Trabalho Prático 3

Ariel Augusto dos Santos

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

ariel.santos@dcc.ufmg.br

1. Implementação

As TADs implementadas no projeto são árvore binária e fila, ambas armazenam chaves/valores no formato `char` que é utilizado através do projeto para armazenar caracteres e códigos de controle. Além destas estruturas e suas classes, a maior parte do estado e fluxo de execução do programa é gerenciado pela classe `Runner`, responsável pela interpretação do input, construção da árvore alfabética e codificação e decodificação das frases.

Há também um programa de teste, com um ponto de entrada separado em `/test/Test.cc`, que foi utilizado durante o desenvolvimento do Trabalho para verificar a codifição das frases realizada por `Runner::encode()`.

3. Compilação e execução

Além das classes e funções necessárias para implementação do Trabalho, que podem ser compiladas através do Makefile padrão & CMake, o programa de teste `/test/Test.cc` só pode ser compilado através da última ferramenta. Tanto o make quanto CMake irão gerar o executável `run.out` na pasta `/bin/`.

4. Análise de complexidade

4.1 Árvore Binária

A classe `BTree` implementa uma árvore binária de busca, com métodos para adicionar (`add_leaf()`), acessar (`get_node()`) e calcular o caminho até chaves (`get_path()`). No pior caso, de uma árvore completamente vertical no formato de uma lista, são necessários $O(n)$ acessos/comparações. O caso médio, aproximando uma árvore balanceada de altura $n \log(n)$, possui complexidade $O(n \log(n))$.

Em todos os casos, a complexidade de espaço é $\theta(n)$.

4.2 Fila

Além da árvore binária requerida pela especificação do Trabalho, também foi utilizada o TAD Fila, para armazenar os caminhos de chaves da árvore e outras sequências. A classe `Queue` possui métodos para inserção (`enqueue()`) e remoção de valores (`dequeue()`), de complexidade constante. Internamente `Queue` utiliza uma lista encadeada, com complexidade de espaço $\theta(n)$.