UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ANDRÉ SCHAIDHAUER LUCKMANN - Turma B
VITOR DA CUNHA PIMENTEL DA ROSA - Turma A

Disciplina: Estrutura de Dados

Análise Comparativa do Desempenho de Diferentes Árvores em Medição de Tempo em Jogos Digitais

Professores: Viviane Pereira Moreira e Dennis Giovani Balreira

Porto Alegre
2025

Contexto

1. Objetivo

O objetivo do trabalho é comparar o desempenho de diferentes estruturas de dados vistas na disciplina em uma aplicação de estimativa de tempo de jogo com base em dados da plataforma Steam.

2. Aplicação

A aplicação consiste em um programa escrito na linguagem C que recebe os seguintes parâmetros:

1. **Dataset:** Um arquivo .csv que armazena nomes de jogos associados ao tempo médio que os jogadores jogaram na Steam.
2. **Lista de Jogos:** Um arquivo .txt que armazena um nome de jogo por linha. Indica uma lista de desejos de jogos do usuário.
3. **Arquivo de saída:** Um caminho para um arquivo .txt que irá conter a estimativa de tempo para jogar os jogos desejados e estatísticas das buscas nas árvores.

A aplicação lê o arquivo **Dataset** e constrói três tipos de árvores, inserindo os dados de cada jogo na ordem dada no arquivo **Dataset.** As árvores escolhidas para este trabalho são:

1. **Árvore Binária de Pesquisa (BST);**
2. **Árvore AVL (AVL);**
3. **Árvore Rubro-Negra (RBT).**

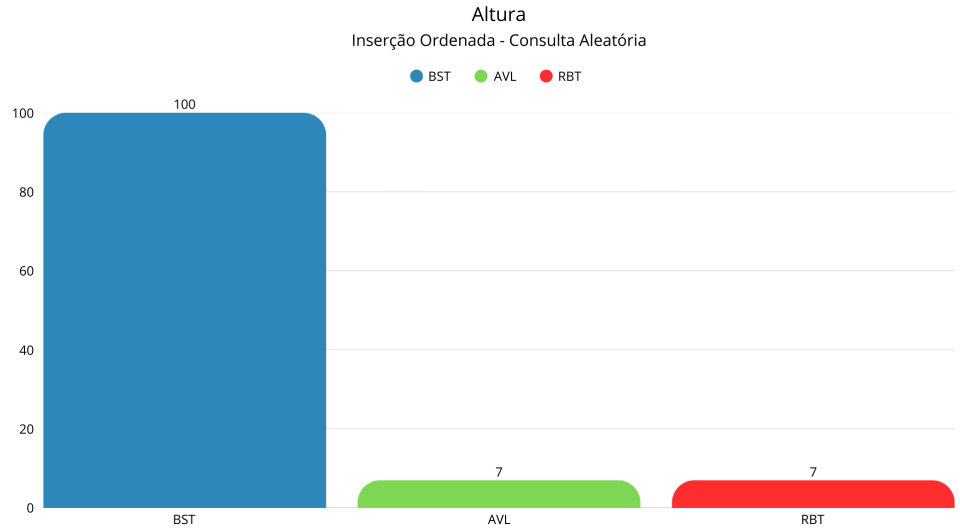
Exemplo de chamada do código:

```
./measurer.o dataset/dataset.csv dataset/player_list.txt output/output.txt
```

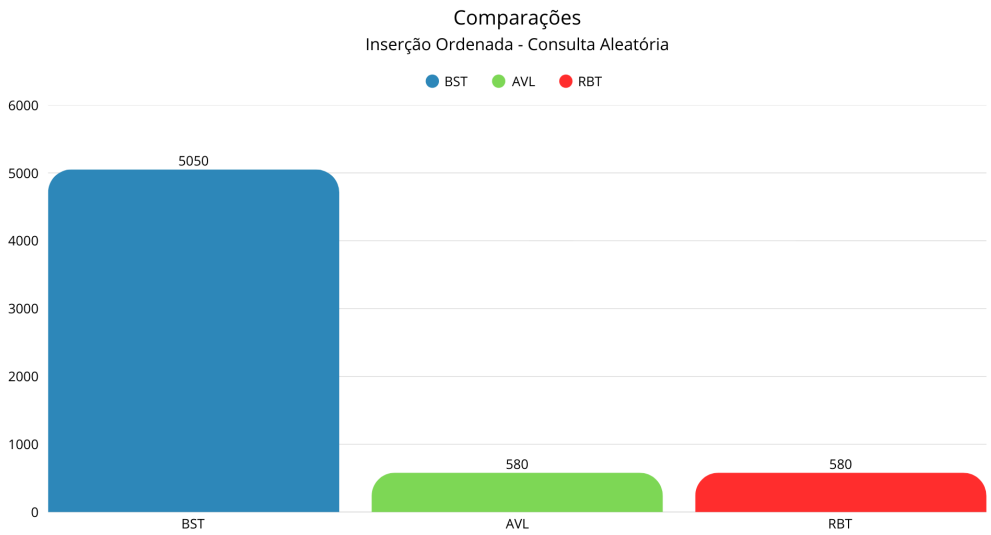
3. Análises de Desempenho

1. Inserção Ordenada - 100 Jogos

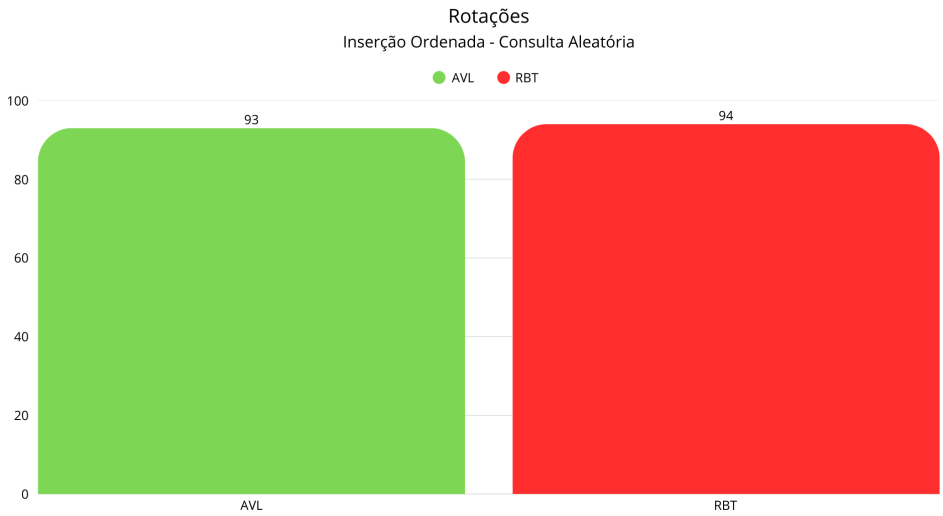
Neste caso, inserimos 100 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 100 jogos ordenados aleatoriamente. Os resultados foram os seguintes:



Nota: Observa-se que a BST ficou totalmente desbalanceada nesse caso, e como ela tem 100 nodos e a altura é 100, ela se comporta basicamente como uma lista encadeada, o que é muito ineficiente para consulta neste caso. Também observamos que tanto a AVL quanto a RBT tiveram a mesma altura nesse caso.



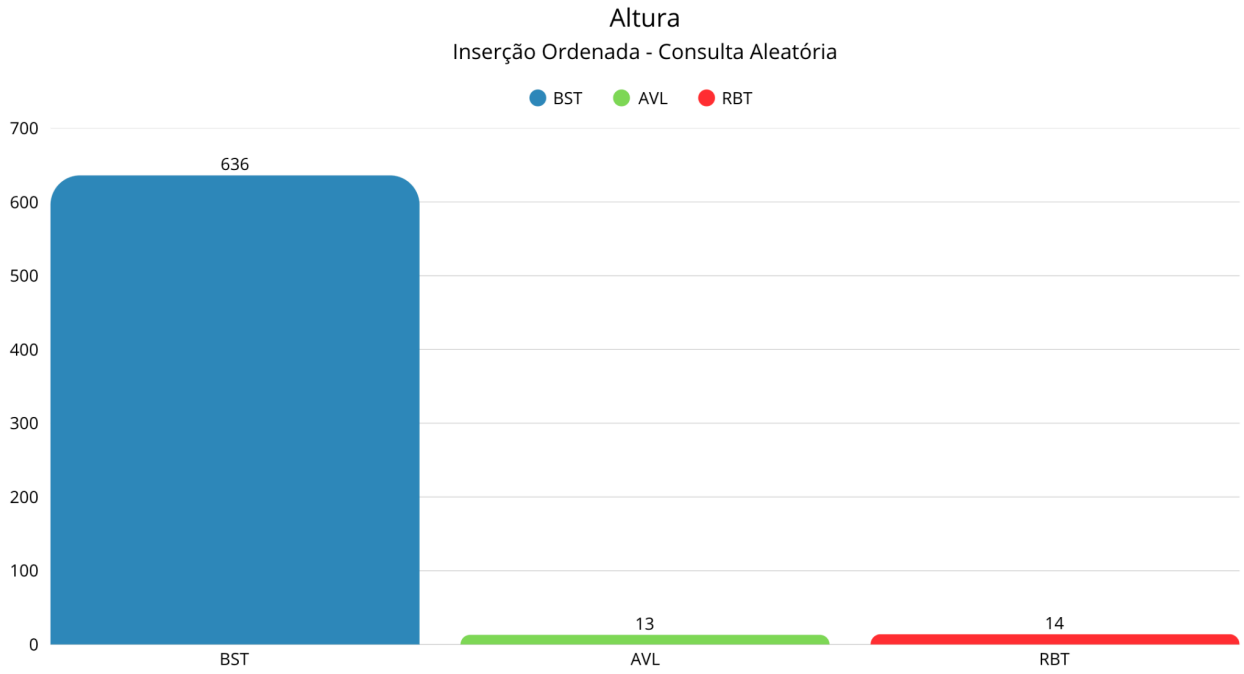
Nota: Nas buscas pelos 100 jogos ordenados de forma aleatória verificamos que a BST denovo observou um número exorbitante nesse caso de buscas ordenadas, o que demonstra sua ineficiência. A AVL e a RBT se comportaram novamente de forma similar.



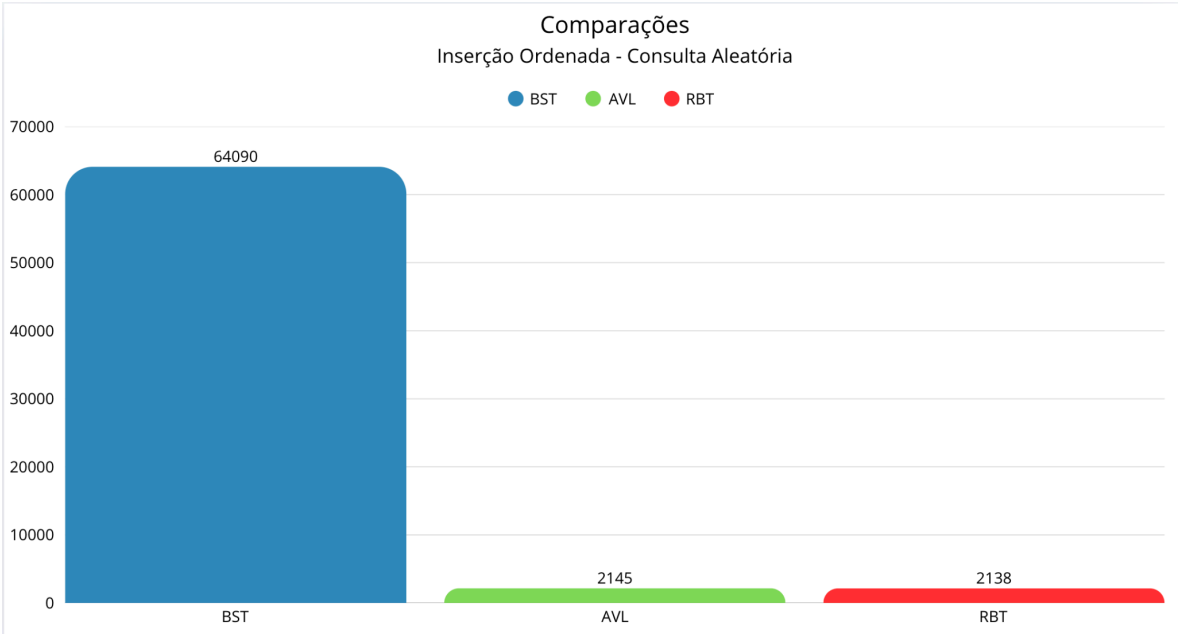
Nota: Nesse caso, a única diferença entre as árvores AVL e RBT são as rotações, em que a AVL realizou uma rotação a menos que a RBT nos momentos de inserção. A BST não aparece aqui pois não realiza rotações.

2. Inserção Ordenada - 3598 Jogos

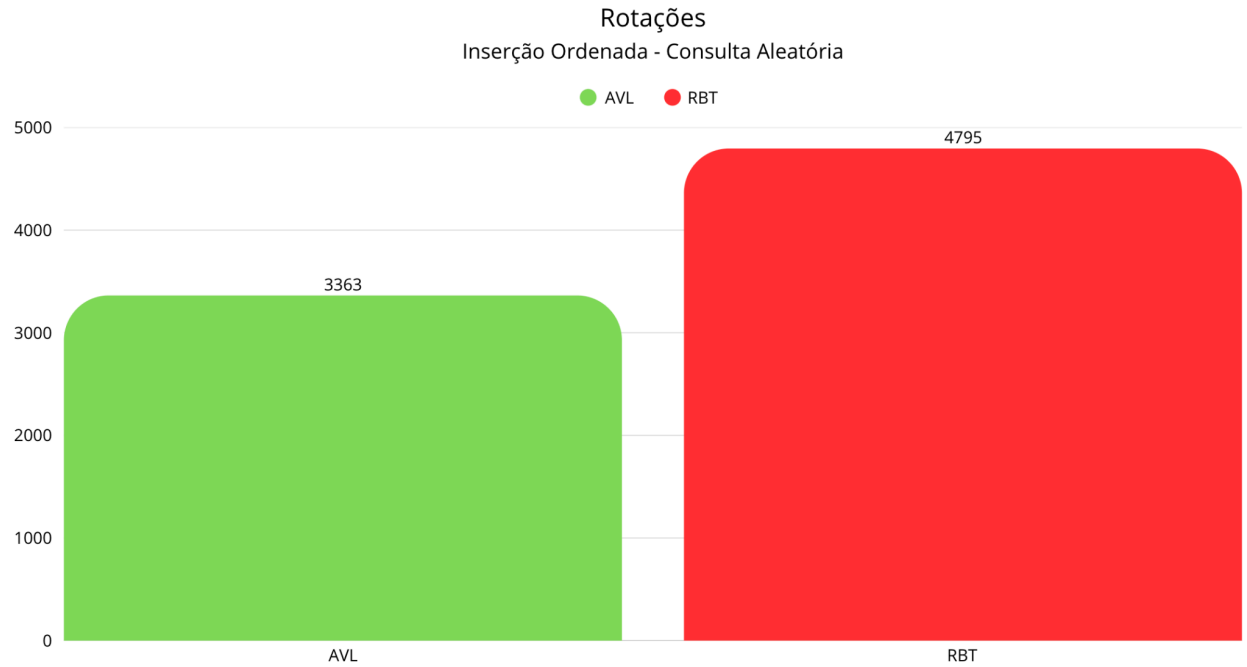
Neste caso, inserimos 3598 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 197 jogos ordenados aleatoriamente. Os resultados foram os seguintes:



Nota: Vemos que o padrão observado para 100 jogos se repete, com a BST ficando enorme. Porém, aqui a AVL ficou um pouco melhor que a RBT no quesito de altura.



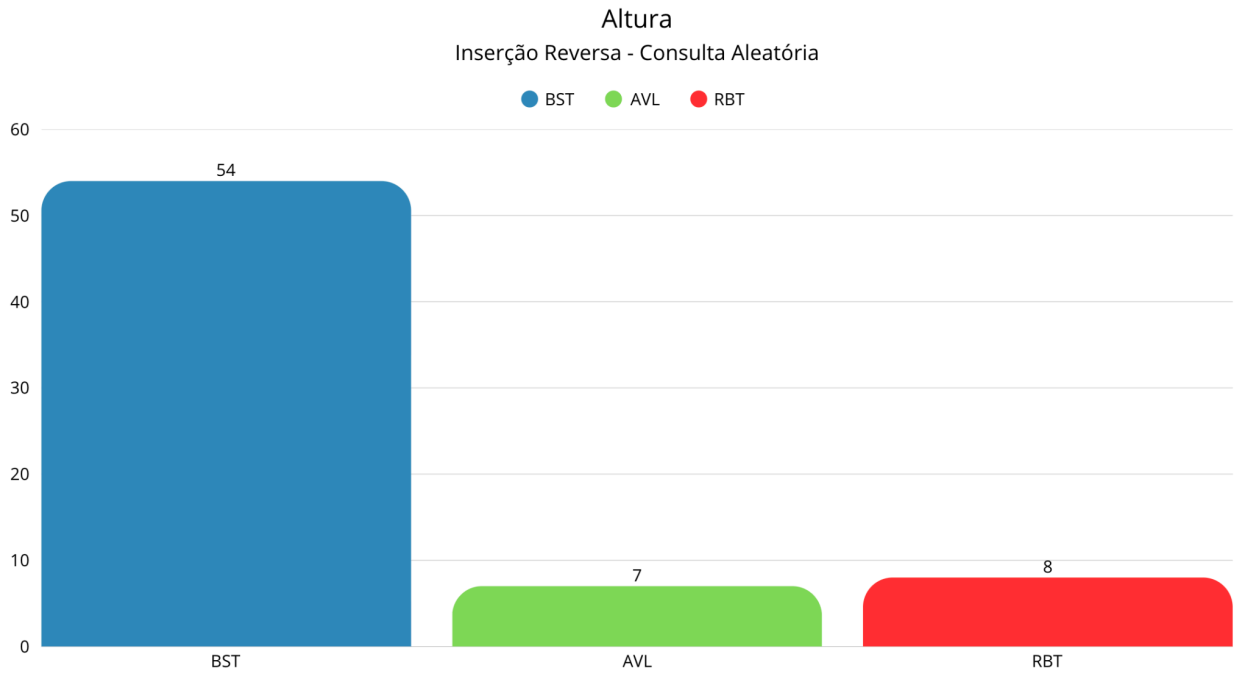
Nota: Novamente observamos que a BST ficou bem desbalanceada no número de comparações, mas nesse caso a RBT se saiu um pouco melhor que a AVL.



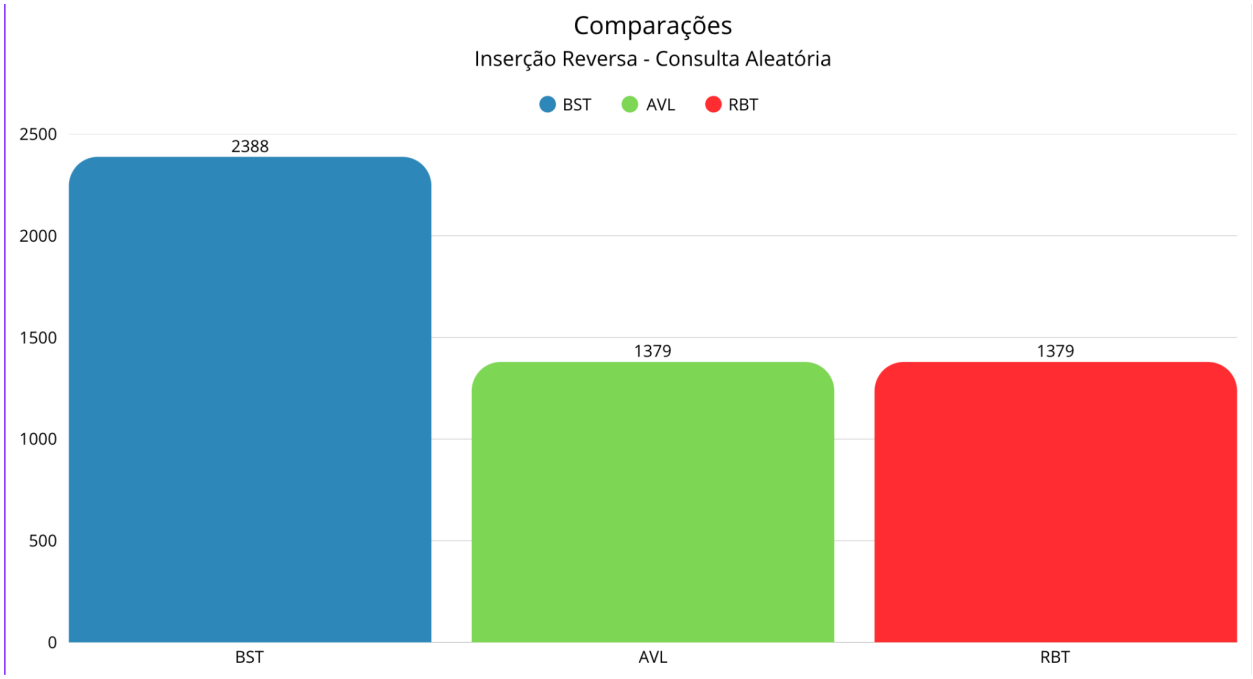
Nota: Aqui percebemos que a AVL foi construída de uma forma mais barata no quesito do número de rotações necessárias para balanceamento.

3. Inserção Reversa - 100 Jogos

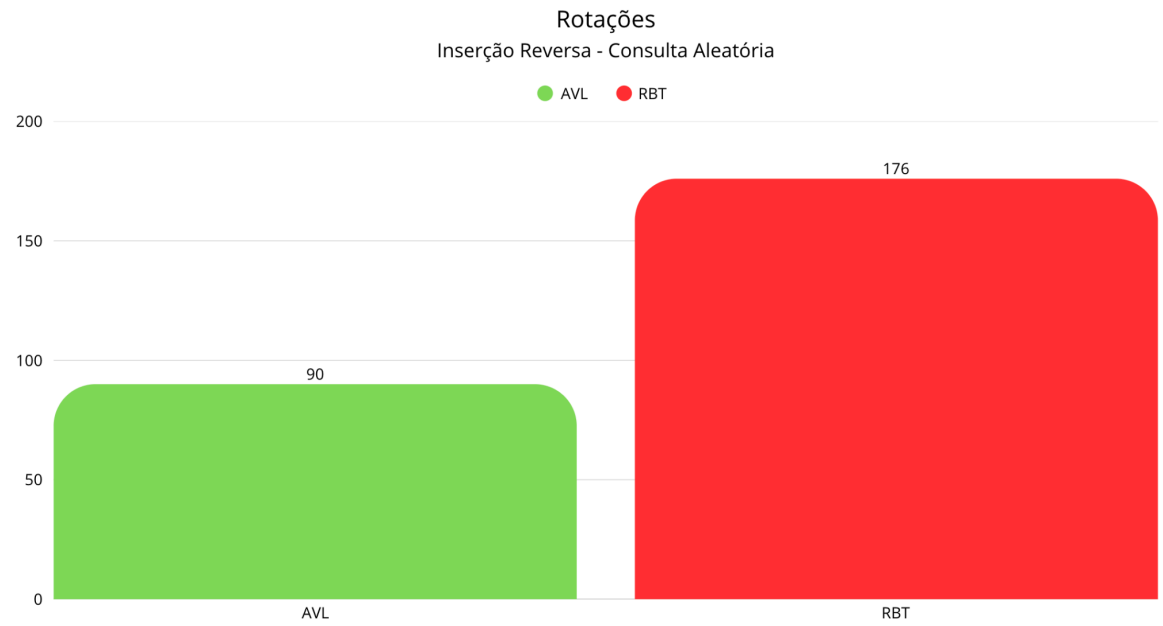
Neste caso, inserimos 100 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 197 jogos ordenados aleatoriamente. Os resultados foram os seguintes:



Nota: Aqui também vemos que pela ordem reversa a BST continua sendo ineficiente. A AVL sai um pouco melhor.



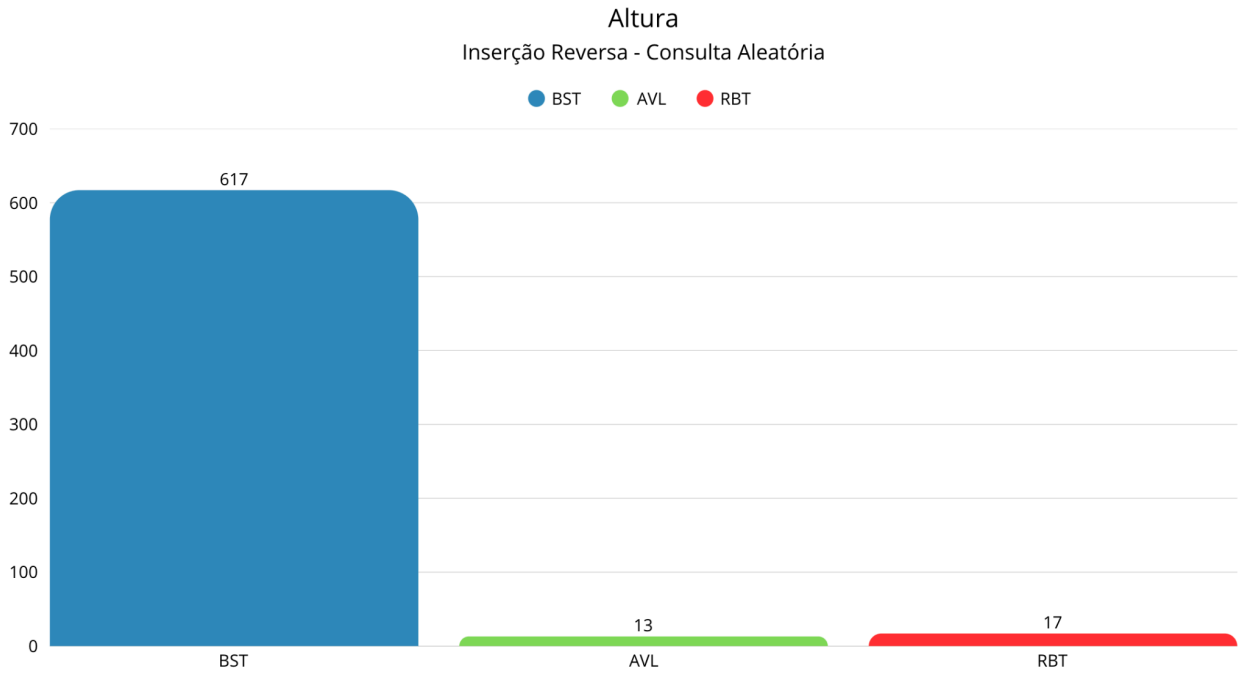
Nota: Aqui observamos que em termos de comparações AVL e RBT se saem iguais para os dados testados, enquanto a BST tem um custo significativo.

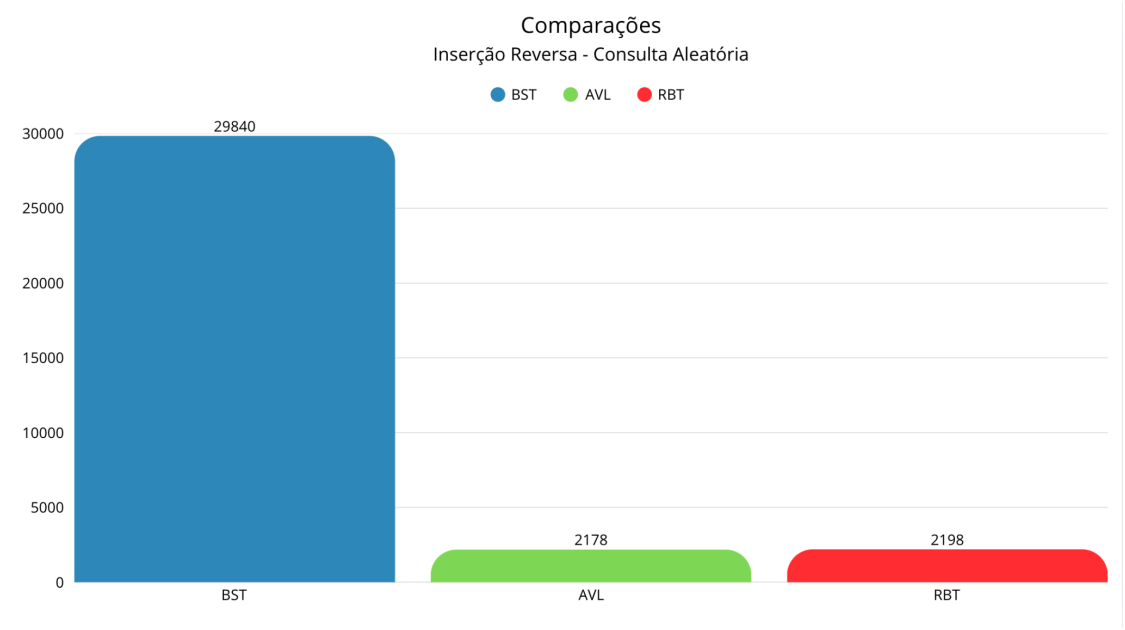


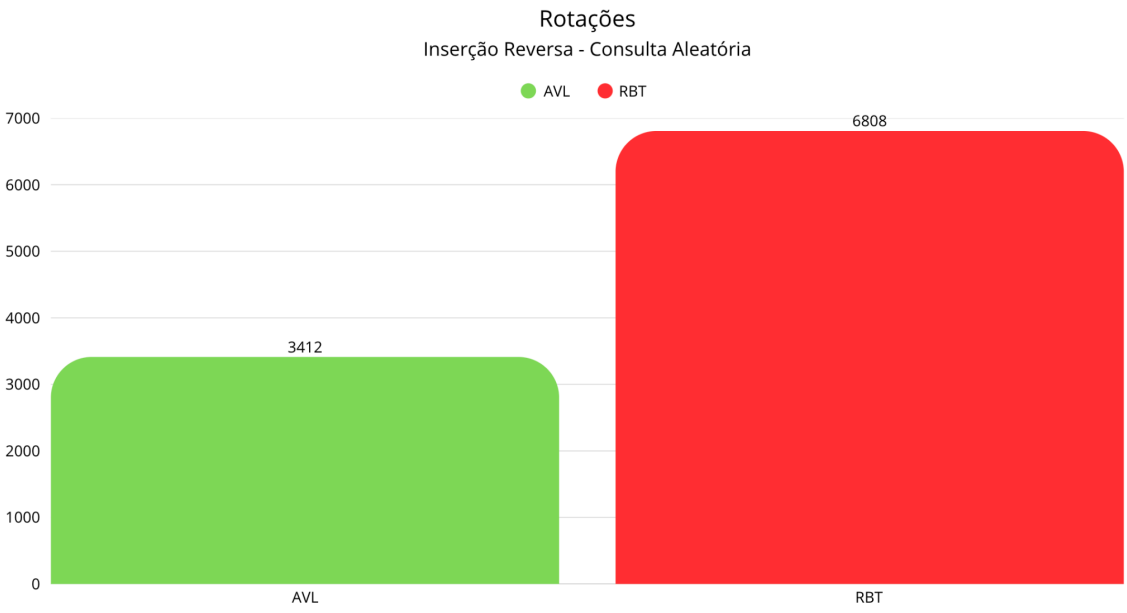
Nota: Observa-se também que na construção da árvore em ordem reversa, a AVL faz quase metade das rotações de uma RBT.

4. Inserção Reversa - 3598 Jogos

Neste caso, inserimos 3598 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 197 jogos ordenados aleatoriamente. Os resultados foram os seguintes:

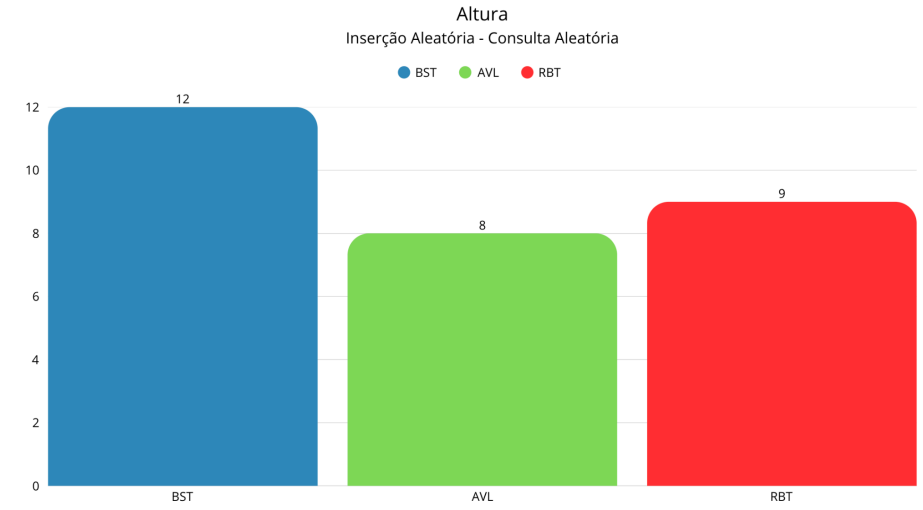


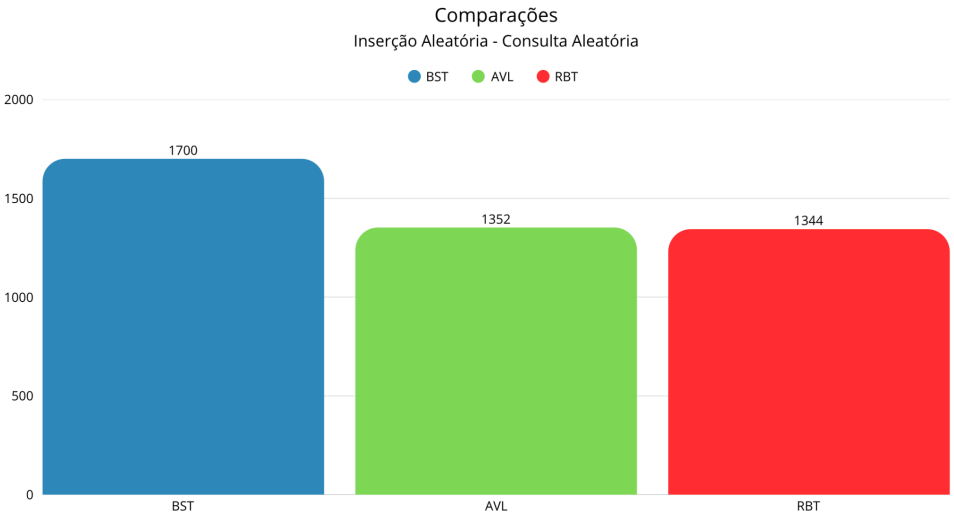


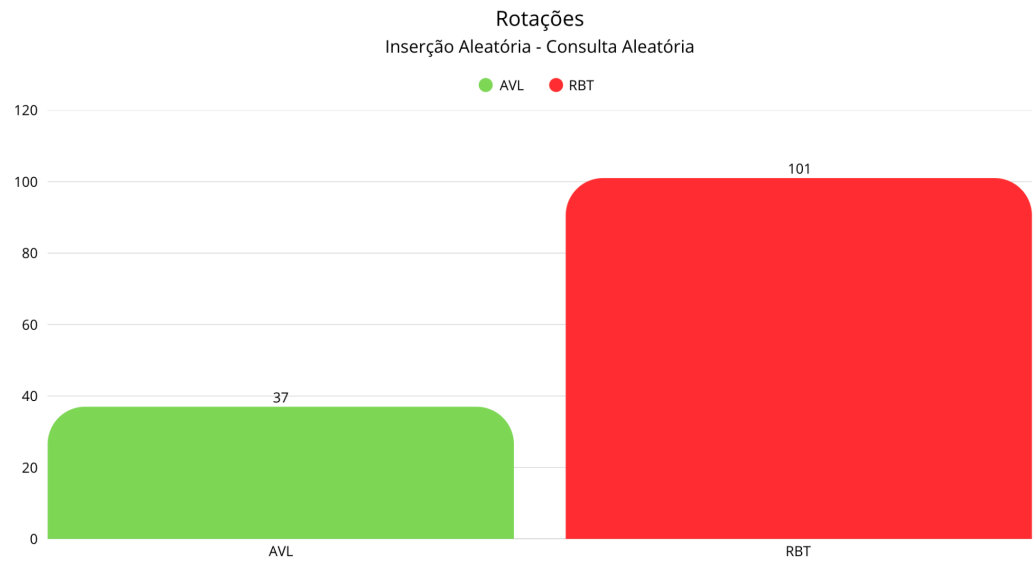


5. Inserção Aleatória - 100 Jogos

Neste caso, inserimos 100 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 197 jogos ordenados aleatoriamente. Os resultados foram os seguintes:

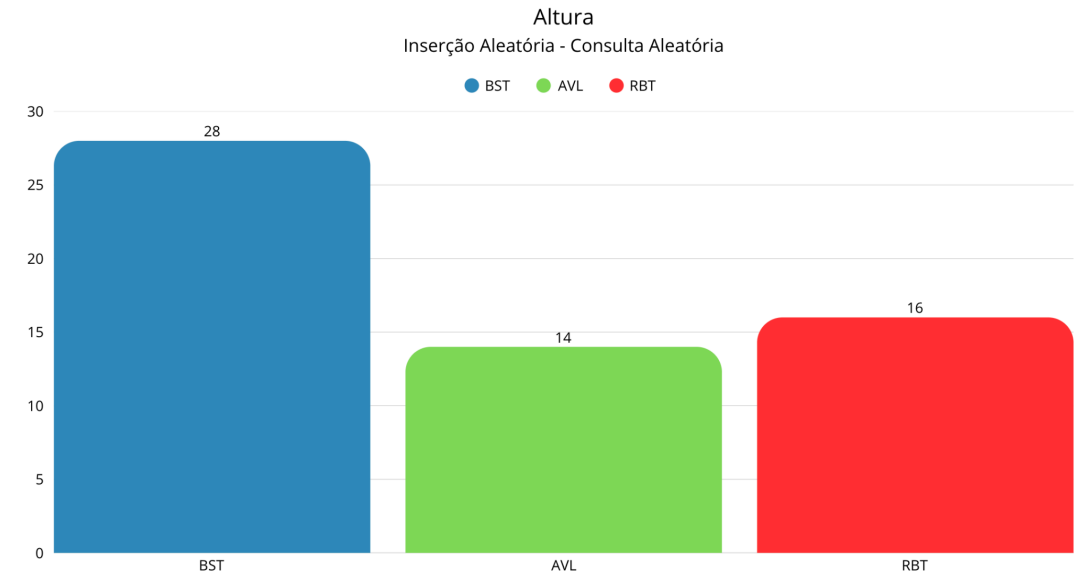

Altura
Inserção Aleatória - Consulta Aleatória
BST
AVL
RBT
12
8
9
12
10
8
6
4
2
0
BST
AVL
RBT

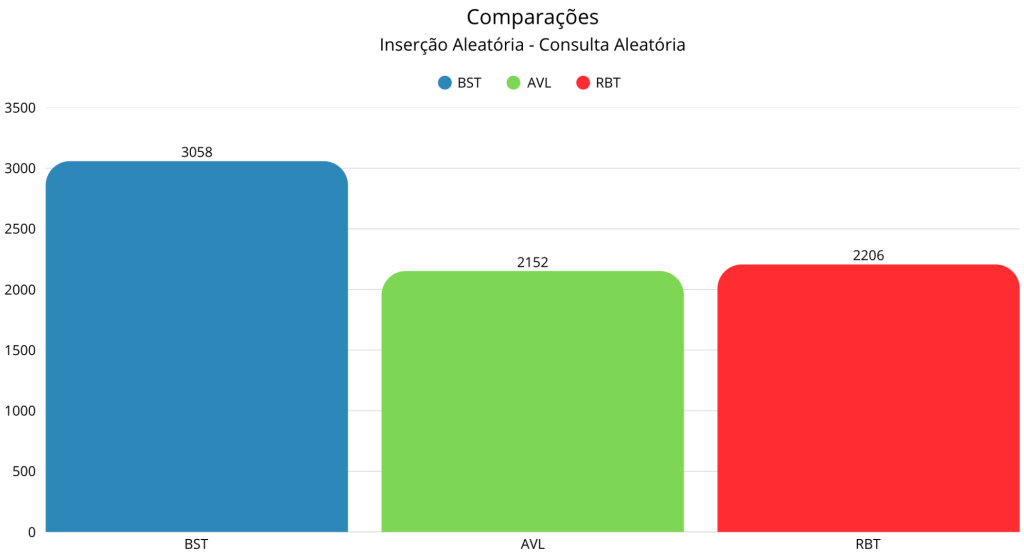

Comparações
Inserção Aleatória - Consulta Aleatória
BST
AVL
RBT
1700
1352
1344
2000
1500
1000
500
0
BST
AVL
RBT

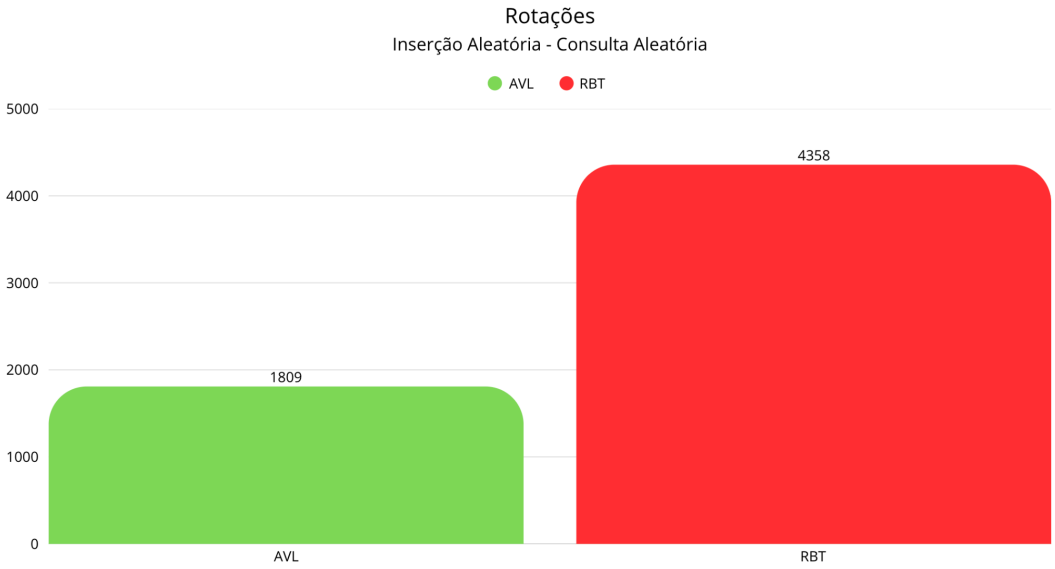

Rotações
Inserção Aleatória - Consulta Aleatória
AVL
RBT
37
101
120
100
80
60
40
20
0
AVL
RBT

6. Inserção Aleatória - 3598 Jogos

Neste caso, inserimos 3598 jogos e seus respectivos tempos de forma ordenada(alfabética) nas árvores. Assim, fizemos uma lista de jogos para consulta contendo 197 jogos ordenados aleatoriamente. Os resultados foram os seguintes:







4. Conclusão

Com este trabalho e análises de dados conseguimos concluir os seguintes aspectos:

1. **Balanceamento:** Em todos os casos analisados, a BST apresentou uma altura maior, em especial quando a ordem de inserção é ordenada, o que faz dela uma árvore desbalanceada. Já a AVL e a RBT apresentam balanceamento similar, com poucas divergências na altura, com a maior diferença de altura sendo 4, no caso da inserção reversa com o dataset grande. Quanto mais balanceada uma árvore é, suas operações de consulta tendem a ser menos custosas e mais rápidas, visto que há menos passos para encontrar seus itens, devido à baixa altura.
2. **Custo de pesquisa (Comparações):** Em geral, observamos uma grande disparidade entre a BST e as árvores balanceadas. Em todos os casos a BST realizou mais comparações que estas, demonstrando desempenho inferior nestes casos. Entre as árvores balanceadas, a AVL demonstrou o menor número de comparações em 3 casos, enquanto empatou em 2 casos com a RBT e perdeu em 1 caso. Esse comportamento condiz com a definição da AVL como uma árvore estritamente balanceada, pois permite que a diferença de altura entre suas subárvores seja no máximo 1, enquanto a rubro-negra utiliza outros critérios para o balanceamento. Assim, para uma mesma quantidade de nós a AVL tende a se destacar.
3. **Custo de construção (Rotações):** Em todos os casos observados a AVL demonstrou uma grande vantagem em relação a RBT na construção da árvore. Em grande parte dos testes a AVL demonstrou um número menor de rotações, o que implica em um custo menor(tempo e operações) na construção.
4. **Custo da Estrutura (Memória):** Como a AVL requer o armazenamento de um inteiro (-1, 0 e 1) indicando o fator de balanceamento para cada nó da árvore, ela custa mais espaço(memória) que a RBT, pois a RBT só precisa armazenar 1 bit de informação

indicando se um nó é vermelho ou preto. Então nesse quesito a RBT acaba sendo mais barata em termos de memória.

Portanto, observamos que mesmo que uma árvore seja organizada de forma binária para pesquisa, ainda há formas de otimizar as operações de consulta por meio de balanceamentos. Porém dependendo da forma em que o balanceamento é feito ele pode implicar em diferentes custos de memória, espaço e simplicidade. Logo, na hora de projetar aplicações que requerem a aplicação de alguma estrutura de dados, é importante ter em mente fatores como: tamanho dos dados, tipos de dados a serem armazenados, definir o que é mais importante entre custo de pesquisa e custo de construção, legibilidade e simplicidade do código a ser implementado. As árvores binárias de pesquisa são estruturas designadas para oferecer uma maior facilidade de pesquisa por meio da inserção ordenada dos dados de entrada, o que em geral possibilita custo ***O(log n)*** nas pesquisas, o que é desejado e eficiente.