ELE32
Introdução a Comunicações
Codificação de Fonte

ITA

2º. Semestre de 2016

manish@ita.br

Como representar informação de forma eficiente?

- Nem sempre fonte de informação está representada de forma eficiente.
- Codificação de fonte associa uma palavra-código para cada símbolo/sequência de símbolos com o objetivo de representar informação de forma eficiente
- Métodos sem perda de informação: Huffman, Golumb, Lempel-Ziv e variantes, entre outros, permitem obter os símbolos/sequência de símbolos de forma perfeita
- Métodos com perda de informação: mp3, jpeg, avi, etc., não permitem.

Códigos de Huffman

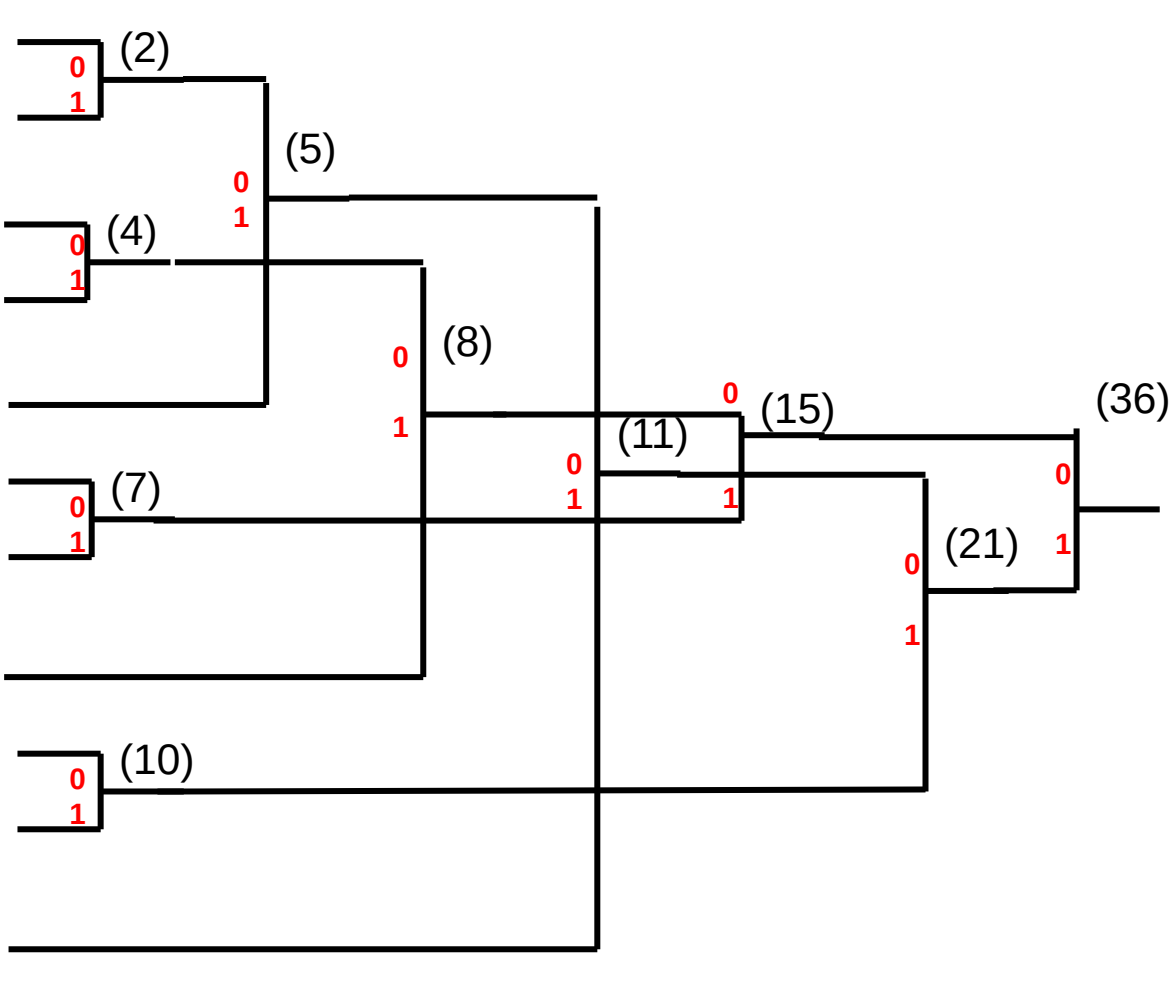
- Método para associar palavras código mais curtas para símbolos mais prováveis.
- Exige e satisfaz condição de ser de prefixo livre: nenhuma palavra código é prefixo de outra
- Esta condição permite identificar o início e término de cada palavra código em uma sequência de bits.

Códigos de Huffman

- Método para associar palavras código mais curtas para símbolos mais prováveis.
- Exige e satisfaz condição de ser de prefixo livre: nenhuma palavra código é prefixo de outra
- Esta condição permite identificar o início e término de cada palavra código em uma sequência de bits
- Associação é feita montando uma árvore e rotulando os ramos da mesma

Códigos de Huffman: exemplo (1/2)

10000	2	(1)
10001	12	(1)
0000	3	(2)
0001	11	(2)
1001	4	(3)
010	10	(3)
011	5	(4)
001	9	(4)
110	6	(5)
111	8	(5)
101	7	(6)



Códigos de Huffman: exemplo (2/2)

- Sequência a ser transmitida: 2· 12· 5· 7· 4· 3
- Bits transmitidos:100000100010111010010000
- É possível identificar o início e término de cada palavra código e associar o símbolo corresspondente.

Problemas com códigos de Huffman

- Exige conhecimento a priori da probilidade de se transmitir cada símbolo
- Só é ótimo se probabilidades forem iguais a 2-k, onde k é um número inteiro

Códigos de Lempel Ziv

- Não exige conhecimento a priori da probabilidade dos símbolos
- É assintoticamente ótimo
- Tenta descrever novas sequências a partir de sequências que já apareceram e que estão contidas em um dicionário.

Lempel-Ziv: Algoritmo de compactação

- Preencher *Dic[i]* com todos os símbolos possíveis
- Obtenha símbolos da fonte de informação até que uma sequência ***Sc*** que não pertence ao código seja encontrada.
- Escreva na saída o endereço da sequência *S* que pertence a *Dic[i]*. Utilize somente a quantidade de bits necessária
- Reinicie a leitura, utilizando c como primeiro símbolo da nova sequência
- Repita o procedimento até que todos os símbolos da entrada sejam processados

Lempel-Ziv: Exemplo

- Sequência **ABCBABABCBCABAC**

i	Dic[i]	Sc	S= Dic[j]	j	b	j (binário)
1	A	-	-	-		
2	B	-	-	-		
3	C	AB	A	1	2	01
4	AB	BC	B	2	3	010
5	BC	CB	C	3	3	011
6	CB	BA	B	2	3	010
7	BA	ABA	AB	4	3	0100
8	ABA	ABC	AB	4	4	0100
9	ABC	CBC	CB	6	4	1010
10	CBC	CA	C	3	4	0011
11	CA	ABAC	ABA	8	4	0110
12	ABAC	C	-	-	4	-

- Sequência de saída: ABC0101001101001000100101000110110C

Lempel-Ziv: Algoritmo de descompactação:

- Sabendo o dicionário que contém todos os símbolos possíveis, leia uma quantidade de bits suficiente para se determinar o endereço do prefixo a ser escrito na saída.
- Nao sabemos neste momento o valor de Dic[i], mas sabemos que vale S+”?”
- Leia a proxima quantidade de bits e determine o conteúdo parcial da proxima posição de Dic[i]. O primeiro símbolo da sequência na próxima posição é igual ao último símbolo da sequência na posição anterior
- Repita até que todos os símbolos sejam processados. Concatene o restante (nao codificado) a saída.

Atividade

- Implemente os algoritmos de compactação e de descompactação. Varie o tamanho permitido para o dicionário, resetando-o quando necessário, mantendo os símbolos.
- Aplique a textos de diversos idiomas. O arquivo obtido após a compactação e descompactação deve ser igual ao original
- Qual é o tamanho do arquivo compactado? Como ele se compara com a entropia do texto?
- Outras análises são solicitadas no roteiro