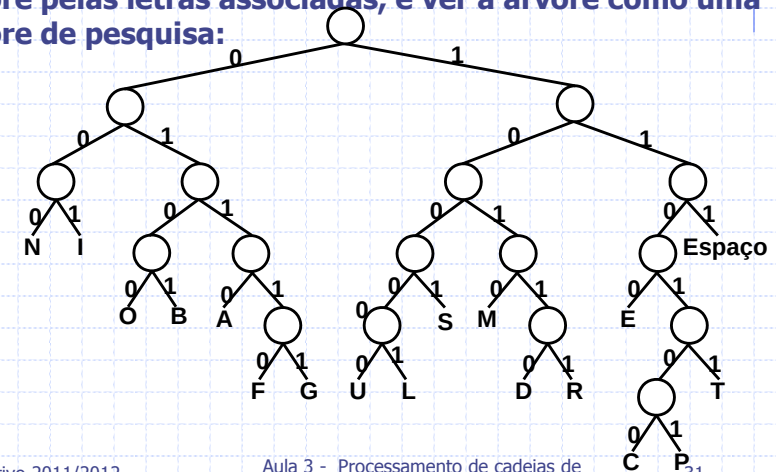
No fim temos a seguinte árvore, sendo de notar que os nós originais com maior frequência ficam mais junto da raiz, enquanto que aqueles com menor frequência ficam mais longe da raiz:

Comprimentos variáveis – Método de Huffman



Comprimentos variáveis – Método de Huffman

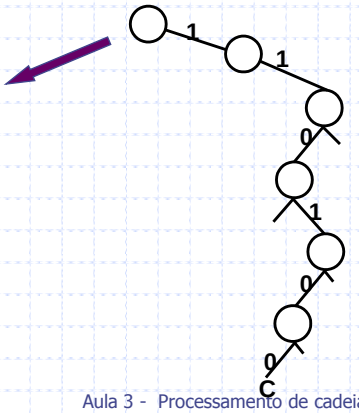
- **3º Passo: Substituir as frequências dos nós no fundo da árvore pelas letras associadas, e ver a árvore como uma árvore de pesquisa:**



Comprimentos variáveis – Método de Huffman

- **4º Passo: Utilizar a árvore para obter directamente o código de cada letra. Por exemplo:**

N – 000
I – 001
C – 110100



Comprimentos variáveis – Método de Huffman

- **5º Passo: Finalmente podemos codificar a mensagem, obtendo-se para o exemplo:**

A	□	S	I	M	P	L	E	□	S	T
0 1 1 0	1 1 1	1 0 0 1	0 0 1	1 0 1 0	1 1 0 1 0 1	1 0 0 0 1	1 1 0 0	1 1 1	1 0 0 1	1 1 0 1 1

1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0

0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 0 0

0 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1 1

0 0 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 0

0 1 1 1 0 1 1 1 0 0 1

Comprimentos variáveis – Método de Huffman

Nota 1 : A mensagem ficará codificada em apenas 236 bits. Note-se que a mensagem original ficaria codificada em 300 bits (economia de 21%) se fosse usada uma codificação compacta, com palavras de 5 bits por caracter:

0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0
1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1
0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 0 0
0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0
0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 1 1 1
0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 1

A I-ésima letra do alfabeto é representada por uma palavra de 5 bits correspondente à representação em binário do número I.
O espaço é representado por 00000

Por exemplo, o S é a 19ª letra donde temos o código 10011 visto que:
$19_{(10)}=10011_{(2)}$

Comprimentos variáveis – Método de Huffman

Nota 2 : Não existem delimitadores entre os caracteres que foram armazenados, muito embora existam caracteres codificados com diferentes números de bits!
Então, como é que será possível descodificar a mensagem, isto é, saber quando termina um caracter e começa o seguinte?

- **Descodificação: É usada a mesma árvore de pesquisa que foi usada para a codificação. Começando na raiz, desce-se na árvore, de acordo com os bits na mensagem codificada, até se atingir um nó externo. Então, escreve-se o caracter respectivo e recomeça-se na raiz.**

Exemplo: Descodificar a mensagem

11001110111001101101000011000 1

Solução: **"É FÁCIL"**

Comprimentos variáveis – Método de Huffman

- **Conclusão:**
  - É necessário guardar a árvore juntamente com a mensagem codificada
  - Assim, a economia de memória não é tão grande como foi referido atrás
  - Método só será eficaz para ficheiros grandes, em que a economia na mensagem compense o gasto com a memorização da árvore; ou em situações em que não é guardada a árvore, por ser sempre a mesma para qualquer mensagem (por exemplo, uma árvore baseada na ocorrência estatística dos caracteres em Português, poderia ser usada para mensagens em Português).

Próxima aula

- Processamento de Cadeias de Caracteres
  - Criptação