UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MIKAELLA FERREIRA DA SILVA

LEONARDO DEORCE LIMA DE OLIVEIRA

**2º TRABALHO PRÁTICO:**

COMPACTADOR/DESCOMPACTADOR DE ARQUIVOS

VITÓRIA

2018

MIKAELLA FERREIRA DA SILVA

LEONARDO DEORCE LIMA DE OLIVEIRA

**2º TRABALHO PRÁTICO:**

COMPACTADOR/DESCOMPACTADOR DE ARQUIVOS

Trabalho apresentado à disciplina Estrutura de Dados I (INF09292) do Curso de Ciência da Computação da Universidade Federal do Espírito Santo no 2º semestre do ano 2018, como requisito para avaliação.

Orientador: Prof.ª Patrícia Dockhorn Costa

VITÓRIA

2018

**RESUMO**

.

Palavras-chave**:** .

**SUMÁRIO**

[1 INTRODUÇÃO 5](#_Toc531624296)

[2 IMPLEMENTAÇÃO 5](#_Toc531624297)

[2.1 Bibliotecas e funções 5](#_Toc531624298)

[2.1.1 lista.h 5](#_Toc531624299)

[2.1.1.1 struct lista (Lista) 5](#_Toc531624300)

[2.1.1.2 arv\_codif 5](#_Toc531624301)

[2.1.1.3 cria\_lista 5](#_Toc531624302)

[2.1.1.4 insere\_ordenado 6](#_Toc531624303)

[2.1.2 arv\_binaria.h 6](#_Toc531624304)

[2.1.2.1 struct arv (Arv) 6](#_Toc531624305)

[2.1.2.2 cria\_arv 6](#_Toc531624306)

[2.1.2.3 codigos 7](#_Toc531624307)

[2.1.2.4 cria\_cabecalho 7](#_Toc531624308)

[2.1.2.5 libera\_arv 7](#_Toc531624309)

[2.1.2.6 libera\_tab 7](#_Toc531624310)

[2.1.2.7 buscaChar 8](#_Toc531624311)

[2.1.2.8 retorna\_freq 8](#_Toc531624312)

[2.1.2.9 retorna\_id 8](#_Toc531624313)

[2.1.2.10 retorna\_char 8](#_Toc531624314)

[2.2 Compactador (compacta.c) 9](#_Toc531624315)

[2.3 Descompactador (descompacta.c) 9](#_Toc531624316)

[3 CONCLUSÃO 9](#_Toc531624317)

# INTRODUÇÃO

# IMPLEMENTAÇÃO

## Bibliotecas e funções

### lista.h

Biblioteca responsável pela manipulação da lista que auxilia na criação da árvore binária de codificação.

#### struct lista (Lista)

* Função: Estrutura para lista encadeada simples sem sentinela;
* Campos: Um ponteiro para uma estrutura de árvore binária Arv e um ponteiro para o próximo elemento da lista.

#### arv\_codif

* Função: Cria árvore de codificação a partir do vetor de frequências;
* Entrada: Vetor cujos índices são os bytes lidos no arquivo original e elementos sao frequências do byte ou índice;
* Saída: Árvore de codificação;
* Pré-condição: Nenhuma;
* Pós-condição: Nós alocados dinamicamente e lista liberada.

#### cria\_lista

* Função: Inicializa lista;
* Entrada: Nenhuma;
* Saída: Lista inicializada;
* Pré-condição: Nenhuma;
* Pós-condição: Nenhuma.

#### insere\_ordenado

* Função: Insere nó ou árvore de forma crescente com relação à frequência absoluta do byte;
* Entrada: Nó ou árvore a ser ordenado(a);
* Saída: Lista encadeada de árvores binárias ordenada;
* Pré-condição: Lista inicializada;
* Pós-condição: Célula da lista para nó ou árvore alocada e lista ordenada em ordem crescente com relação à frequência.

### arv\_binaria.h

Biblioteca responsável pela manipulação da árvore binária de codificação.

#### struct arv (Arv)

* Função: Estrutura para armazenar nó da árvore binária de codificação;
* Campos: Tipo de nó (interno ou folha), um byte único lido do arquivo original, a frequência com a qual este aparece e ponteiros para os filhos do nó.

#### cria\_arv

* Função: Cria nó de árvore binária Arv;
* Entrada: Campos da estrutura Arv;
* Saída: Nó de árvore binária Arv criado;
* Pré-condição: Nenhuma;
* Pós-condição: Nó alocado dinamicamente.

#### codigos

* Função: Cria tabela cujos índices representam bytes e o conteúdo de cada elemento é o código do byte no algoritmo. Cada elemento diferente de NULL é alocado dinamicamente;
* Entrada: Árvore binária de codificação, string auxiliar, matriz tabela e inteiro auxiliar para recursão;
* Saída: Nenhuma;
* Pré-condição: Árvore, string auxiliar e tabela inicializados;
* Pós-condição: Tabela criada e elementos relevantes da mesma alocados dinamicamente.

#### cria\_cabecalho

* Função: Cria maior parte do cabeçalho para arquivo compactado;
* Entrada: Arquivo no qual escrever cabeçalho, árvore binária de codificação;
* Saída: Nenhuma;
* Pré-condição: Arquivo de saída e árvore de codificação inicializados;
* Pós-condição: Parte do cabeçalho escrita no arquivo de saída.

#### libera\_arv

* Função: Libera árvore binária Arv;
* Entrada: Árvore binária Arv;
* Saída: Nenhuma;
* Pré-condição: Árvore não nula;
* Pós-condição: Árvore liberada.

#### libera\_tab

* Função: Libera tabela de códigos;
* Entrada: Tabela de códigos;
* Saída: Nenhuma;
* Pré-condição: Tabela inicializada;
* Pós-condição: Elementos da tabela liberados.

#### buscaChar

* Função: Auxilia no processo de descompactação escrevendo bytes a partir de seu código;
* Entrada: Nó de árvore binária Arv, direção (0 para esquerda e 1 para direita);
* Saída: Nó na direção fornecida a partir do nó fornecido;
* Pré-condição: Árvore não nula;
* Pós-condição: Nenhuma.

#### retorna\_freq

* Função: Retorna frequência absoluta no arquivo original do byte armazenado no nó de uma árvore binária Arv;
* Entrada: Nó de árvore binária Arv;
* Saída: Frequência absoluta do byte referente ao nó no arquivo original;
* Pré-condição: Nó inicializado;
* Pós-condição: Nenhuma.

#### retorna\_id

* Função: Retorna tipo de nó;
* Entrada: Nó de árvore binária Arv;
* Saída: Campo identificador do nó;
* Pré-condição: Nó inicializado;
* Pós-condição: Nenhuma.

#### retorna\_char

* Função: Retorna caractere representando byte armazenado no nó de uma árvore binária Arv;
* Entrada: Nó de árvore binária Arv;
* Saída: Campo que contém caractere do nó;
* Pré-condição: Nó inicializado;
* Pós-condição: Nenhuma.

## Compactador (compacta.c)

## Descompactador (descompacta.c)

# CONCLUSÃO

**BIBLIOGRAFIA**