Análise de Algoritmos
Aula 12

Prof. Murilo V. G. da Silva

DINF/UFPR

(material da disciplina: André Guedes, Renato Carmo, Murilo da Silva)

Código de Huffman (1952)

Um *código binário* (*codicação binária*) para um alfabeto Σ:

Código de Huffman (1952)

Um *código binário* (*codicação binária*) para um alfabeto Σ:
Uma função $c : \Sigma \to \{0,1\}^+$.

Código de Huffman (1952)

Um *código binário* (*codicação binária*) para um alfabeto Σ:
Uma função $c : \Sigma \to \{0,1\}^+$.

Exemplo 62

$$\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{e}, \mathtt{i}, \mathtt{m}, \mathtt{n}, \mathtt{r}, \mathtt{s}, *\},$$

σ	$c_1(\sigma)$
a	0
b	1
c	00
e	01
i	10
m	11
n	000
r	001
s	010
*	011

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Exemplo 64

Nesta aula vamos usar constantemente a seguinte string m sobre Σ^+:

$$m = \texttt{ana e mariana merecem mais seis bananas}$$

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Exemplo 64

Nesta aula vamos usar constantemente a seguinte string m sobre Σ^+:

$$m = \texttt{ana e mariana merecem mais seis bananas}$$

- Pergunta: Quantos bits precisamos para codificar m em c_1?

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Exemplo 64

Nesta aula vamos usar constantemente a seguinte string m sobre Σ^+:

$$m = \texttt{ana e mariana merecem mais seis bananas}$$

- Pergunta: Quantos bits precisamos para codificar m em c_1?
- Respota: 83

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Exemplo 64

Nesta aula vamos usar constantemente a seguinte string m sobre Σ^+:

$$m = \texttt{ana e mariana merecem mais seis bananas}$$

- Pergunta: Quantos bits precisamos para codificar m em c_1?
- Respota: 83
- A codificação c_1 é uma boa codificação?

Código de Huffman (1952)

A codificação de $m = (m_1, \ldots, m_n)$, sendo $m \in \Sigma^+$ em c_1 é a string binária:

$$c_1(m) = c_1(m_1) \cdot \ldots \cdot c_1(m_n) \in \{0,1\}^+.$$

Exemplo 63

Na codificação c_1 do slide passado, a palavra `mais` é

$$c_1(\texttt{mais}) = c_1(\texttt{m}) \cdot c_1(\texttt{a}) \cdot c_1(\texttt{i}) \cdot c_1(\texttt{s}) = 11 \cdot 0 \cdot 10 \cdot 010 = 11010010$$

Exemplo 64

Nesta aula vamos usar constantemente a seguinte string m sobre Σ^+:

$$m = \texttt{ana e mariana merecem mais seis bananas}$$

- Pergunta: Quantos bits precisamos para codificar m em c_1?
- Respota: 83
- A codificação c_1 é uma boa codificação? Veremos...

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Exemlo: ASCII

A Tabela ASCII é um código de tamanho fixo pois cada letra do alfabeto é representado por uma sequência de tamanho 8.
A árvore representando a Tabela ASCII tem altura 8 e todas as letras estão nas folhas.

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Exemlo: ASCII

A Tabela ASCII é um código de tamanho fixo pois cada letra do alfabeto é representado por uma sequência de tamanho 8.
A árvore representando a Tabela ASCII tem altura 8 e todas as letras estão nas folhas.

- Quantos bits precisaríamos para reprentar usando código ASCII

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Exemlo: ASCII

A Tabela ASCII é um código de tamanho fixo pois cada letra do alfabeto é representado por uma sequência de tamanho 8.
A árvore representando a Tabela ASCII tem altura 8 e todas as letras estão nas folhas.

- Quantos bits precisaríamos para reprentar usando código ASCII
- Respota: 312

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Exemlo: ASCII

A Tabela ASCII é um código de tamanho fixo pois cada letra do alfabeto é representado por uma sequência de tamanho 8.
A árvore representando a Tabela ASCII tem altura 8 e todas as letras estão nas folhas.

- Quantos bits precisaríamos para reprentar usando código ASCII
- Respota: 312
- Para codificações fixas, isso é bom?

Código de Huffman (1952)

Um código binário pode ser representado por uma árvore binária (ver desenho em aula)

Código de Tamanho fixo

O código c é *de tamanho fixo* se

- Todas as letras de Σ são representadas por sequências de mesmo tamanho.
- Obs: Em códigos de tamanho fixo, as letras de Σ estão nas folhas

Exemlo: ASCII

A Tabela ASCII é um código de tamanho fixo pois cada letra do alfabeto é representado por uma sequência de tamanho 8.
A árvore representando a Tabela ASCII tem altura 8 e todas as letras estão nas folhas.

- Quantos bits precisaríamos para reprentar usando código ASCII
- Respota: 312
- Para codificações fixas, isso é bom? Veremos...

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

Necessariamente $2^t \geq |\Sigma|$

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja,

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja, $t \geq \lg |\Sigma|$

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja, $t \geq \lg |\Sigma|$

Com isso, t deve ser no mínimo:

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja, $t \geq \lg |\Sigma|$

Com isso, t deve ser no mínimo:

$$t = \lceil \lg |\Sigma| \rceil$$

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja, $t \geq \lg |\Sigma|$

Com isso, t deve ser no mínimo:

$$t = \lceil \lg |\Sigma| \rceil$$

Consequentemente para **toda string** $m \in \Sigma^+$ e **toda codificação de tamanho fixo** c,

$$|c(m)| \geq t|m| = \lceil \lg |\Sigma| \rceil \, |m|.$$

Código de Huffman (1952)

Existem 2^t strings binárias de tamanho t.

- Considere uma codificação de tamanho fixo t para Σ

$$\text{Necessariamente} \quad 2^t \geq |\Sigma|$$

ou seja, $t \geq \lg |\Sigma|$

Com isso, t deve ser no mínimo:

$$t = \lceil \lg |\Sigma| \rceil$$

Consequentemente para **toda string** $m \in \Sigma^+$ e **toda codificação de tamanho fixo** c,

$$|c(m)| \geq t|m| = \lceil \lg |\Sigma| \rceil \, |m|.$$

Codificando a mensagem `m` do nosso exemplo

No exemplo,

$$|c_1(m)| \geq \lceil \lg |\Sigma| \rceil \, |m| = \lceil \lg 10 \rceil \times 39 = 4 \times 39 = 156.$$

Código de Huffman (1952)

Voltando a codificação de tamanho variável

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Ou seja,

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Ou seja, $t \geq \lceil \lg(|\Sigma| + 2) \rceil - 1 = (\lfloor \lg(|\Sigma| + 1) \rfloor + 1) - 1 = \lfloor \lg(|\Sigma| + 1) \rfloor$.

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Ou seja, $t \geq \lceil \lg(|\Sigma| + 2) \rceil - 1 = (\lfloor \lg(|\Sigma| + 1) \rfloor + 1) - 1 = \lfloor \lg(|\Sigma| + 1) \rfloor$.

Então para todo $m \in \Sigma^+$ e toda codificação c,

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Ou seja, $t \geq \lceil \lg(|\Sigma| + 2) \rceil - 1 = (\lfloor \lg(|\Sigma| + 1) \rfloor + 1) - 1 = \lfloor \lg(|\Sigma| + 1) \rfloor$.

Então para todo $m \in \Sigma^+$ e toda codificação c,

$$|m| \leq |c(m)| \leq \lfloor \lg(|\Sigma| + 1) \rfloor \, |m|.$$

Código de Huffman (1952)

Voltando a codificação de tamanho variável

$$\text{Existem} \quad \sum_{i=1}^{t} 2^i = 2^{t+1} - 2 \quad \text{strings binárias de tamanho até } t$$

- Num código de tamanho varável para Σ é necessário que $2^{l+1} - 2 \geq |\Sigma|$

Ou seja, $t \geq \lceil \lg(|\Sigma| + 2) \rceil - 1 = (\lfloor \lg(|\Sigma| + 1) \rfloor + 1) - 1 = \lfloor \lg(|\Sigma| + 1) \rfloor$.

Então para todo $m \in \Sigma^+$ e toda codificação c,

$$|m| \leq |c(m)| \leq \lfloor \lg(|\Sigma| + 1) \rfloor \, |m|.$$

No nosso exemplo da mensagem `m`

$$39 = |m| \leq |c_1(m)| \leq \lfloor \lg(k+1) \rfloor \, |m| = \lfloor \lg 11 \rfloor \, 39 = 117.$$

Código de Huffman (1952)

Olhando o Número de Ocorrências de uma palavra:

Código de Huffman (1952)

Olhando o Número de Ocorrências de uma palavra:

- dado $m \in \Sigma^+$

Código de Huffman (1952)

Olhando o Número de Ocorrências de uma palavra:

- dado $m \in \Sigma^+$
- $f(\sigma, m)$ o número de ocorrências (frequência) da letra σ na palavra m

Código de Huffman (1952)

Olhando o Número de Ocorrências de uma palavra:

- dado $m \in \Sigma^+$
- $f(\sigma, m)$ o número de ocorrências (frequência) da letra σ na palavra m

símbolos e frequência no nosso exemplo `m`

σ	$\lvert c_1(\sigma)\rvert$	$f(\sigma, m)$
a	1	8
b	1	1
c	2	1
e	2	5
i	2	3
m	2	4
n	3	4
r	3	2
s	3	4
*	3	6

Código de Huffman (1952)

Olhando o Número de Ocorrências de uma palavra:

- dado $m \in \Sigma^+$
- $f(\sigma, m)$ o número de ocorrências (frequência) da letra σ na palavra m

símbolos e frequência no nosso exemplo `m`

σ	$\lvert c_1(\sigma)\rvert$	$f(\sigma, m)$
`a`	1	8
`b`	1	1
`c`	2	1
`e`	2	5
`i`	2	3
`m`	2	4
`n`	3	4
`r`	3	2
`s`	3	4
`*`	3	6

Observe que o tamanho da palavra $c(m)$ é:

$$|c_1(m)| = \sum_{i=1}^{|m|} |c_1(m_i)| = \sum_{\sigma \in \Sigma} |c_1(\sigma)| f(\sigma, m).$$

Código de Huffman (1952)

No nosso exemplo inicial para c_1 para `m`

$$
\begin{aligned}
|c_1(m)| &= \sum_{\alpha \in \Sigma} |c_1(\alpha)| f(\alpha, m) \\
&= |c_1(a)| f(a,m) + |c_1(b)| f(b,m) + |c_1(c)| f(c,m) + |c_1(e)| f(e,m) \\
&+ |c_1(i)| f(i,m) + |c_1(m)| f(m,m) + |c_1(n)| f(n,m) + |c_1(r)| f(r,m) \\
&+ |c_1(s)| f(s,m) + |c_1(*)| f(*,m) \\
&= 1 \times 8 + 1 \times 1 + 2 \times 1 + 2 \times 5 \\
&+ 2 \times 3 + 2 \times 4 + 3 \times 4 + 3 \times 2 \\
&+ 3 \times 4 + 3 \times 6 \\
&= 8 + 1 + 2 + 10 + 6 + 8 + 12 + 6 + 12 + 18 \\
&= 83
\end{aligned}
$$

Código de Huffman (1952)

No nosso exemplo inicial para c_1 para `m`

$$
\begin{aligned}
|c_1(m)| &= \sum_{\alpha \in \Sigma} |c_1(\alpha)| f(\alpha, m) \\
&= |c_1(a)| f(a,m) + |c_1(b)| f(b,m) + |c_1(c)| f(c,m) + |c_1(e)| f(e,m) \\
&+ |c_1(i)| f(i,m) + |c_1(m)| f(m,m) + |c_1(n)| f(n,m) + |c_1(r)| f(r,m) \\
&+ |c_1(s)| f(s,m) + |c_1(*)| f(*,m) \\
&= 1 \times 8 + 1 \times 1 + 2 \times 1 + 2 \times 5 \\
&+ 2 \times 3 + 2 \times 4 + 3 \times 4 + 3 \times 2 \\
&+ 3 \times 4 + 3 \times 6 \\
&= 8 + 1 + 2 + 10 + 6 + 8 + 12 + 6 + 12 + 18 \\
&= 83
\end{aligned}
$$

Um problema com o código inicial:

Código de Huffman (1952)

No nosso exemplo inicial para c_1 para `m`

$$
\begin{aligned}
|c_1(m)| &= \sum_{\alpha \in \Sigma} |c_1(\alpha)| f(\alpha, m) \\
&= |c_1(a)| f(a,m) + |c_1(b)| f(b,m) + |c_1(c)| f(c,m) + |c_1(e)| f(e,m) \\
&+ |c_1(i)| f(i,m) + |c_1(m)| f(m,m) + |c_1(n)| f(n,m) + |c_1(r)| f(r,m) \\
&+ |c_1(s)| f(s,m) + |c_1(*)| f(*,m) \\
&= 1 \times 8 + 1 \times 1 + 2 \times 1 + 2 \times 5 \\
&+ 2 \times 3 + 2 \times 4 + 3 \times 4 + 3 \times 2 \\
&+ 3 \times 4 + 3 \times 6 \\
&= 8 + 1 + 2 + 10 + 6 + 8 + 12 + 6 + 12 + 18 \\
&= 83
\end{aligned}
$$

Um problema com o código inicial:

- Como decodificar?
- $00 = c_1(\mathrm{aa})$ ou $00 = c_1(\mathrm{c})$?

Código de Huffman (1952)

Códigos Livre de Prefixos

Códigos Livre de Prefixos (CLP)

Um código c é *livre de prefixos (preffix free code)* se

$$c(\sigma) \text{ não é prefixo de } c(\sigma'), \text{ para todo } \sigma, \sigma' \in \Sigma.$$

Código de Huffman (1952)

Códigos Livre de Prefixos

Códigos Livre de Prefixos (CLP)

Um código c é *livre de prefixos (preffix free code)* se

$$c(\sigma) \text{ não é prefixo de } c(\sigma'), \text{ para todo } \sigma, \sigma' \in \Sigma.$$

Exemplo de CLP

O seguinte código é livre de prefixos.

σ	$c_2(\sigma)$
a	0000
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	10
*	11

Código de Huffman (1952)

Nossa mensagem `m` no CLP c_2 anterior

$$
\begin{aligned}
|c_2(m)| &= \sum_{\sigma \in m} c_2(\sigma) f(\sigma) \\
&= |c_2(a)|f(a) + |c_2(b)|f(b) + |c_2(c)|f(c) + |c_2(e)|f(e) + |c_2(i)|f(i) \\
&+ |c_2(m)|f(m) + |c_2(n)|f(n) + |c_2(r)|f(r) + |c_2(s)|f(s) + |c_2(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 8 + 12 \\
&= 132
\end{aligned}
$$

Código de Huffman (1952)

Nossa mensagem `m` no CLP c_2 anterior

$$
\begin{aligned}
|c_2(m)| &= \sum_{\sigma \in m} c_2(\sigma) f(\sigma) \\
&= |c_2(a)|f(a) + |c_2(b)|f(b) + |c_2(c)|f(c) + |c_2(e)|f(e) + |c_2(i)|f(i) \\
&+ |c_2(m)|f(m) + |c_2(n)|f(n) + |c_2(r)|f(r) + |c_2(s)|f(s) + |c_2(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 8 + 12 \\
&= 132
\end{aligned}
$$

- Pergunta: Existem outros CLP melhores?

Código de Huffman (1952)

Nossa mensagem `m` no CLP c_2 anterior

$$
\begin{aligned}
|c_2(m)| &= \sum_{\sigma \in m} c_2(\sigma) f(\sigma) \\
&= |c_2(a)|f(a) + |c_2(b)|f(b) + |c_2(c)|f(c) + |c_2(e)|f(e) + |c_2(i)|f(i) \\
&+ |c_2(m)|f(m) + |c_2(n)|f(n) + |c_2(r)|f(r) + |c_2(s)|f(s) + |c_2(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 8 + 12 \\
&= 132
\end{aligned}
$$

- Pergunta: Existem outros CLP melhores?
- Resposta: Sim!

Código de Huffman (1952)

Seja T uma árvore para um CPL:

Código de Huffman (1952)

Seja T uma árvore para um CPL:

- A árvore T é cheia (todo nó que não é folha tem dois filhos)

Código de Huffman (1952)

Seja T uma árvore para um CPL:

- A árvore T é cheia (todo nó que não é folha tem dois filhos)
- As letras de Σ estão nas folhas de T

Código de Huffman (1952)

Seja T uma árvore para um CPL:

- A árvore T é cheia (todo nó que não é folha tem dois filhos)
- As letras de Σ estão nas folhas de T

Ideia: escolher as menores sequências para representar as letras mais frequentes

Código de Huffman (1952)

Seja T uma árvore para um CPL:

- A árvore T é cheia (todo nó que não é folha tem dois filhos)
- As letras de Σ estão nas folhas de T

Ideia: escolher as menores sequências para representar as letras mais frequentes

Uma CLP usando essa ideia

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

Código de Huffman (1952)

Seja T uma árvore para um CPL:

- A árvore T é cheia (todo nó que não é folha tem dois filhos)
- As letras de Σ estão nas folhas de T

Ideia: escolher as menores sequências para representar as letras mais frequentes

Uma CLP usando essa ideia

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

- Obs: neste exemplo escolhemos arbitrariamente ("sem método") as duas strings mais frequentes e usamos um CLP

Código de Huffman (1952)

Código de Huffman (1952)

CLP do slide anterior

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

Código de Huffman (1952)

CLP do slide anterior

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

$$
\begin{aligned}
|c_3(m)| &= \sum_{\sigma \in m} |c_3(\sigma)| f(\sigma) \\
&= |c_3(a)|f(a) + |c_3(b)|f(b) + |c_3(c)|f(c) + |c_3(e)|f(e) + |c_3(i)|f(i) \\
&+ |c_3(m)|f(m) + |c_3(n)|f(n) + |c_3(r)|f(r) + |c_3(s)|f(s) + |c_3(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 16 + 12 \\
&= 124
\end{aligned}
$$

Código de Huffman (1952)

CLP do slide anterior

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

$$
\begin{aligned}
|c_3(m)| &= \sum_{\sigma \in m} |c_3(\sigma)| f(\sigma) \\
&= |c_3(a)|f(a) + |c_3(b)|f(b) + |c_3(c)|f(c) + |c_3(e)|f(e) + |c_3(i)|f(i) \\
&+ |c_3(m)|f(m) + |c_3(n)|f(n) + |c_3(r)|f(r) + |c_3(s)|f(s) + |c_3(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 16 + 12 \\
&= 124
\end{aligned}
$$

- Aqui: Escolha arbitraria de duas strings mais frequentes para códigos menores

Código de Huffman (1952)

CLP do slide anterior

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

$$
\begin{aligned}
|c_3(m)| &= \sum_{\sigma \in m} |c_3(\sigma)| f(\sigma) \\
&= |c_3(a)|f(a) + |c_3(b)|f(b) + |c_3(c)|f(c) + |c_3(e)|f(e) + |c_3(i)|f(i) \\
&+ |c_3(m)|f(m) + |c_3(n)|f(n) + |c_3(r)|f(r) + |c_3(s)|f(s) + |c_3(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 16 + 12 \\
&= 124
\end{aligned}
$$

- Aqui: Escolha arbitraria de duas strings mais frequentes para códigos menores
- Pergunta: Da para fazer melhor?

Código de Huffman (1952)

CLP do slide anterior

σ	$c_3(\sigma)$
a	10
b	0001
c	0010
e	0011
i	0100
m	0101
n	0110
r	0111
s	0000
*	11

$$
\begin{aligned}
|c_3(m)| &= \sum_{\sigma \in m} |c_3(\sigma)| f(\sigma) \\
&= |c_3(a)|f(a) + |c_3(b)|f(b) + |c_3(c)|f(c) + |c_3(e)|f(e) + |c_3(i)|f(i) \\
&+ |c_3(m)|f(m) + |c_3(n)|f(n) + |c_3(r)|f(r) + |c_3(s)|f(s) + |c_3(*)|f(*) \\
&= 2 \times 8 + 4 \times 1 + 4 \times 1 + 4 \times 5 + 4 \times 3 \\
&+ 4 \times 4 + 4 \times 4 + 4 \times 2 + 4 \times 4 + 2 \times 6 \\
&= 16 + 4 + 4 + 20 + 12 + 16 + 16 + 8 + 16 + 12 \\
&= 124
\end{aligned}
$$

- Aqui: Escolha arbitraria de duas strings mais frequentes para códigos menores
- Pergunta: Da para fazer melhor?
- Pergunta: O que seria o ótimo neste contexto? (CLP + Frequências)

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Usar uma árvore binária T representando o código c

- $T = (r(T), E(T), D(T))$ é uma tripla

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Usar uma árvore binária T representando o código c

- $T = (r(T), E(T), D(T))$ é uma tripla
- $r(T)$ é um par (Γ, f_Γ),

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Usar uma árvore binária T representando o código c

- $T = (r(T), E(T), D(T))$ é uma tripla
- $r(T)$ é um par (Γ, f_Γ), $\Gamma \subseteq \Sigma$,

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Usar uma árvore binária T representando o código c

- $T = (r(T), E(T), D(T))$ é uma tripla
- $r(T)$ é um par (Γ, f_Γ), $\Gamma \subseteq \Sigma$, e f_Γ é a soma dos $f(\sigma)$ para todo $\sigma \in \Gamma$

Código de Huffman (1952)

Código de Huffman (CH)

Definição do Problema:

- Entrada: Um alfabeto Σ e uma função $f : \Sigma \to \mathbb{Q}$.
- Saída: Um código binário livre de prefixos para Σ que minimiza

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma).$$

Usar uma árvore binária T representando o código c

- $T = (r(T), E(T), D(T))$ é uma tripla
- $r(T)$ é um par (Γ, f_Γ), $\Gamma \subseteq \Sigma$, e f_Γ é a soma dos $f(\sigma)$ para todo $\sigma \in \Gamma$

Codifica(Σ, f)

$S \leftarrow \{((\{\sigma\}, f(\sigma)), \Lambda, \Lambda)) \mid \sigma \in \Sigma\}$
Enquanto $|S| > 1$
    remova de S os dois pares (T_1, f_1) e (T_2, f_2) com os menores valores de f_1 e f_2
    acrescente o par $(((r(T_1) \cup r(T_2), f_1 + f_2), T_1, T_2))$ a S
Devolva T *onde* (T, f_Σ) *é o único par em* S

Código de Huffman (1952)

No nosso exemplo

α	$c_H(\alpha)$
a	10
b	00100
c	00101
e	000
i	0100
m	0101
n	110
r	0011
s	111
*	011

$$
\begin{aligned}
|c_H(m)| &= \sum_{\alpha \in s} |c_H(\alpha)| f(\alpha) \\
&= |c_H(a)|f(a) + |c_H(b)|f(b) + c_H(c)f(c) + |c_H(e)|f(e) \\
&+ |c_H(i)|f(i) + |c_H(m)|f(m) + |c_H(n)|f(n) \\
&+ |c_H(r)|f(r) + |c_H(s)|f(s) + |c_H(*)|f(*) \\
&= 2 \times 8 + 5 \times 1 + 5 \times 1 + 3 \times 5 \\
&+ 4 \times 3 + 4 \times 4 + 3 \times 4 \\
&+ 4 \times 2 + 3 \times 4 + 3 \times 6 \\
&= 16 + 5 + 5 + 15 + 12 + 16 + 12 + 8 + 12 + 18 \\
&= 119.
\end{aligned}
$$

Código de Huffman (1952)

O algoritmo está correto?

Código de Huffman (1952)

O algoritmo está correto?

Lema

Seja (Σ, f) uma instância de CH e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Existe uma solução desta instância na qual $((\{\sigma_1\}, f(\sigma_1)), \Lambda, \Lambda)$ e $((\{\sigma_2\}, f(\sigma_2)), \Lambda, \Lambda)$ são folhas irmãs de profundidade máxima.

Código de Huffman (1952)

O algoritmo está correto?

Lema

Seja (Σ, f) uma instância de CH e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Existe uma solução desta instância na qual $((\{\sigma_1\}, f(\sigma_1)), \Lambda, \Lambda)$ e $((\{\sigma_2\}, f(\sigma_2)), \Lambda, \Lambda)$ são folhas irmãs de profundidade máxima.

Lema

Seja (Σ, f) uma instância de CH e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Seja $\Sigma' = \Sigma - \{\sigma_1, \sigma_2\} \cup \tau$, onde $\tau \notin \Sigma$ e seja $f' : \Sigma' \to \mathbb{Q}$ dada por

$$f'(\sigma) = \begin{cases} f(\sigma), & \text{se } \sigma \neq \tau, \\ f(\sigma_1) + f(\sigma_2), & \text{se } \sigma = \tau. \end{cases}$$

As respostas c e c' das instâncias (Σ, f) e (Σ', f') tem mesmo custo, isto é,

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma) = \sum_{\sigma \in \Sigma'} |c'(\sigma)| f'(\sigma).$$

Código de Huffman (1952)

O algoritmo está correto?

Lema

Seja (Σ, f) uma instância de CH e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Existe uma solução desta instância na qual $((\{\sigma_1\}, f(\sigma_1)), \Lambda, \Lambda)$ e $((\{\sigma_2\}, f(\sigma_2)), \Lambda, \Lambda)$ são folhas irmãs de profundidade máxima.

Lema

Seja (Σ, f) uma instância de CH e sejam σ_1 e σ_2 duas letras frequência mínima nessa instância. Seja $\Sigma' = \Sigma - \{\sigma_1, \sigma_2\} \cup \tau$, onde $\tau \notin \Sigma$ e seja $f' \colon \Sigma' \to \mathbb{Q}$ dada por

$$f'(\sigma) = \begin{cases} f(\sigma), & \text{se } \sigma \neq \tau, \\ f(\sigma_1) + f(\sigma_2), & \text{se } \sigma = \tau. \end{cases}$$

As respostas c e c' das instâncias (Σ, f) e (Σ', f') tem mesmo custo, isto é,

$$\sum_{\sigma \in \Sigma} |c(\sigma)| f(\sigma) = \sum_{\sigma \in \Sigma'} |c'(\sigma)| f'(\sigma).$$

Teorema de Corretude

O Algoritmo Codifica é uma solução para CH.

Código de Huffman (1952)

Quanto custa?

Código de Huffman (1952)

Quanto custa?

Teorema

A execução de Codifica(Σ, f) com o conjunto S implementado por uma fila de prioridades toma tempo $\Omega(n) + \mathcal{O}(n \log n)$ onde $n = |\Sigma|$.

Código de Huffman (1952)

Quanto custa?

Teorema

A execução de Codifica(Σ, f) com o conjunto S implementado por uma fila de prioridades toma tempo $\Omega(n) + \mathcal{O}(n \log n)$ onde $n = |\Sigma|$.

Prova:

Código de Huffman (1952)

Quanto custa?

Teorema

A execução de Codifica(Σ, f) com o conjunto S implementado por uma fila de prioridades toma tempo $\Omega(n) + \mathcal{O}(n \log n)$ onde $n = |\Sigma|$.

Prova:

Seja (Σ, f) uma instância de CH e seja $n = |\Sigma|$.

Código de Huffman (1952)

Quanto custa?

Teorema

A execução de Codifica(Σ, f) com o conjunto S implementado por uma fila de prioridades toma tempo $\Omega(n) + \mathcal{O}(n \log n)$ onde $n = |\Sigma|$.

Prova:

Seja (Σ, f) uma instância de CH e seja $n = |\Sigma|$. Se o conjunto S é implementado por uma fila de prioridades, a análise linha a linha do Algoritmo Codifica resulta em

Código de Huffman (1952)

Quanto custa?

Teorema

A execução de Codifica(Σ, f) com o conjunto S implementado por uma fila de prioridades toma tempo $\Omega(n) + \mathcal{O}(n \log n)$ onde $n = |\Sigma|$.

Prova:

Seja (Σ, f) uma instância de CH e seja $n = |\Sigma|$. Se o conjunto S é implementado por uma fila de prioridades, a análise linha a linha do Algoritmo Codifica resulta em

$$\begin{aligned} T(\Sigma, f) &= \Theta(n) + (n-1)(\mathcal{O}(\log n) + \mathcal{O}(\log n)) + \Theta(1) \\ &= \Theta(n) + \mathcal{O}(n \log n) \\ &= \Omega(n) + \mathcal{O}(n \log n). \end{aligned}$$

Código de Huffman (1952)

Quanto custa (cont.)?

Código de Huffman (1952)

Quanto custa (cont.)?

Teorema

O tempo de pior caso do Algoritmo Codifica com o conjunto S implementado por uma fila de prioridades é $\Theta(n \log n)$ onde $n = |\Sigma|$.

Código de Huffman (1952)

Quanto custa (cont.)?

Teorema

O tempo de pior caso do Algoritmo Codifica com o conjunto S implementado por uma fila de prioridades é $\Theta(n \log n)$ onde $n = |\Sigma|$.

Prova:

Código de Huffman (1952)

Quanto custa (cont.)?

Teorema

O tempo de pior caso do Algoritmo Codifica com o conjunto S implementado por uma fila de prioridades é $\Theta(n \log n)$ onde $n = |\Sigma|$.

Prova:

Considere uma instância (Σ, f) de CH na qual todas as letras de Σ tem a mesma frequência.

Código de Huffman (1952)

Quanto custa (cont.)?

Teorema

O tempo de pior caso do Algoritmo Codifica com o conjunto S implementado por uma fila de prioridades é $\Theta(n \log n)$ onde $n = |\Sigma|$.

Prova:

Considere uma instância (Σ, f) de CH na qual todas as letras de Σ tem a mesma frequência.
Neste caso, cada uma das primeiras $\lfloor (n-1)/2 \rfloor$ inserções em S tomará tempo $\Omega(\log n)$ pois cada elemento inserido terá o valor de f maior do que $\lceil (n-1)/2 \rceil$ dos elementos em S.

Código de Huffman (1952)

Quanto custa (cont.)?

Teorema

O tempo de pior caso do Algoritmo Codifica com o conjunto S implementado por uma fila de prioridades é $\Theta(n \log n)$ onde $n = |\Sigma|$.

Prova:

Considere uma instância (Σ, f) de CH na qual todas as letras de Σ tem a mesma frequência.
Neste caso, cada uma das primeiras $\lfloor (n-1)/2 \rfloor$ inserções em S tomará tempo $\Omega(\log n)$ pois cada elemento inserido terá o valor de f maior do que $\lceil (n-1)/2 \rceil$ dos elementos em S.
Assim, a execução do laço tomará tempo $\Omega(n \log n)$ e como o tempo de execução do algoritmo é $\mathcal{O}(n \log n)$ (Teorma anterior) concluímos que, neste caso, será $\Theta(n \log n)$.