Trabalho Prático III: Decifrando os Segredos de Arendelle

Eduardo de Paiva Dias
2018126657

Belo Horizonte – MG – Brasil

Introdução

O problema proposto dizia respeito em decifrar antigos códigos morse do Reino de Arendelle, mais precisamente o trabalho consiste em duas partes:

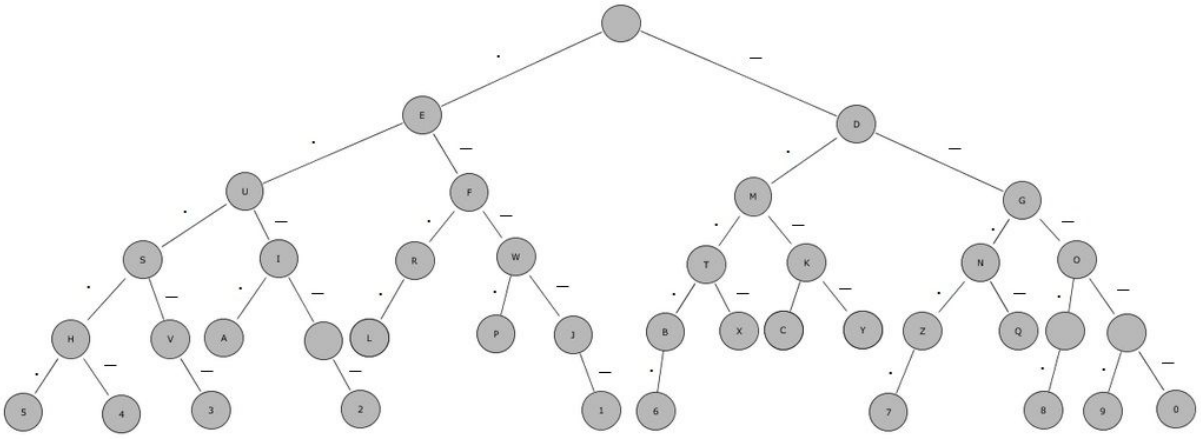
Construção da árvore: Seu programa deverá ler um arquivo contendo a definição do Código Morse (morse.txt) e construir a árvore que ser´a utilizada na pesquisa (decodificação dos caracteres em Morse).

Conversão das mensagens: Com a árvore construída, seu programa deverá ser capaz de converter mensagens em código Morse.

Implementação

Funcionamento geral do programa:

Primeiramente, para visualizar o funcionamento da árvore, foi criado a figura abaixo:



Dessa forma, podemos notar que há algumas lacunas vazias na árvore para se chegar a um determinado valor, com isso foi alterado o arquivo txt para que a árvore preenchesse essas lacunas. Ademais, o preenchimento dos nós da árvore binária foi usado recursividade.

Para a implementação de busca dos caracteres em morse correspondente a letras foi usado basicamente a mesma função recursiva usada para preenchimento da árvore binária, mas com algumas alterações.

As possíveis entradas para o funcionamento do programa seria:

./nomedoprograma
em que o usuário digita as entradas no próprio programa e obtém a saída pelo terminal.

./nomedoprograma < caso1.in > caso1.out,
em que as entradas são obtidas apartir do arquivo “caso.in” e a saída e coloca em “caso1.out”.

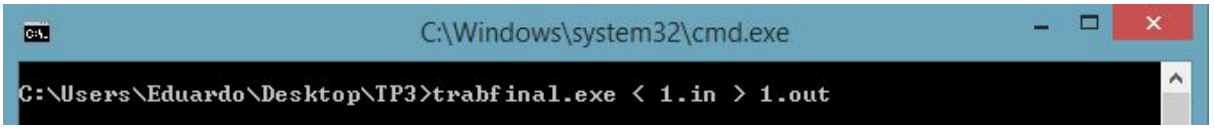
./nomedoprograma -a < caso1.in > caso1.out,
nesta entrada as entradas e a saídas vão ser colocadas em um arquivo, contudo a árvore será impressa em preordem e colocada no arquivo com seus correspondentes em morse.

Funcionamento das funções:

TipoRegistro, Tipo No, TipoDicionario, TipoApontador: Estruturas usadas na criação de árvore
GuardaMorse: Estrutura que guarda os itens da árvore antes de inseri-lo.
GuardaFraseMorse: Estrutura que guarda cada frase obtida do arquivo em morse.
GuardaFrase: Estrutura que contém a tradução de cada frase.
Inicializa(): Função que inicia a árvore
ler_arquivo_preencher_arvore(), readFile() ,Insere(), preenche_arvore(): Funções que leem o arquivo txt e preenche o que foi lido na struct e logo depois na árvore.
ler_entradas(): Lê as entradas do usuário ou do arquivo.
traduzir_morse(), BuscaLetra(): Traduz o morse e coloca em uma estrutura.
preOrdem(), procuraCaracter(): Imprime a árvore em pré ordem e procura o caracteres correspondente em morse.

Instruções de compilação e execução

O programa foi desenvolvido no sistema operacional windows na linguagem C e compilado pelo ambiente de desenvolvimento integrado de código aberto e multiplataforma livre Code::Blocks, contudo para executá-lo inseri o seguinte comando no terminal:



Entretanto, podemos executá-lo com qualquer comando citado na implementação.

Análise de complexidade

A **Análise de complexidade de tempo** não representa diretamente o tempo de execução, mas o número de vezes que certa operação relevante é executada. No caso deste algoritmo, temos a construção da árvore e a conversão de mensagens. Como as duas funções são basicamente iguais o melhor caso seria **O(1)** e o pior caso seria **O(n log n)**.

Análise de complexidade de espaço é a quantidade de espaço necessária para alocar os nós da árvore binária que é **O(n)**.

Conclusão

O trabalho foi concluído com êxito, tanto em questão da construção da árvore como na conversão das mensagens, entretanto tive algumas dificuldades ao fazer este trabalho prático, como redirecionamento de entrada e saída pelo terminal. Além disso a saída dos arquivos de teste não estavam formatadas o que dificultou verificar se minha saída condizia com a fornecida. Ademais, nem todas saídas ficaram iguais aos do teste, as minhas tinham em alguns casos mais frases e estavam bem formatadas. Consta no arquivo.zip todas minhas entradas e saidas.

Bibliografia

CHAIMOWICZ, Luiz. Árvores. Belo Horizonte, 2019/1.

CHAIMOWICZ, Luiz. Árvore Binária de Pesquisa sem Balanceamento. Belo Horizonte, 2019/1.