Relatório T4 - INF1010 - Estrutura de Dados Avançados

Componentes do grupo:

Diego Miranda -2210996

Eric Goulart - 2110878

**INTRODUÇÃO:**

Neste trabalho fizemos um programa de compactação e descompactação de arquivos tendo como base o algoritmo de Huffman. Pegar uma arquivo txt, contamos o numero de ocorrências de cada caracter e com base nisso criamos a Árvore de Huffman. Uma vez com essa árvore nós vamos percorrendo ela e encontrando uma recodificação em bytes para cada chave.

**DESCRIÇÃO:**

Primeiro definimos as duas estruturas. A struct **“no**” representa um nó da árvore de Huffman, enquanto a struct “**vet**” representa o vetor de frequências de cada caracter e a struct dic que pega o carácter, seu respectivo código e o tamanho deste código.

Nossas funções são:

* **cria\_o\_vetor**

Cria uma lista encadeada ordenada pelo número de ocorrências. Recebe um elemento do vetor de ocorrências e um ponteiro para a lista encadeada.

* **arvoriza**

Constrói a árvore de Huffman a partir da lista encadeada.

* **imprimearv**

Função para percorrer a árvore de Huffman em pré-ordem e imprimir seus elementos.

* **atribuiCodigos**

Função recursiva que percorre a árvore de Huffman e vai atribuindo códigos binários a cada caracter.

* **contaletras**

Responsável por saber o número de caracteres no texto, utilizada para conseguir alocar dinamicamente a memória a ser utilizada pelo vetor dicionário.

* **calculacaminho**

Responsável por ir percorrendo a árvore em busca de uma folha, e enquanto isso grava o percurso percorrido. Ao encontrar uma folha, insere no vetor o caractere e o caminho percorrido até aquele momento, assim dando o dicionário para cada caractere utilizado.

* **compactar**

Essa função vai lendo o arquivo txt e achando o código equivalente para cada caractere presente no arquivo. Com o código ele insere bit a bit no arquivo binário.

* **descompactar**

Esse arquivo lê bit a bit do arquivo compactado e verifica se existe um código igual na tabela com tamanho igual, caso ele encontre ele insere o caractere no arquivo txt e caso não ele adiciona o próximo bit e incrementa o tamanho.

* **main**

Aqui primeiro abrimos três arquivos: o arquivo de entrada com o texto a ser criptografado, o arquivo comprimido onde vai ser escrito os códigos em binário e o arquivo descomprimido onde vai ser escrita a mensagem em txt. Em seguida inicializamos um vetor para contar a ocorrências de caracteres e percorremos o arquivo de entrada atualizando o vetor. Depois utilizamos esse vetor para criar nós que logo em seguida são passados como parâmetro para a criação da árvore de Huffman. Calculamos um código para cada nó da árvore e armazenamos ele em um dicionário. O próximo passo é utilizar esse dicionário para compactar o arquivo.txt para arquivo.bin. Por fim, descompactar o arquivo binário para voltar a ter o texto original.

**OBSERVAÇÕES:**

Tivemos como maior dificuldade a compactação do arquivo, não conseguimos realizar com tanta facilidade a escrita em binária bit a bit (tentamos utilizar a biblioteca BitFile mas não conseguimos entender como ela funcionava) mas no final conseguimos fazer tudo funcionar.