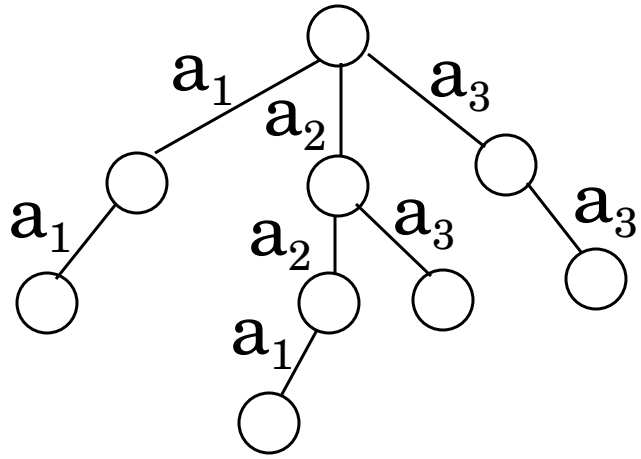
Aula 35: Árvore de Huffman

- Conceito de árvore binária de prefixo
- Árvore de Huffman
- Método guloso: princípios

Árvore Binária de Prefixo

- Árvore digital: alfabeto = { a_1, ... , a_m }



m = 3

- Códigos: Sejam

  S = { s_1, ..., s_n } = conjunto de elementos denominados símbolos

  A = { a_1, ..., a_m } = alfabeto

- Problema de codificação: atribuir a cada símbolo $s_i \in S$,

  uma seqüência de elementos $a_j \in A$, denominada código de s_i.

Árvore Binária de Prefixo

- Problema de codificação: atribuir a cada símbolo $s_i \in S$, uma seqüência de elementos $a_j \in A$, denominada código de s_i.
  - Exemplo:
  - $A = \{ a_1, a_2, a_3 \}$

    $a_2\ a_1\ a_3\ a_3\ a_1$ código de s_i

    $a_3\ a_1\ a_1$ código de s_k

Codificação

- Uma codificação de prefixo é aquela em que nenhum código é prefixo de outro.

- Exemplo: A = { a_1, a_2, a_3 } S = { s_1, s_2, s_3, s_4 }

$s_1 : a_1\ a_3\ a_2$	$s_1 : a_1\ a_3\ a_2$
$s_2 : a_3\ a_1\ a_2$	$s_2 : a_3\ a_1\ a_2$
$s_3 : a_3\ a_2\ a_1$	$s_3 : a_3\ a_1$
$s_4 : a_1\ a_2$	$s_4 : a_1\ a_2$
é de prefixo.	não é de prefixo, pois o código de s_3 é prefixo do de s_2.

Árvores Digitais Representam Códigos

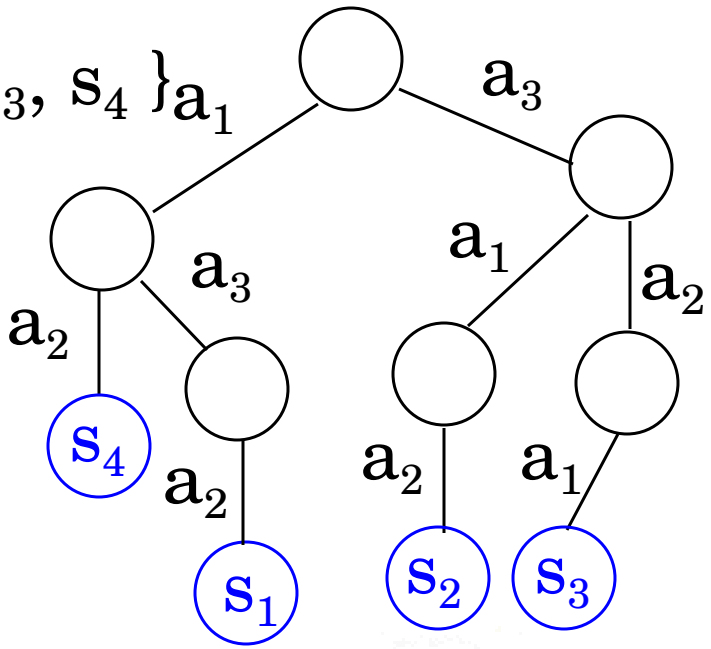
- $A = \{ a_1, .., a_m \}$ $\quad S = \{ s_1, ..., s_n \}$
- Existe uma árvore digital T, de alfabeto A, tal que:
  - cada símbolo $s_i \in S$ corresponde a algum nó de T.
  - o caminho da raiz de T até o nó correspondente a s_i é o símbolo de s_i.
- Ex.: $A = \{ a_1, a_2, a_3 \}$, $S = \{ s_1, s_2, s_3, s_4 \}$

$s_1 : a_1\ a_3\ a_2$

$s_2 : a_3\ a_1\ a_2$

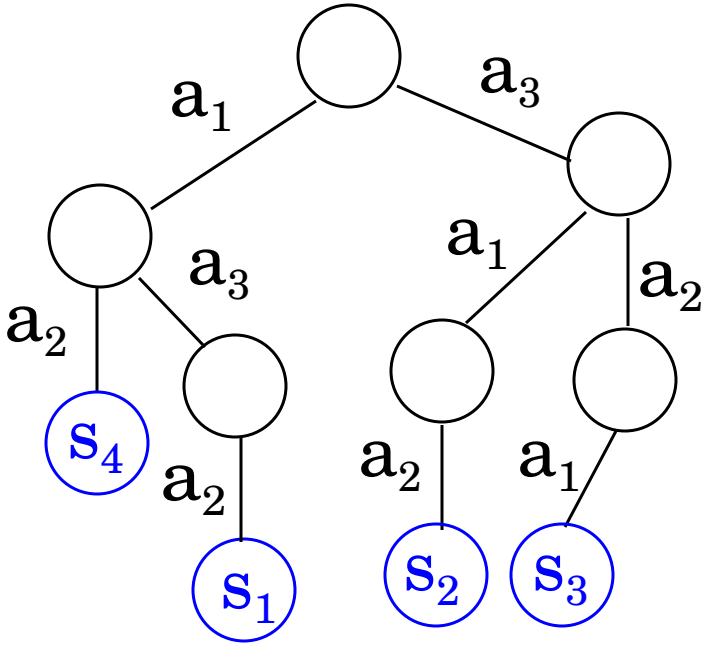
$s_3 : a_3\ a_2\ a_1$

$s_4 : a_1\ a_2$

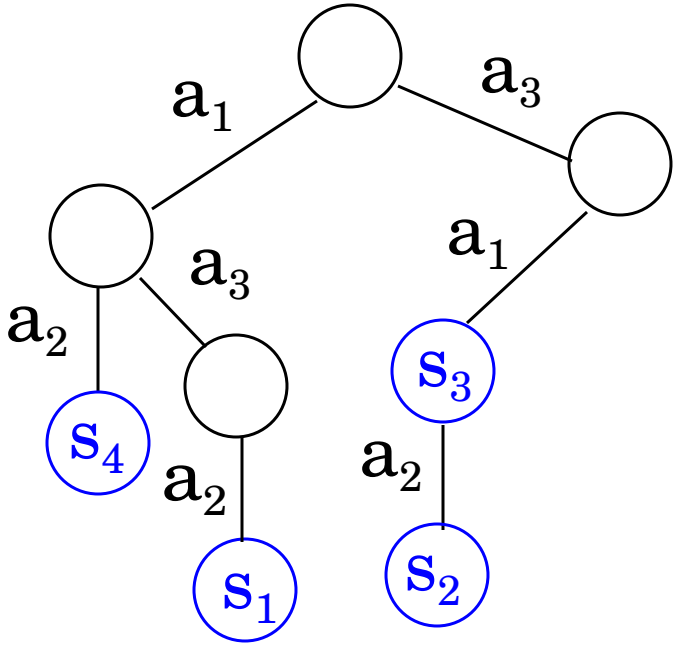


Árvore Digital de Prefixo

Uma árvore digital que representa uma codificação de prefixo é uma árvore digital de prefixo.



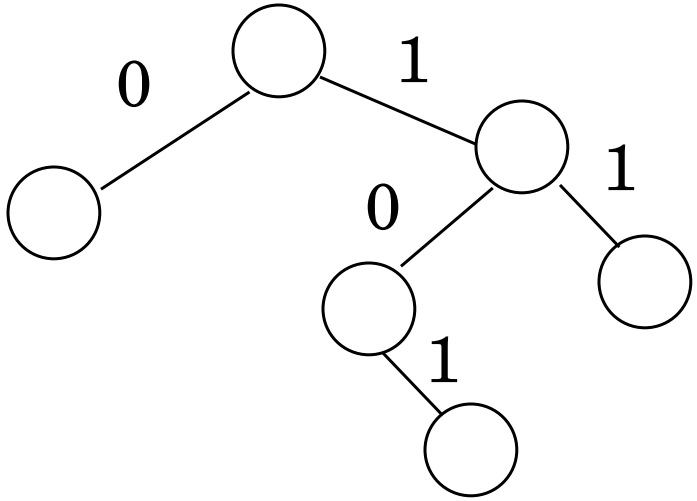
árvore digital de prefixo



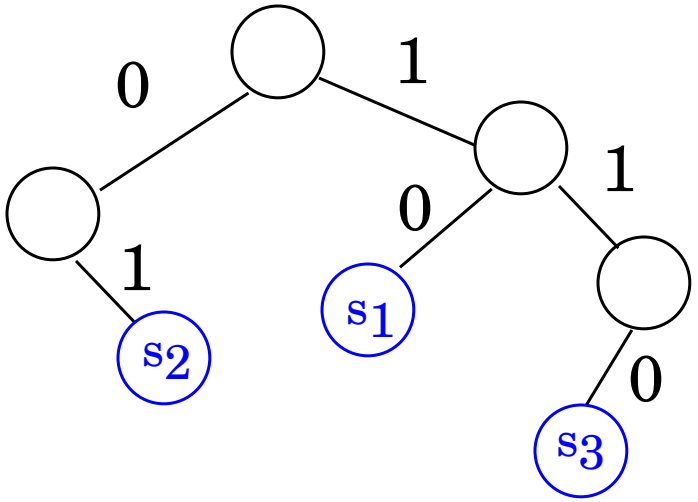
não é árvore digital de prefixo

Árvore Digital Binária

Uma árvore digital de alfabeto { 0, 1 } é uma árvore digital binária (m = 2).



Uma árvore digital binária de prefixo é uma árvore digital binária que é de prefixo.



árvore binária
de prefixo

Exercício

- Provar ou dar contra-exemplo.

- Uma árvore binária é de prefixo se e somente se todos os símbolos se localizam em folhas.

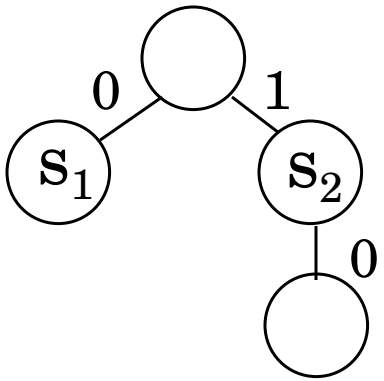
Tempo: 8 minutos

Solução

- Uma árvore binária é de prefixo se e somente se todos os símbolos se localizam em folhas.

- 1. árvore binária de prefixo ≠> todos os filhos se localizam em folhas

contra exemplo



- 2. todos os filhos se localizam em folhas => árvore binária de prefixo

Pois nenhum caminho raiz-folha pode ser subcaminho de outro caminho raiz-folha.

Códificação e Decodificação

- código de prefixo: significa codificação binária de prefixo
  texto: significa uma seqüência de símbolos

- Questões de interesse:
  codificar um texto
  decodificar um texto codificado

- A codificação e a decodificação podem ser facilmente realizadas através das árvores de prefixo.

Codificação e Decodificação

Exemplo:

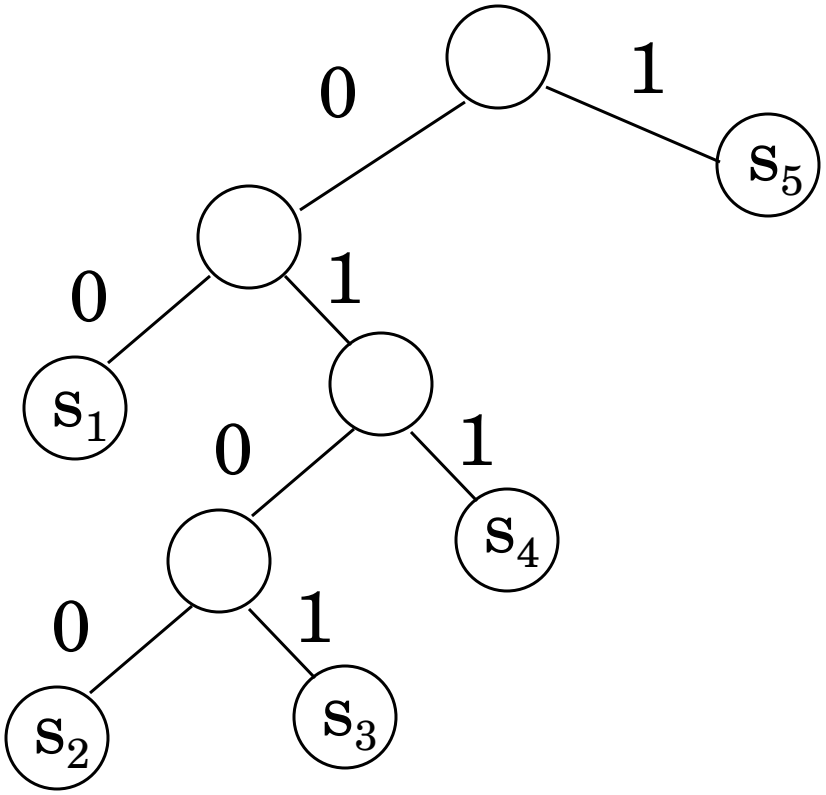
s_1: 00

s_2: 0100

s_3: 0101

s_4: 011

s_5: 1



$s_2\ s_1\ s_1\ s_4\ s_1\ s_5$ ⟵ codificação ⟶ 01000000011001 decodificação

Codificação e Decodificação

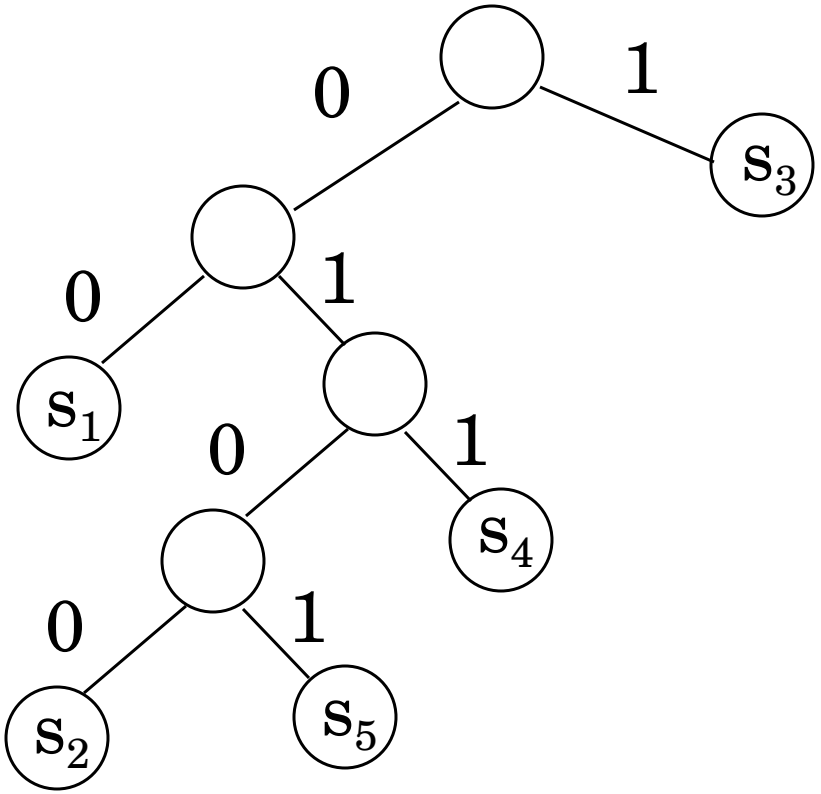
- Complexidade: a codificação e a decodificação ambas podem ser realizadas em tempo linear no tamanho total da mensagem codificada.
- Vantagens:
  - simplicidade
  - eficiência
- Desvantagens:
  - a troca de apenas um bit na mensagem pode ocasionar um erro considerável

Questão Central

- Dado um conjunto de símbolos e um texto, deseja-se atribuir um código binário (de prefixo) aos símbolos, de modo a minimizar o tamanho do texto codificado.

- Se T é uma árvore binária de prefixo correspondente à codificação de um certo texto, o tamanho do texto codificado é o custo de T, denotado por c(T).

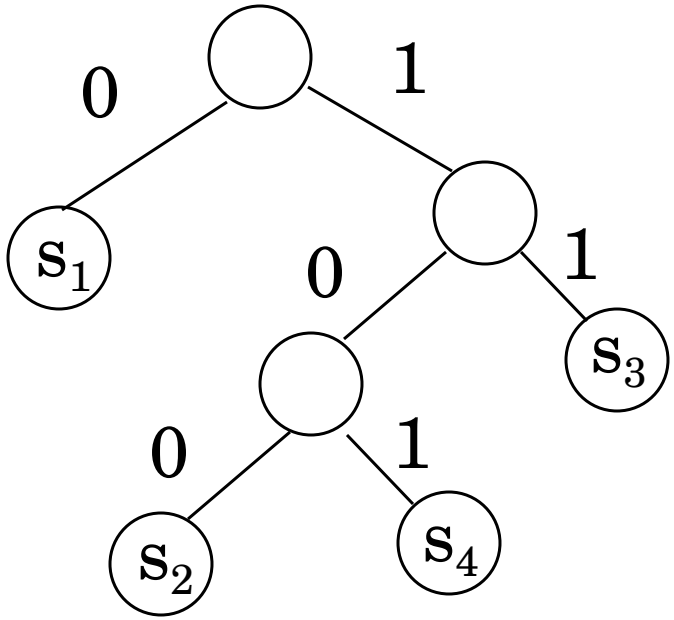
Exemplo



Texto: $s_2\ s_1\ s_1\ s_4\ s_1\ s_3$

Texto codificado:
01000000011001

c(T) = 14



Texto: $s_2\ s_1\ s_1\ s_4\ s_1\ s_3$

Texto codificado:
10000101011

c(T) = 11

Árvore de Huffman

- Uma árvore (binária) de prefixo que possui custo mínimo para um certo texto é uma árvore de Huffman para aquele texto.

- Obs.: o que é relevante, em um texto, é a freqüência f_i

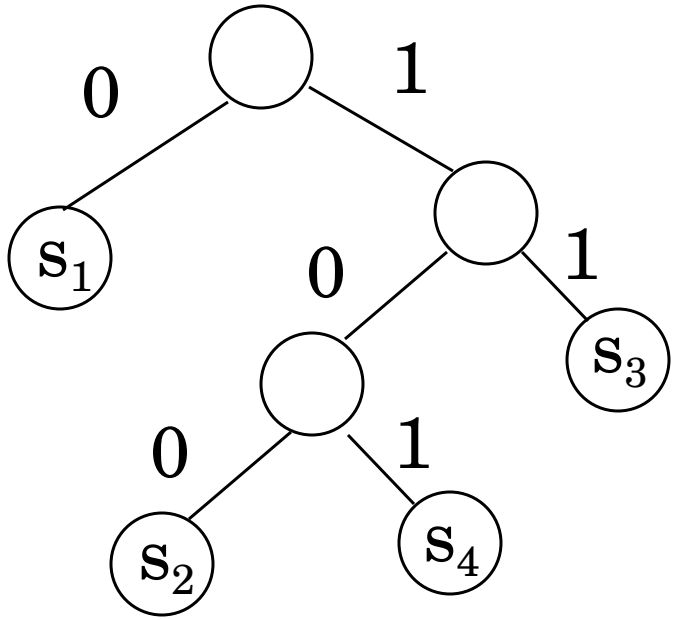
  com que cada símbolo s_i aparece no texto.

- Logo, a árvore de Huffman está sempre associada a um conjunto de símbolos, cada qual com sua freqüência.

Exemplo

Exemplo:

i	1	2	3	4
f_i	3	1	1	1



s_1 : 0

s_2 : 100

s_3 : 11

s_4 : 101

Árvore de Huffman

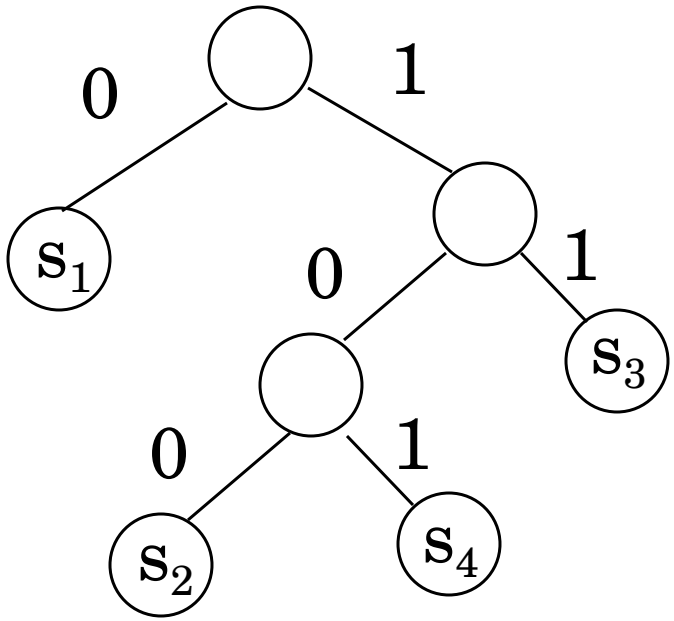
Árvore de Huffman

- Seja dado um texto, onde cada símbolo s_i aparece f_i vezes.
- Seja T uma árvore de prefixo correspondente ao texto dado.
- Seja l_i o comprimento do código binário de s_i.
- O produto $f_i.l_i$ representa a contribuição de s_i no custo c(T).
- Então:

$$c(T) = \sum_{1 \le i \le n} f_i \, . \, l_i$$ = comprimento do caminho externo ponderado de T

Exemplo

i	1	2	3	4
f_i	3	1	1	1



$s_1 : 0 \quad l_1 = 1$

$s_2 : 100 \quad l_2 = 3$

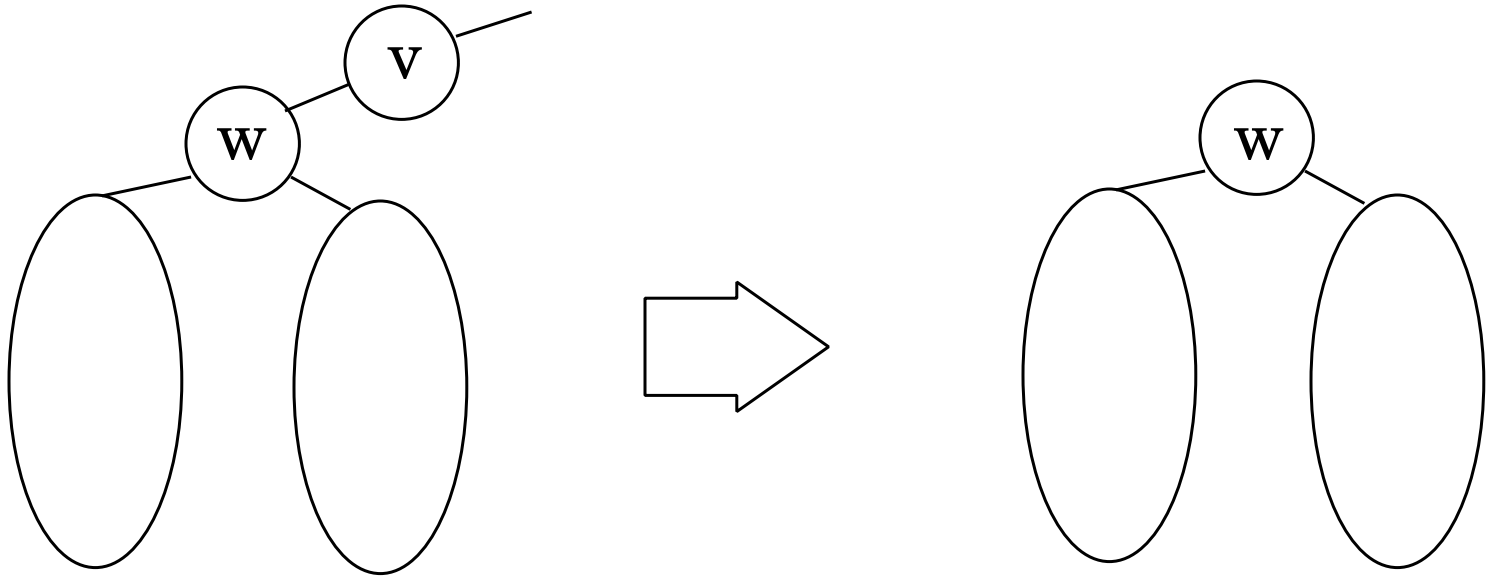
$s_3 : 11 \quad l_3 = 2$

$s_4 : 101 \quad l_4 = 3$

$\Rightarrow c(T) = f_1l_1 + f_2l_2 + f_3l_3 + f_4l_4 = 3.1 + 1.3 + 1.2 + 1.3 = 11$

Árvores de Huffman e Árvores Binárias

- Toda árvore de Huffman é estritamente binária.
- Porque se a árvore não for estritamente binária, há um vértice v com um único filho w. A identificação de v e w produziria uma árvore de custo menor.



- Toda árvore estritamente binária é arvore de Huffman para algum conjunto de freqüências de símbolos.

Exercício

- Provar ou dar contra-exemplo:

- Sejam s_i e s_j dois símbolos e l_i e l_j os respectivos níveis das folhas correspondentes a s_i, s_j em uma árvore de Huffman. Então $f_i < f_j \Rightarrow l_i \geq l_j$

Tempo: 5 minutos

Solução

- Sejam s_i e s_j dois símbolos e l_i e l_j os respectivos níveis das folhas correspondentes a s_i, s_j em uma árvore de Huffman.

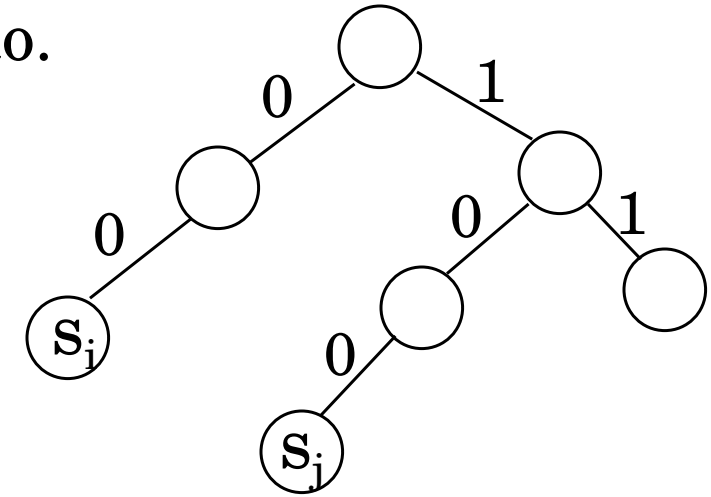
  Então $f_i < f_j \Rightarrow l_i \geq l_j$

- Prova: a afirmativa é verdadeira. Pois, caso contrário,

  $f_i < f_j \Rightarrow l_i < l_j$. Então $f_i.l_j + f_j.l_i < f_i.l_i + f_j.l_j$.

  Isto significa que a troca de posições entre s_i e s_j produziria uma árvore de custo menor, uma contradição.

$l_i < l_j$



Árvore de Huffman

- Para construir uma árvore de Huffman para um dado conjunto de caracteres, utiliza-se um método denominado algoritmo guloso.

- O algoritmo guloso procura resolver problemas de otimização através de um processo iterativo. A solução final do problema é obtida acrescentando-se um elemento, em cada iteração, à solução parcial até então obtida.
O critério para a escolha desse elemento a ser adicionado é otimizar, localmente, a solução parcial, desde que obedeça às condições de viabilidade definidas pelo problema.

Características

- O algoritmo guloso quando resolve o problema geralmente é eficiente.
- A solução final é obtida, passo a passo, a partir das soluções parciais.
- Um elemento se for adicionado à solução parcial, necessariamente será parte da solução final.
- Para conhecer se a solução fornecida pelo algoritmo guloso, de fato, resolve o problema é necessário formular uma prova matemática de sua correção.