

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**

**CENTRO TECNOLÓGICO**

**DEPARTAMENTO DE INFORMÁTICA**

**Alan Herculano Diniz**

**Rafael Belmock Pedruzzi**

**Compactador de Arquivos: Trabalho para a disciplina de Estrutura de Dados I**

**Vitória**

**2018**

**Alan Herculano Diniz**

**Rafael Belmock Pedruzzi**

**Compactador de Arquivos: Trabalho para a disciplina de Estrutura de Dados I**

Trabalho para a disciplina de Estrutura de Dados I do curso de Ciência da Computação da Universidade Federal do Espírito Santo.

**Professor(a):** Patrícia Dockhorn Costa

**Disciplina:** Estrutura de Dados I

**Turma:** INF09292

**Vitória**

**2018**

Sumário

[1 – Introdução 4](#__RefHeading___Toc2064_1859988581)

[2 – Implementação 5](#__RefHeading___Toc2066_1859988581)

[2.1 – Árvore 5](#__RefHeading___Toc2068_1859988581)

[2.2 – Lista Genérica 6](#__RefHeading___Toc121_1097127904)

# 1 – Introdução

Nos dias atuais, arquivos e dados possuem tamanho e complexidade muito grandes para serem enviados e recebidos sem nenhuma modificação. Portanto, é preciso de um método de compactar arquivos para que seu transporte seja viável e eficiente. A partir desse fato, têm-se a base deste segundo trabalho da disciplina de Estrutura de Dados I, em que o objetivo é programar um compactador na linguagem de programação C que possa criar uma versão compactada de um arquivo de entrada e que possa descompactar arquivos compactados.

Para criar esse compactador, será utilizado o algoritmo de Huffman, que é um método que usa o número de ocorrências dos bytes e suas probabilidades de ocorrerem para criar um arquivo com tamanho reduzido.

# 2 – Implementação

Para utilizar o algoritmo de Huffman, é necessário implementar certos tipos abstratos de dados que auxiliaram na construção do compactador. Nos próximos subitens, serão listados e explicados esses TAD’s.

## 2.1 – Árvore

O tipo árvore é um dos principais tipos utilizados na construção do compactador, já que ele é utilizado para codificar os bytes de tal forma que será possível construir o arquivo compactado. Ela possui os campos de ponteiros para as subárvores da direita e da esquerda, já que é uma árvore binária, outro para manter o dado e outro para manter a ocorrência desse dado.

Também são definidas funções que manipulam variáveis do tipo árvore, entre as principais, estão as seguintes:

Funções de criação e destruição na memória: essas funções são responsáveis por alocar dinamicamente e apagar da memória o espaço com o tamanho necessário para uma árvore. Na verdade, todo TAD possui essas funções, entretanto, a árvore possui um diferencial: ela possui duas funções de alocação dinâmica de espaço de memória, uma para criar um nó que possui subárvores e outra para criar uma nó folha (que não possui nenhuma subárvore). Note que somente nós folha possuem referência para dados do arquivo que será compactado, isso é importante para o algoritmo de Huffman (que será explicado mais a frente).

Também há funções que retornam os dados existentes em um nó de uma árvore (dado, número de ocorrências e subárvores). Novamente, todo TAD possui esse tipo de funções.

Porém, essas funções não são as únicas que são relacionadas com o tipo árvore. Existe uma função que verifica se um nó de uma árvore é ou não um nó folha, outra que verifica se um certo dado está ou não na árvore, outra que retorna uma lista com todos os nós que pertencem ao caminho entre o nó raiz e outro nó dado com entrada e uma função que imprime a árvore.

## 2.2 – Lista Genérica

A lista usada nessa implementação do código de Huffman é uma implementação genérica, ou seja, ela pode ser aplicada e reutilizada para qualquer tipo de dado. Foram implementados dois tipos de dados, um para a lista (o sentinela da lista, para ser mais específico) e outro para o item de uma lista, o qual aponta para o conteúdo desejado.

A lista possui os seguintes campos: ponteiros para o primeiro e o último item da lista, uma tag com o tipo de dado que a lista possui (para evitar que uma lista possua itens de tipos diferentes) e a quantidade de itens na lista. Enquanto isso, o item possui os seguintes campos: um ponteiro opaco para o conteúdo do item, uma tag com o tipo do conteúdo