




 master ▼

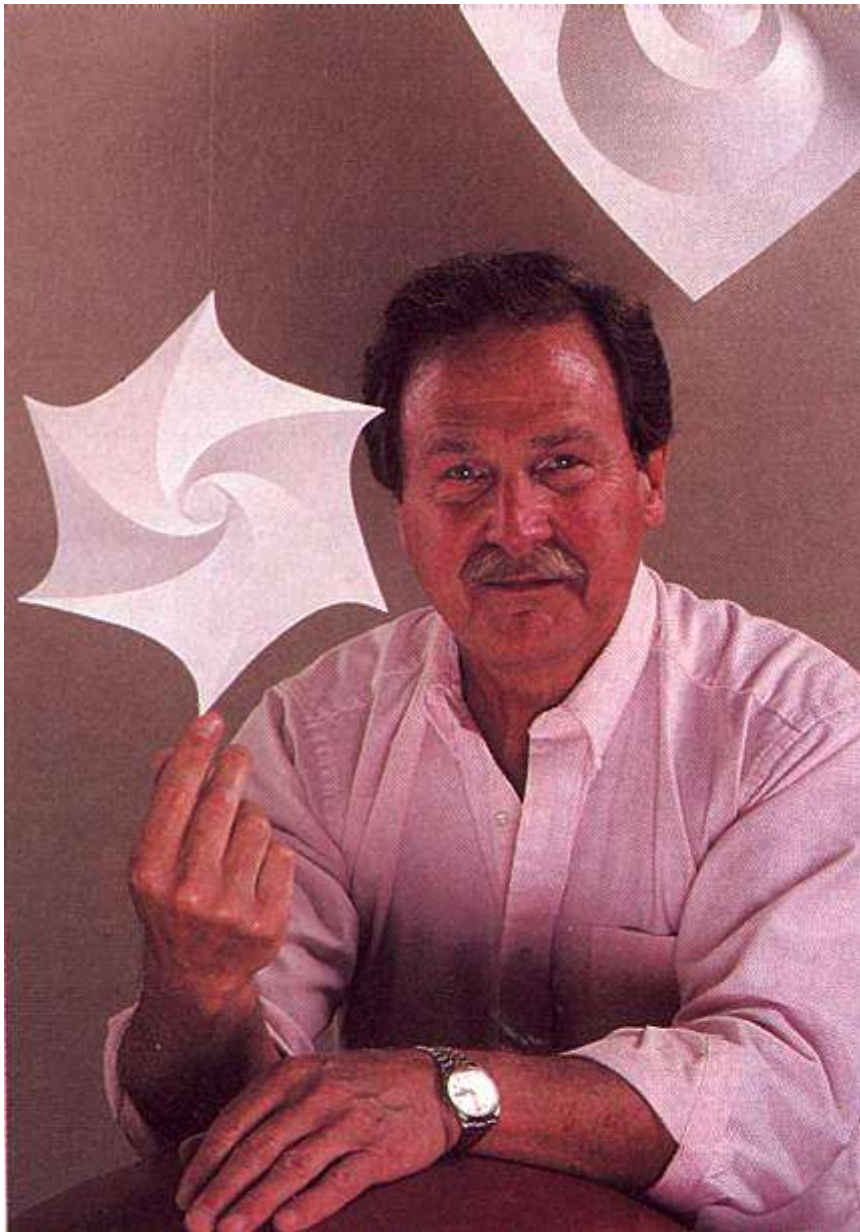
...

**huffman** / README.MD

 **danilojoi** README.MD - Versão 22/06/2021 - 18:56h 

 1 contributor

 129 lines (90 sloc) | 6.91 KB 



## Codificação de Huffman:

David Albert Huffman (9 de agosto de 1925 — Santa Cruz, 7 de outubro de 1999) foi um pioneiro no campo da ciência da computação.

Este método de compressão é um dos mais utilizados na atualidade para compressão de diversos tipos de mídias diferentes – na maioria das vezes, como componente de codecs mais complexos (Salomon, 2007) – apesar de ter sido inicialmente desenvolvido para compressão de texto (Pu, 2006).

David Huffman, que desenvolveu o método como projeto da disciplina de "Teoria da Informação" em 1950 (Blelloch, 2001), enquanto era aluno de Doutorado no MIT (Sayood, 2006) (Wikimedia Foundation, Inc., 2016), e publicou as suas conclusões em 1952 no artigo "A Method for the Construction of Minimum-Redundancy Codes" na revista *Proceedings of the I.R.E.* (Huffman, 1952).

A codificação de Huffman é baseada na "Teoria de Informação" de Shannon (Shannon, 1948), onde em um elemento de informação (texto, imagem, áudio, etc.) representado por símbolos, alguns desses símbolos ocorrem mais vezes que outros. Se representarmos esses símbolos mais comuns utilizando códigos menores (usando menos bits), por exemplo; iremos obter uma codificação binária com menos bits, como resultado de uma diminuição do comprimento médio (em bits) de cada código (Huffman, 1952) (Pu, 2006). O algoritmo de Huffman é, portanto, baseado na categoria de Codificação de Entropia – a informação é encarada como uma sequência de símbolos genéricos, menosprezando a semântica dos mesmos -, sendo um modo de codificação sem perdas – o código resultante é totalmente reversível; ou seja, a sua decodificação resulta num fluxo de dados exactamente igual ao fluxo de dados de origem (Ribeiro, Apontamentos das Aulas, 2016).

## Pré-Requisitos:

---

- [Ambiente linux](#)
- [Compilador GCC](#)
- [Makefile](#)

Também pode ser executado em um ambiente Windows, desde que se tenha uma versão do gcc instalado. Neste caso, o MakeFile não funcionará, e um arquivo .bat deverá ser criado.

IDE's também podem executar esta aplicação sem nenhum problema. Tais como:

- [Code:Blocks](#)
- [DevC++](#)
- [Visual Studio Code](#)

Esta aplicação foi desenvolvida e testada em um ambiente Linux Ubuntu, com testes também realizados em ambiente Windows com a IDE CodeBlocks.

## Instalação e Execução:

---

Nenhuma instalação adicional é requerida. Apenas os prés-requisitos citados acima.

A única configuração utilizada para o ambiente Linux, foi o Makefile para simplificar o processo de compilação.

## Gerando o Arquivo Binário:

- Execute o comando a seguir, para gerar o arquivo binário:

```
$~ Makefile
```

## Executando a Aplicação:

- Caso o comando Makefile tenha funcionado corretamente, a aplicação poderá ser executada da seguinte maneira:

```
$~ ./nome_da_aplicacao [Parâmetro 1] [Parâmetro 2] [Parâmetro 3]
```

Onde - Compressão:

- Parâmetro 1 = "-c" Comprime o Arquivo
- Parâmetro 2 = Nome do Arquivo a Ser Comprimido (entrada)
- Parâmetro 3 = Nome do Arquivo a Ser Gerado (saída)

Onde - Descompressão:

- Parâmetro 1 = "-d" Descomprime o Arquivo
- Parâmetro 2 = Nome do Arquivo a Ser Descomprimido (entrada)
- Parâmetro 3 = Nome do Arquivo a Ser Gerado (saída)

Exemplo:

```
$~ ./huffman -c arquivo_origem.extensão arquivo_destino.hx
```

NOTA: Por padrão, o arquivo compactado terá a extensão ".hx"

## Características do Código Fonte:

---

A aplicação foi desenvolvida utilizando a técnica de TDA (Tipo de Dado Abstrato), onde existem métodos públicos e privados, interligados através de "interfaces".

Ela implementa uma separação de especificação e implementação, que permite o uso do TAD sem conhecer nada sobre a sua implementação. Portanto, um TAD pode ter mais de uma implementação.

A aplicação em si, utiliza a ideia de uma árvore binária, para a implementação da codificação de Huffman.

## Vantagens do uso do TAD:

Mais seguro programar: apenas as operações do Tipo Abstrato de Dados alteram os dados. Maior independência e portabilidade de código e Manutenção: alterações na implementação de um TAD não implicam em alterações em seu uso.

## Métodos Públicos:

```
#include "huffman_privado.h"
#include <stdio.h>

// Cria um novo nó/lista
nodeLista *novoNodeLista(nodeArvore *nArv);

// Cria um novo nó/árvore
nodeArvore *novoNodeArvore(byte c, int frequencia, nodeArvore *esquerda, nodeArvc

// Insere um nó na lista
void insereLista(nodeLista *n, lista *l);

// Função auxiliar utilizada na geração da árvore, para determinar o menor valor
nodeArvore *popMinLista(lista *l);

// Determina a "frequência" em que um caracter é encontrado dentro do
// arquivo passado como parâmetro de entrada
void getByteFrequency(FILE *entrada, unsigned int *listaBytes);

// Função auxiliar, que lê os "códigos" binários armazenados na árvore, antes da
int pegaCodigo(nodeArvore *n, byte c, char *buffer, int tamanho);

// Cria a árvore de Huffman (faz a montagem)
nodeArvore *BuildHuffmanTree(unsigned *listaBytes);

// Faz a "limpeza" (libera) a árvore binária.
void FreeHuffmanTree(nodeArvore *n);

// Faz a leitura de um "caracter" a partir de uma posição específica de um
// arquivo de entrada, e o converte em
int geraBit(FILE *entrada, int posicao, byte *aux );

// Função que imprime uma mensagem de erro, quando o arquivo passado como parâmet
// não é localizado
void erroArquivo();

// Função que realiza a compressão do arquivo
void CompressFile(const char *arquivoEntrada, const char *arquivoSaida);

// Função que realiza a descompressão do arquivo
void DecompressFile(const char *arquivoEntrada, const char *arquivoSaida);
```

## Referências:

- Salomon, D. (2007). Data Compression (4th ed.). London: Springer.
- Pu, I. M. (2006). Fundamental Data Compression. Oxford, United Kingdom: Elsevier.

- Blelloch, G. E. (2001). Introduction to Data Compression. In G. E. Blelloch, Algorithms in the Real World. Pittsburg, PA, United States of America: Carnegie Mellon University.
- Sayood, K. (2006). Introduction to Data Compression. San Francisco: Morgan Kauffman.
- Wikimedia Foundation, Inc. (30/03/2016). Huffman Coding. Acessado em 21/06/2021, de Wikipedia, the free encyclopedia: [https://en.wikipedia.org/wiki/Huffman\\_coding](https://en.wikipedia.org/wiki/Huffman_coding)
- Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the I.R.E., 1098-1101.
- Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27, 379–423 & 623–656.
- Ribeiro, N. (2016). Aparentamentos das Aulas. Multimédia II. Porto: Universidade Fernando Pessoa.