Trabalho – árvores AVL, rubro-negra e B

Artur Machado[1], **Daniella Vasconcellos**[1], **Gabriel Anselmo**[1], **Grasiela Ferreira**[1], **Kashimann Lehmkuhl**[1], **Victor Requia**[1]

[1]Departamento de Ciência da Computação – Universidade Estadual de Santa Catarina (UDESC)

artur.vinicius@gmail.com, daniellavasconc@gmail.com

anselmogabriel421@gmail.com, grasif03@gmail.com

kashimannlehmkuhl@gmail.com, victorrequia@gmail.com

Resumo. *Trabalho feito para a disciplina de Estrutura de Dados II (EDA0002), ministrada pelo Prof. Allan Rodrigo Leite. O trabalho se divide nas seguintes sessões: A discussão da geração dos códigos, a apresentação dos gráficos gerados, a execução do programa e o destaque do ambiente.*

1. Códigos

No enunciado do trabalho, lia-se:

O objetivo deste trabalho consiste em analisar a complexidade algorítmica das operações de adição de nós e balanceamento em árvores AVL, rubro-negra e B. A análise deve ser realizada considerando a geração de um conjunto de dados (chaves) com tamanho variando entre 1 e 1000. As chaves devem ser geradas prevendo o pior caso (chaves ordenadas crescente ou decrescente) e caso médio (chaves aleatórias). Para geração das chaves aleatórias, sugere-se o uso da função rand e srand em C considerando um tamanho de amostra de 10 conjuntos para validade estatística. O resultado final do experimento deve ser exibido em dois gráficos de linha (um para o pior caso e outro para o caso médio), onde o eixo X representa o tamanho dos conjuntos de dados (1 a 1000) e o eixo Y representa o esforço computacional das operações (adição de chaves e balanceamento). Cada gráfico deve apresentar 3 linhas, as quais representam as respectivas operações para cada estrutura de dados avaliada.

Portanto, para facilitar a geração e observação dos resultados, os arquivos foram separados para cada árvore, sendo eles: *ArvoreAVL.c*, *ArvoreB.c* e *ArvoreRubroNegra.c*. Cada um deles possui uma main que é a responsável por printar os resultados no terminal, os quais foram utilizados para a geração dos gráficos a serem mostrados na próxima sessão.

Para que os trabalhos possam ser compilados e os resultados exibidos (lembrando que a cada interação os resultados serão diferentes por conta das chaves aleatórias geradas), os seguintes comandos devem ser aplicados para cada projeto:

gcc arvoreAVL.c -o arvoreAVL
./arvoreAVL

gcc arvoreB.c -o arvoreB
./arvoreB

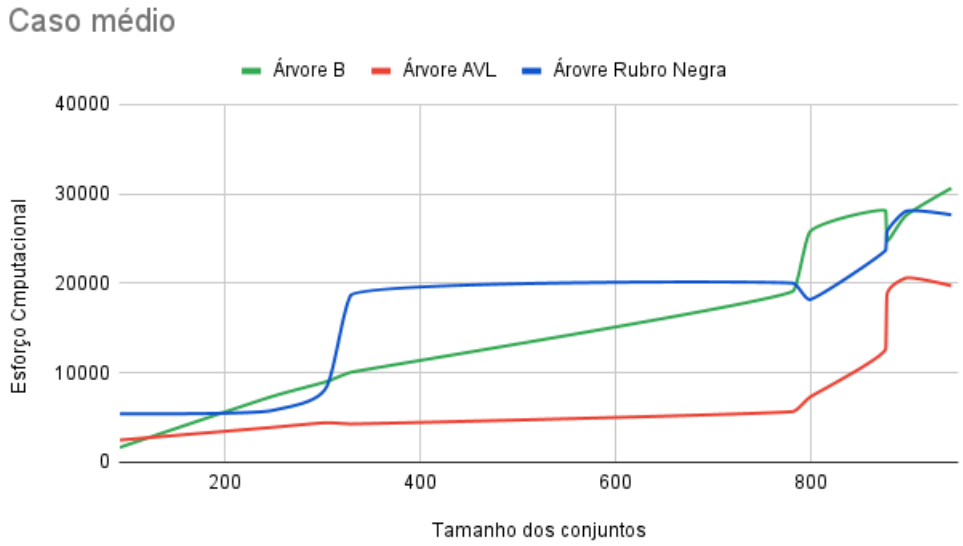
gcc arvoreRubroNegra.c -o arvoreRubroNegra
./arvoreRubroNegra

2. Gráficos

Esta sessão é direcionada para a apresentação dos gráficos gerados a partir dos códigos. É importante mencionar que por mais que o tamanho dos conjuntos tenha sido aleatoriamente criado, eles foram colocados em ordem crescente para melhor construção e análise.

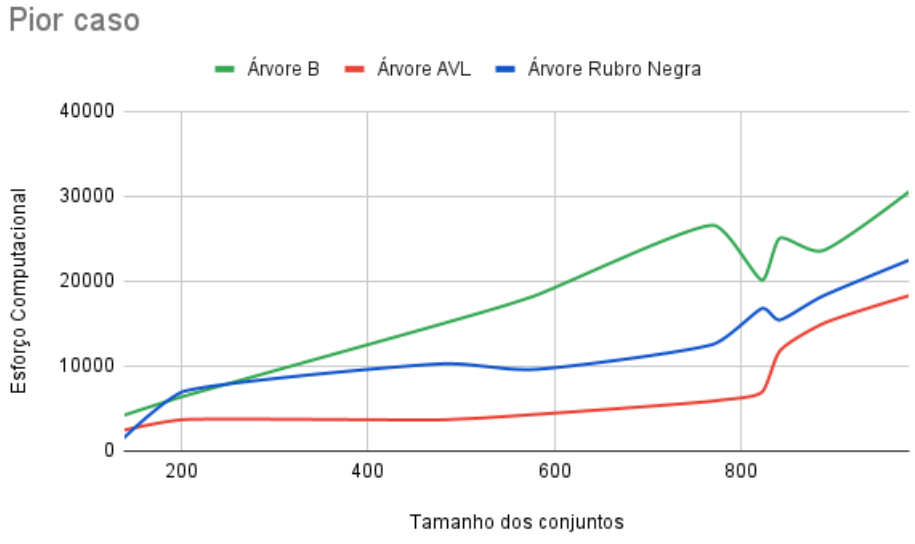
2.1. Caso médio

Aqui está o gráfico da geração dos resultados onde as chaves foram geradas de forma aleatória.



2.2. Pior caso

Aqui está o gráfico da geração dos resultados onde as chaves foram geradas de forma decrescente.



3. Destaque do ambiente

Esse projeto foi compilado utilizando um processador *Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz*, com o sistema operacional *Linux Ubuntu 20.04 focal*.

4. Considerações finais

Agradecimentos ao Professor Allan Rodrigo Leite pela disponibilização dos materiais utilizados como base para o estudo feito neste trabalho.