Estrutura de Dados: Trabalho 2

A **codificação de Huffman** é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo. Ele foi desenvolvido em 1952 por David A. Huffman que era, na época, estudante de doutorado no MIT, e foi publicado no artigo "A Method for the Construction of Minimum-Redundancy Codes". É um dos métodos mais usados para compressão de dados e arquivos.

Uma árvore binária completa, chamada de árvore de Huffman é construída recursivamente a partir da junção dos dois símbolos de menor probabilidade, que são então somados em símbolos auxiliares e estes símbolos auxiliares recolocados no conjunto de símbolos. O processo termina quando todos os símbolos forem unidos em símbolos auxiliares, formando uma árvore binária. A árvore é então percorrida, atribuindo-se valores binários de *1* ou *0* para cada aresta, e os códigos são gerados a partir desse percurso.

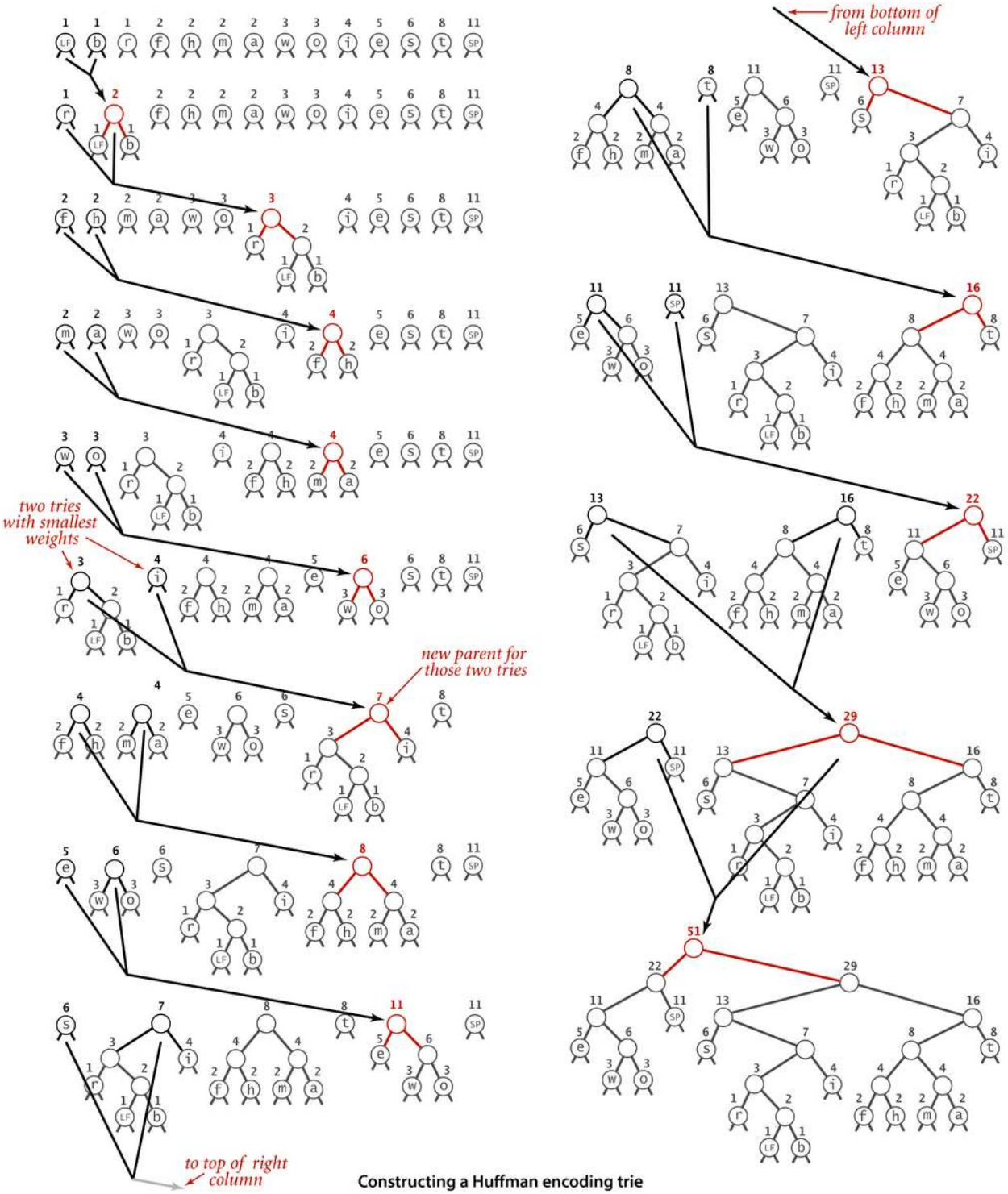
Algoritmo: O segredo do algoritmo de Huffman está na maneira de escolher a tabela de códigos.

- A tabela de códigos de Huffman não é universal. Ela é construída sob medida para a string a ser codificado de tal modo que o comprimento da cadeia codificada seja o menor possível (quando comparado com outros códigos livres de prefixos).
- Primeiro, construímos a árvore do código, depois extraímos dela a tabela de códigos. Dada a string a ser codificada, a árvore é construída assim:
    1. determine a frequência (ou seja, o número de ocorrências) de cada caractere da string original,
    2. para cada caractere, crie um nó (por exemplo o nó *x*) e armazene o caractere e sua frequência nos campos ch e freq (*x.ch* e *x.freq*);
    3. organize o conjunto de nós em uma árvore conforme o seguinte algoritmo.
        - Algoritmo de construção da árvore do código:
        - no início de cada iteração temos um conjunto de árvores mutuamente disjuntas (no início da primeira iteração, cada árvore tem um único nó);
        - escolha duas árvores cujas raízes, digamos *x* e *y*, tenham freq mínima (as duas mínimos frequências, se tiver empate escolher a árvore de menor tamanho, se mesmo assim tiver empate, não faz diferença a árvore a ser escolhida entre as que se encontram empatadas);
        - crie um novo nó pai (*parent)* e faça com que *x* e *y* sejam filhos desse nó;
        - faça *parent.freq* igual a *x.freq* + *y.freq*;
        - repita esse processo até que o conjunto de árvores seja reduzido a uma única árvore.
        - O código de Huffman de cada caractere, é obtido seguindo o caminho da árvore. Onde filho da direita deve ser atribuído o bit 1 e a esquerda o bit 0 (ou vice-versa).

EXEMPLO: Considere a string "*it was the best of times it was the worst of times*" (sendo LF o caractere 10, ou 0A em hexadecimal, que indica fim de linha) e SP espaço (32 ou 20 em hexadecimal). As frequências dos caracteres são

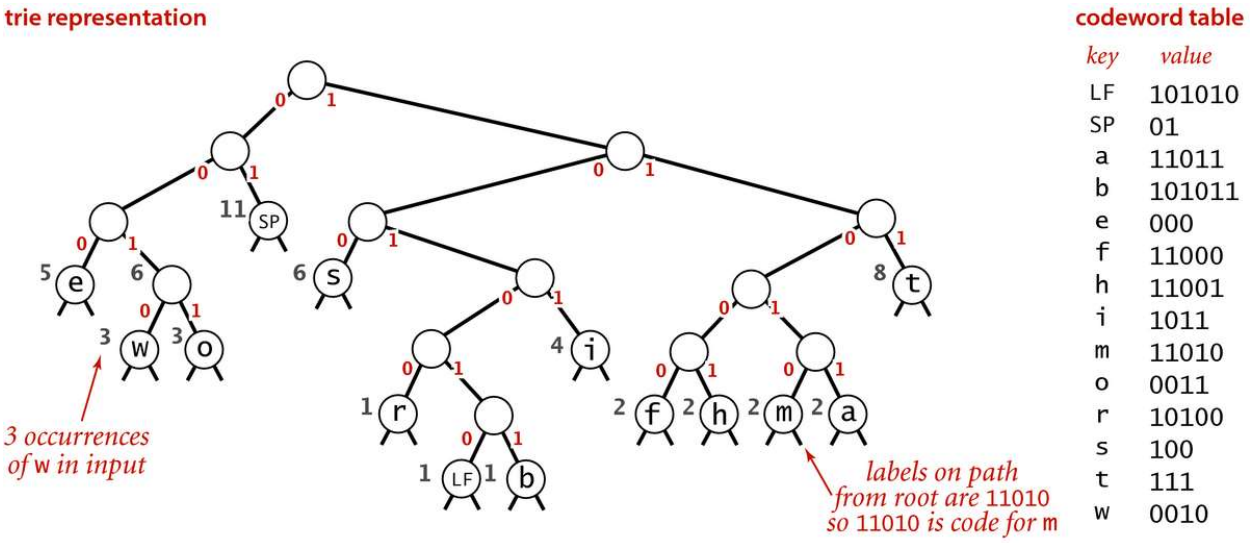
a	b	e	f	h	i	m	o	r	s	t	w	LF	SP
2	1	5	2	2	4	2	3	1	6	8	3	1	11

A construção da árvore de Huffman fica:



Constructing a Huffman encoding trie

O tabela com os códigos de Huffman fica:



Huffman code for the character stream "it was the best of times it was the worst of times LF"

Como parte 1 do trabalho 2 da disciplina implemente um programa chamado "comprime" que recebe por argumento o nome de um outro arquivo de TEXTO (ex.: ./comprime myfile.txt). O programa deve construtir a arvore de huffman utilizando uma árvore encadeada. Deve dar como saida dos arquivos tabela.txt e myfile.huff. O primeiro é um arquivo de TEXTO deve mostrar os cógidos de huffman para cada caractere do arquivo original. O segundo é um arquivo BINARIO onde deve ser a versão comprimida do arquivo original, isto é, subtituir cada caractere de código ASCII pelo código de Huffman (note que os códigos de huffman tem tamanho váriavel, logo vc deve trabalhar com BITS, para isso recomenda-se criar um buffer em memória que seja multtiplo de 1 BYTE – char, int, short int, etc. – usar operações de SHIFTS e lógicas para ir "escrevendo" os bits corretos no buffer e quando o mesmo tiver cheio escrever no arquivo).

Como parte 2 do trabalho, elabore um programa chamado "descomprime" que recebe dois argumento (ex.: ./ descomprime tabela.txt myfile.huff) e da como saída um arquivo de texto que deve ser a versão descomprimida de myfile.huff.