UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN

CENTRO DE ENSINO SUPERIOR DO SERIDO – CERES

DEPARTAMENTO DE COMPUTAÇÃO E TECNOLOGIA – DCT

BACHARELADO EM SISTEMAS DE INFORMACAO

Prof. DSc. JOAO PAULO DE SOUZA MEDEIROS

**MATEUS MEDEIROS DE ARAÚJO**

# TRABALHO DE IMPLEMENTAÇÃO UNIDADE II

Caicó – RN

Junho - 2018

## Resumo

O presente relatório tem como objetivo apresentar estudos coletados durante a segunda unidade da disciplina de estruturas de dados ministrada pelo professor DSc. João Paulo de Souza Medeiros na Universidade Federal do Rio Grande no Norte. Serão apresentados gráficos e informações acerca de alguns algoritmos de busca em lista encadeada, árvore binária, arvore balanceada e tabela de disperção. Para implementação dos códigos foi utilizado o sistema operacional Linux Mint, a linguagem de programação C, o compilador GCC. A ferramenta GNUPLOT foi utilizada para criação dos gráficos.



## Abstract

This report presents a data structures unit taught by DSc for a decade. João Paulo de Souza Medeiros at the Federal University of Rio Grande do Norte. Graphs and information about some search algorithms will be presented in linked list, binary tree, balanced tree and dispersion table. Implementation of the codes was used the Linux Mint operating system, a C programming language, the GCC compiler. The GNUPLOT tool was used to create the graphics.

# SUMÁRIO

# 1 Introdução aos algoritmos de busca

Um algotitmo de busca em termos gerais é aquele que toma um problema como entrada e retorna uma solução, geralmente após resolver um número possível de soluções. Neste presente trabalho será discutido como se comporta um algoritmo de busca em determinadas estruturas de dados, mais especificamente a lista encadeada, árvore binária, árvore balanceada e tabela de disperção.

# 2 Busca em estruturas de dados

## 2.1 Busca em lista encadeada

Uma lista encadeada é uma maneira de representar um conjunto de dados que são adicionados em células, ou seja, o primeiro elemento é adicionado na primeira célula, o segundo elemento na segunda célula e assim por diante. Uma lista encadeada é uma sequência de células; cada célula contém um objeto de algum tipo e o endereço da célula seguinte.

Uma busca realizada em uma lista encadeada tem tempo de execução $O(1)$ em seu melhor caso e $O(n)$ em seu médio e pior caso.

O melhor caso da busca em uma lista encadeada ocorre quando o elemento buscado é encontrado logo na primeira verificação, sendo assim tendo tempo de execução constante $O(1)$. Seu melhor caso pode ser equacionado como

$$T_b(n) = C_1$$

onde $C_1$é a primeira linha executada da função de busca. É possível visualizar o gráfico de tempo de execução na figura 1.

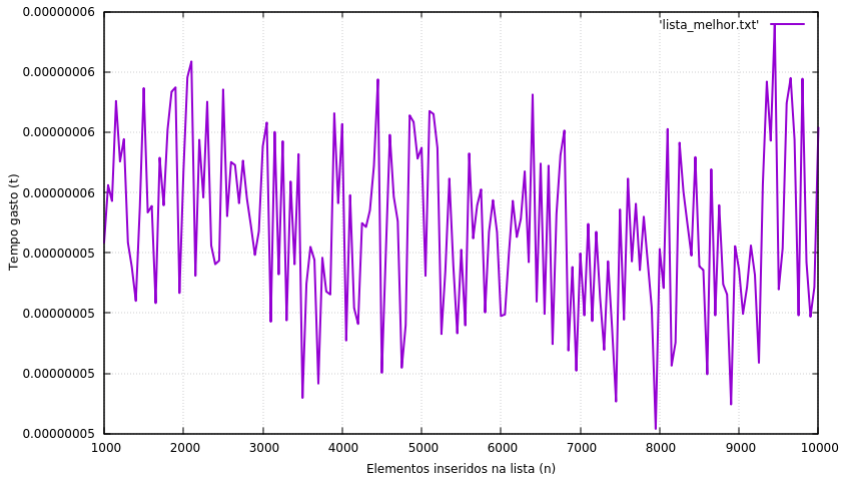


Figure 1: Tempo de execução do melhor caso na busca em lista encadeada

Nota-se que embora haja variações o tempo de execução é constante em um intervalo. Para geração do gráfico a lista foi preenchida de forma aleatória e o elemento procurado sempre foi encontrado na primeira verificação.

O pior caso ocorre quando o elemento buscado não está presente na lista, assim tendo tempo de execução linear $O(n)$. O pior caso pode ser equacionado como

$$T_w(n) = C_1 + C_2 + C_3 + T_w(n-1)$$

observa-se que se obtém uma relação de recorrência que tem como caso base

$$T_w(0) = C_1$$

então, calculando a próxima relação

$$T_w(n-1) = C_1 + C_2 + C_3 + T_w(n-2)$$

$$T_w(n) = C_1 + C_2 + C_3 + [C_1 + C_2 + C_3 + T_w(n-2)] =$$
$$2(C_1 + C_2 + C_3) + T_w(n-2) =$$

substituindo pelo padrão de troca

$$x(C_1 + C_2 + C_3) + T_w(n-x)$$

$$n - x = 0$$

$$x = n$$

então,

$$n(C_1 + C_2 + C_3) + C_1$$

É possível visualizar o gráfico de tempo de execução na figura 2.

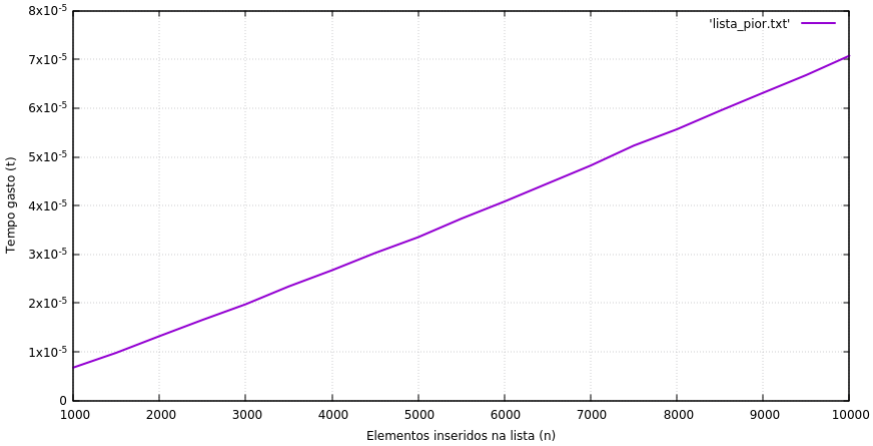


Figure 2: Tempo de execução do pior caso da busca em lista encadeada

Para geração do gráfico a lista foi preenchida de forma aleatória e o elemento procurado não foi encontrado.

Para o caso médio esperado, a lista foi preenchida de forma aleatória e foi procurado um número aleatório. Na figura 3 é possível visualizar seu tempo de execução.

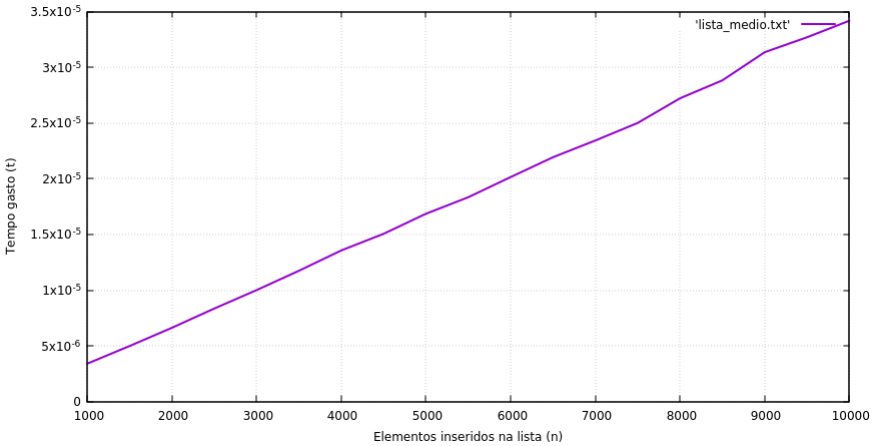


Figure 3: Tempo de execução do caso médio da busca em lista encadeada

Na figura 4 é possível visualizar a comparação entre o tempo de execução dos três casos.

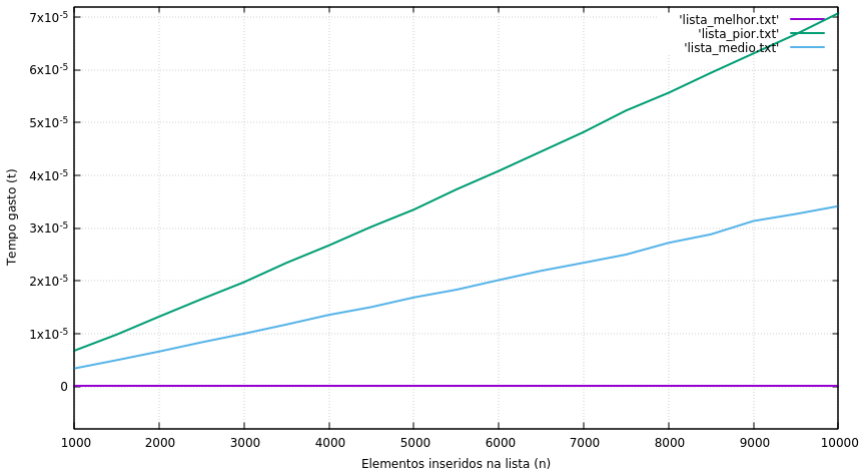


Figure 4: Tempo de execução do caso médio da busca em lista encadeada

## 2.2 Busca em árvore binária

Árvore binária de busca é uma estrutura de dados baseada em nós sendo que todos os valores numéricos à esquerda do nó são inferiores a ele e todos os valores numéricos à direita são maiores do que ele, essencialmente. O objetivo dessa estrutura é permitir que uma busca seja feita em ordem $O\left(log_2 n\right)$.

Uma busca realizada em uma árvore binária tem tempo de execução $O\left(n\right)$ em seu pior caso, $O\left(log_2 n\right)$ no médico caso, e tem tempo de execução $O\left(1\right)$ em seu melhor caso.

O melhor caso da busca em uma árvore binária, assim como quase todas outras estruturas, ocorre quando o elemento buscado é encontrado logo na primeira verificação, portanto sendo de tempo de execução constante $O\left(1\right)$. Seu melhor caso pode ser equacionado como

$$T_b\left(n\right) = C_1 + C_2 + C_3$$

É possível visualizar o gráfico de tempo de execução na figura 5

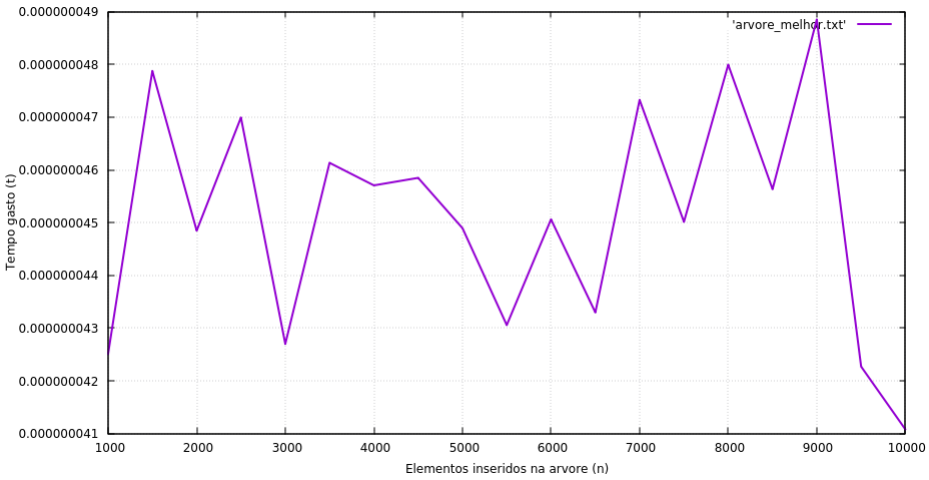


Figure 5: Tempo de execução do melhor caso da busca em árvore binária

Para geração do gráfico a árvore foi preenchida de forma aleatória e o elemento procurado sempre foi encontrado na primeira verificação.

O pior caso ocorre quando árvore é preenchida em ordem crescente ou decrescente tornando-a semelhante à uma lista encadeada, já que todos elementos estarão apenas de um lado. Neste caso, seu tempo de execução é linear $O(n)$. É possível visualizar o gráfico de tempo de execução na figura 6

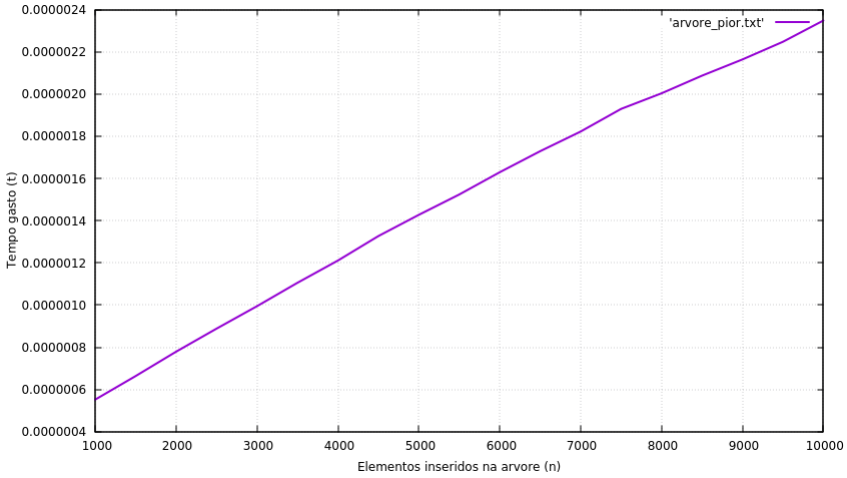


Figure 6: Tempo de execução do pior caso da busca em árvore binária

Para geração do gráfico a lista foi preenchida em ordem crescente e o elemento procurado não foi encontrado.

Para o caso médio esperado, a árvore foi preenchida de forma aleatória e foi procurado um número aleatório. Na figura 7 é possível visualizar seu tempo de execução.

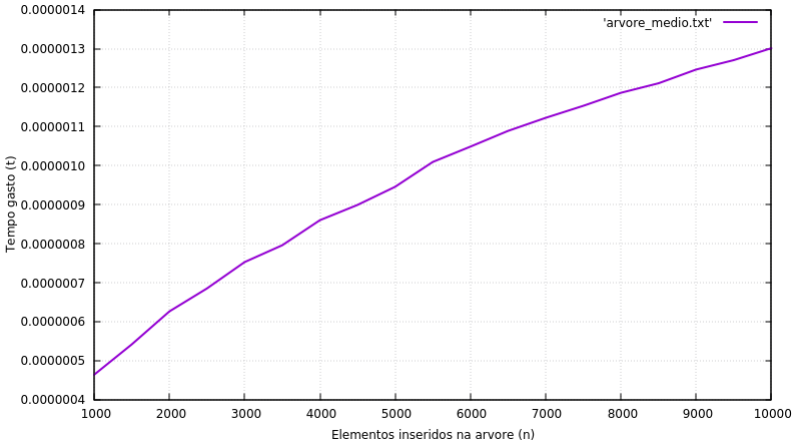


Figure 7: Tempo de execução do caso médio da busca em árvore binária

Na figura 8 é possível visualizar a comparação entre o tempo de execução dos três casos.

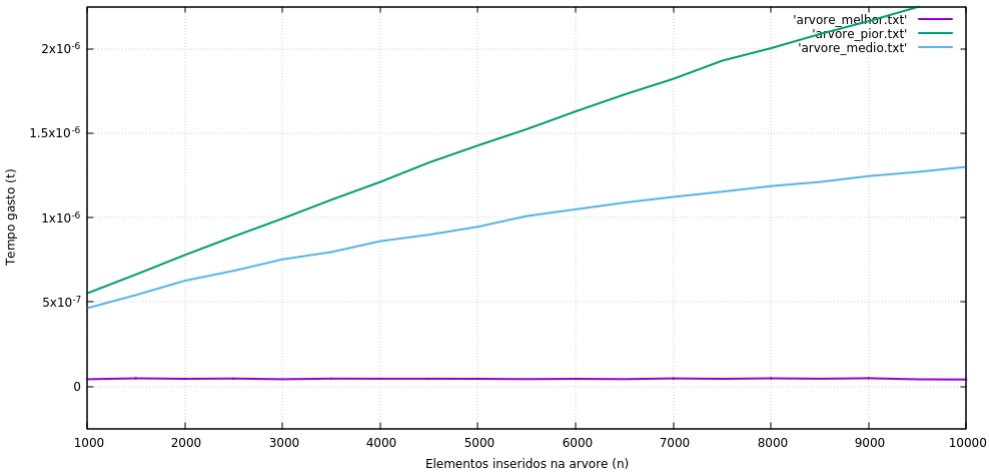


Figure 8: Tempo de execução dos três casos da busca em árvore binária

## 2.3 Árvore balanceada

Árvore balanceada ou árvore AVL funciona de maneira semelhante à árvore binária, entretando, mantendo um balanceamento entre o lado direito e es-

querdo de cada nó, garantindo assim que seja efetuado um menor número de verificações. Para manter esse balanceamento geralmente se verifica a cada nova inserção se existe algum lado desbalanceado, se houver, é necessário fazer algumas rotações para torna-la balanceada novamente. O pior e médio caso da busca em árvore balanceada tem tempo de execução $O\left(log_2 n\right)$ e seu melhor caso $O\left(1\right)$.

O melhor caso da busca em uma árvore balanceada ocorre quando o elemento buscado é encontrado logo na primeira verificação. O melhor caso pode ser equacionado como

$$T_b\left(n\right) = C_1 + C_2 + C_3$$

O pior caso da busca ocorre quando o elemento buscado não é encontrato, pode ser equacionado como

$$T_w\left(n\right) = C_1 + C_2 + C_{4,6} + C_{3,5} + T_w\left(\frac{n}{2}\right)$$

observa-se que se obtém uma relação de recorrência que tem como caso base

$$T\left(1\right) = C_1$$

então, calculando a próxima relação

$$T_w\left(\frac{n}{2}\right) = C_1 + C_2 + C_{4,6} + C_{3,5} + T_w\left(\frac{n}{4}\right)$$

substituindo na equação original

$$T_w\left(n\right) = C_1 + C_2 + C_{4,6} + C_{5,7} + \left[C_1 + C_2 + C_{4,6} + C_{3,5} + T_w\left(\frac{n}{4}\right)\right] =$$

$$2\left(C_1 + C_2 + C_{4,6} + C_{5,7}\right) + T_w\left(\frac{n}{4}\right)$$

substituindo pelo padrão de troca

$$T_w\left(n\right) = x\left(C_1 + C_2 + C_{4,6} + C_{5,7}\right) + T_w\left(\frac{n}{2^x}\right)$$

$$\frac{n}{2^x} = 0$$

$$n = 2^x$$

$$log_2 n = x$$

substituindo na equação original se obtém

$$T_w\left(n\right) = log_2 n\left(C_1 + C_2 + C_{4,6} + C_{5,7}\right) + T_w\left(\frac{log_2 n}{log_2 n}\right) =$$

então

$$log_2 n\,(C_1 + C_2 + C_{4,6} + C_{5,7}) + C_1$$

É possível visualizar o gráfico de tempo de execução na figura 9.

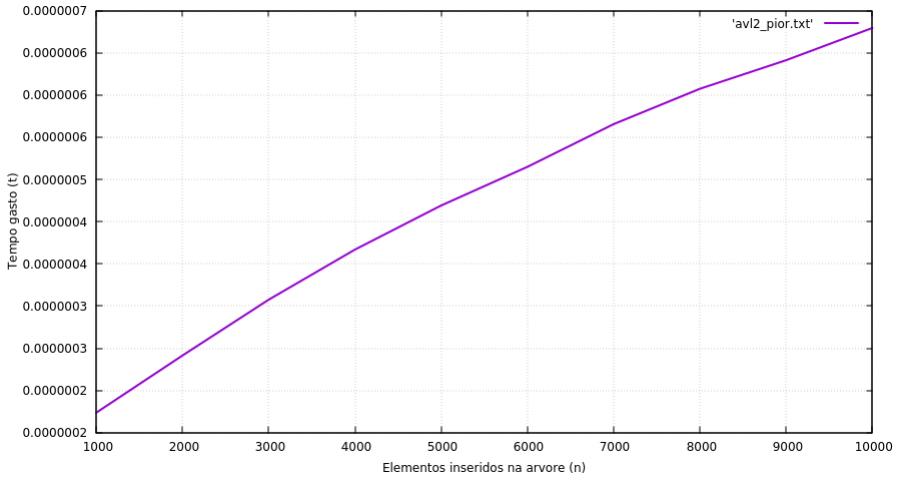


Figure 9: Tempo de execução do pior caso da busca em árvore balanceada

Para geração do gráfico a árvore foi preenchida de forma aleatória e o elemento procurado não foi encontrado.

Para o caso médio esperado, a árvore foi preenchida de forma aleatória e foi procurado um número aleatório. Na figura 10 é possível visualizar seu tempo de execução.

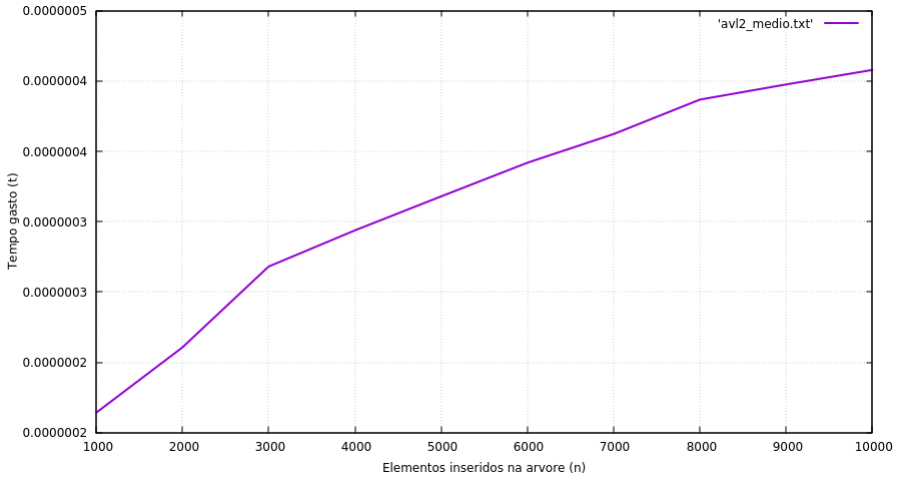


Figure 10: Tempo de execução do caso médio da busca em árvore balanceada

Na figura 11 é possível visualizar a comparação entre o tempo de execução dos dois casos.

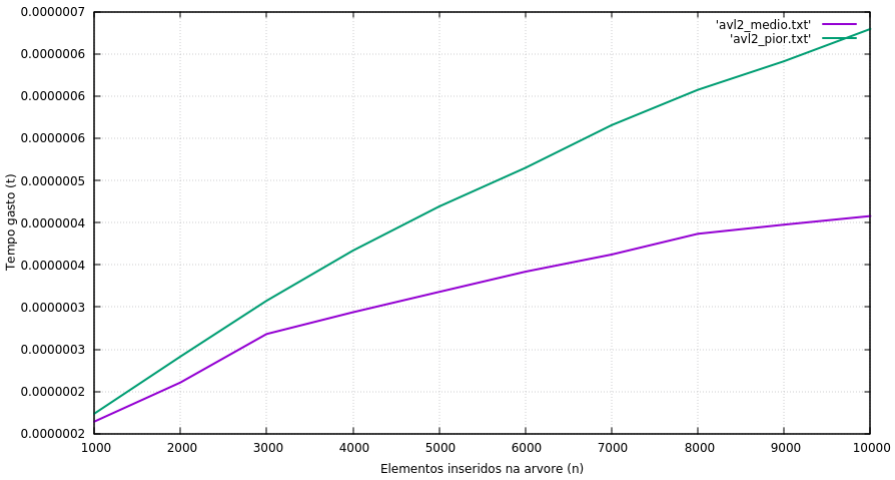


Figure 11: Tempo de execução dos casos da busca em árvore balanceada

## 2.4 Busca em tabela de disperção

Tabela de disperção, também conhecida como tabela hash ou tabela de espalhamento, é uma estrutura de dados que funciona associando chaves à valores. O principal objetivo dessa estrutura de dados é fazer com que a partir de uma chave seja possível fazer uma rápida busca e encontrar o elemento desejado. A tabela de disperção tem tempo de execução $O(1)$ em seu melhor e médio caso e, pode chegar a ter tempo de execução até $O(n)$ em seu pior caso.

É possível equacionar o tempo de execução esperado de uma tabela de disperção como

$$T_a(n) = \frac{1}{n} + (m-1)\left(\frac{1}{m}\right) =$$

$$\frac{1}{n}n + \left(\frac{n-1}{m}\right) =$$

$$1 + 1 - \frac{1}{n}$$

O gráfico de tempo de execução médio da tabela de disperção pode ser visto na figura 12.

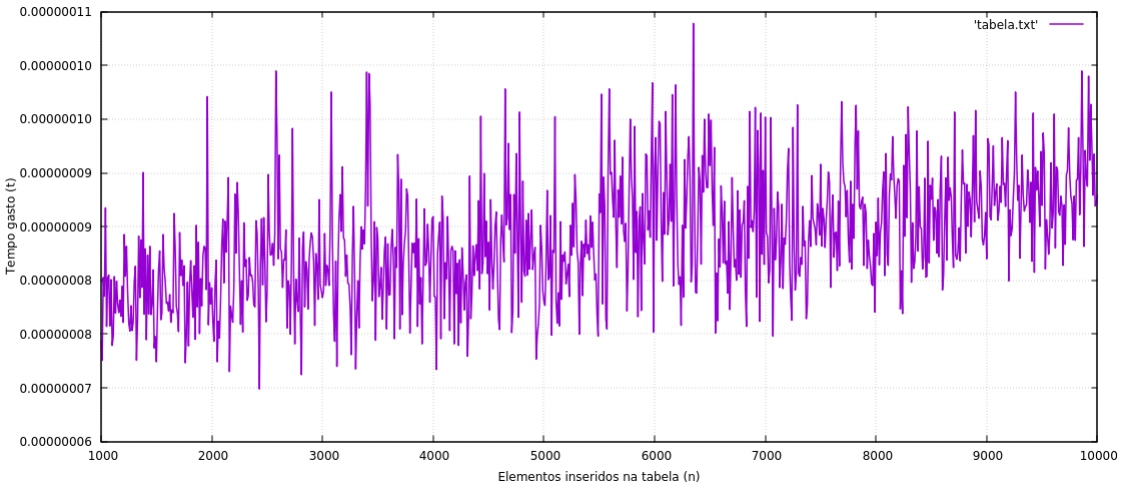


Figure 12: Tempo de execução médio de busca em uma tabela hash

# 3 Comparação entre as buscas em estruturas de dados

Todas as estruturas utilizadas para a análise foram testadas com uma quantidade de elementos inseridos variando entre 1000 e 10000. Essa sessão tratará de comparar as estruturas entre elas afim de verificar a diferença entre os tempos de execução, para isso, as comparações serão separadas em algumas ordens, que serão: ordem 1, onde serão comparados todas as estruturas em seu pior caso (se tiver); ordem 2, onde serão comparados todos os casos médios; ordem 3, onde serão comparados os casos da árvore binária e árvore balanceada especificamente; caso 4, onde serão comparados todos os casos de todas as estruturas; e por fim, o caso 5, onde serão mostrados todos os casos de todas estruturas exceto a lista encadeada

## 3.1 Ordem 1

Aqui serão comparadas todas estruturas em seu pior caso. É possível visualizar os tempos de execução na figura 13.

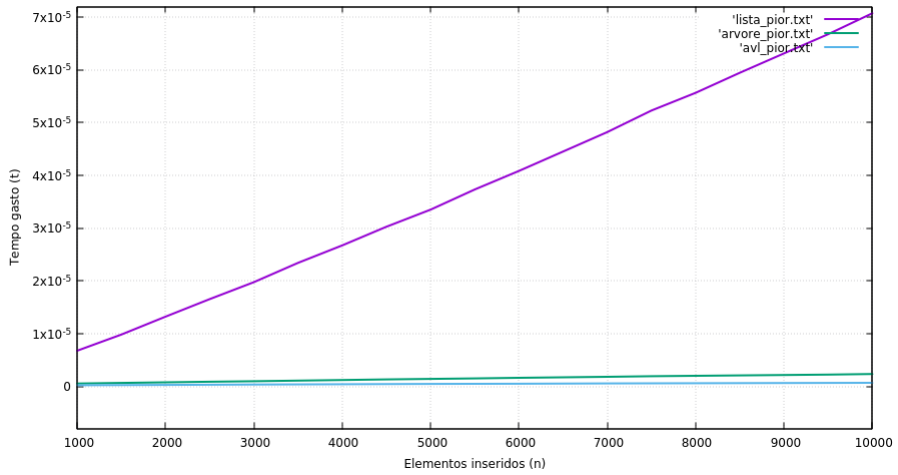


Figure 13: Tempo de execução das estruturas em seu pior caso

A partir do gráfico é possível concluir que o tempo de execução de uma lista encadeada em seu pior caso (linear) é muito superior às demais estruturas no mesmo caso.

## 3.2 Ordem 2

Aqui serão comparadas todas estruturas em seu caso médio. É possível visualizar os tempos de execução na figura 14.

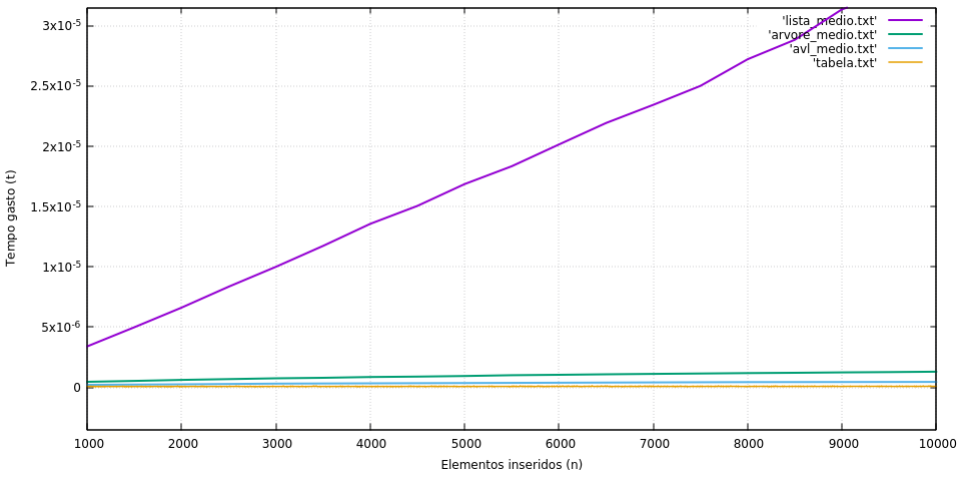


Figure 14: Tempo de execução das estruturas em seu caso médio

Assim como foi visto na ordem 14, o tempo de execução da lista encadeada

em seu caso médio é muito superior às demais estruturas no mesmo caso.

## 3.3 Ordem 3

Aqui serão comparadas os casos da árvore balanceada e binária mais especificamente. Na figura 15 é possível visualizar o tempo de execução da árvore balanceada e binária em seus piores casos.

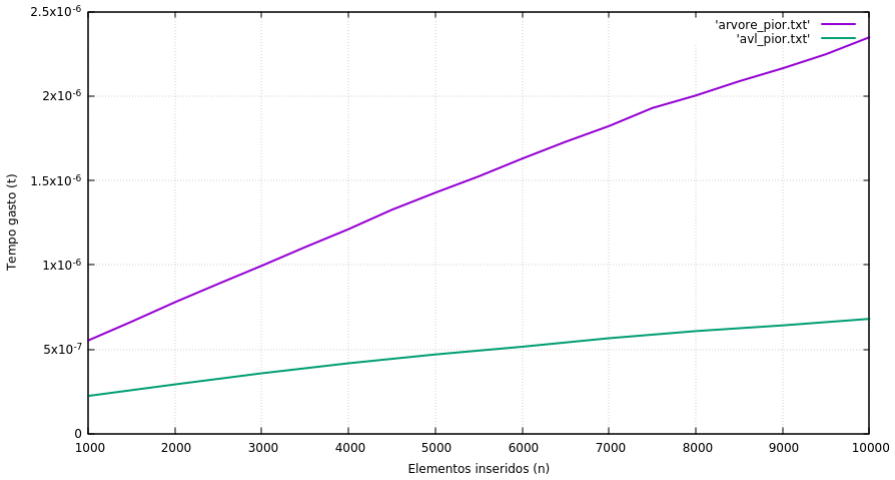


Figure 15: Tempo de execução das árvore balanceada e binária em seu pior caso

A partir da figura é possível concluir que o tempo de execução da árvore balanceada em seu pior caso é inferior à árvore binária no mesmo caso.

Na figura 16 é mostrado o tempo de execução das árvores balanceada e binária em seus casos médios.

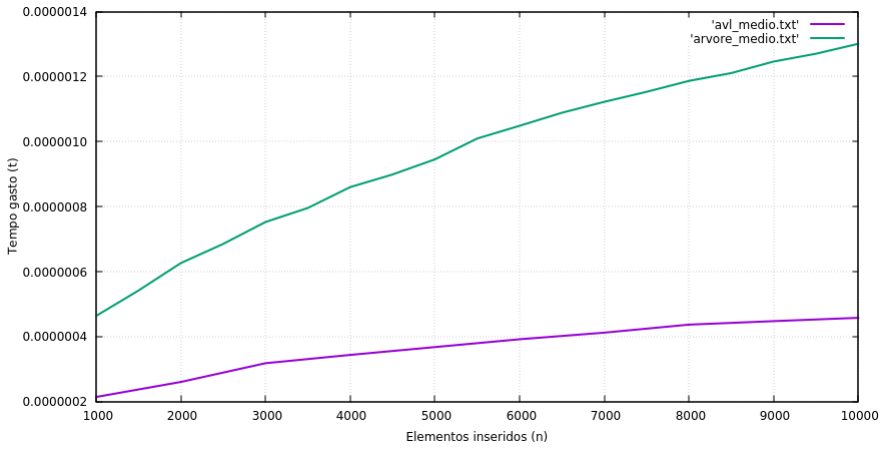


Figure 16: Tempo de execução das árvore balanceada e binária em seu caso médio

Nota-se que, assim como no pior caso de ambas, no caso médio o tempo de execução da árvore balanceada segue sendo inferior à árvore binária.

## 3.4 Ordem 4

Nesta ordem serão apresentadas todos os casos de todas as estruturas. Na figura 17 é possível visualizar os tempos de execução

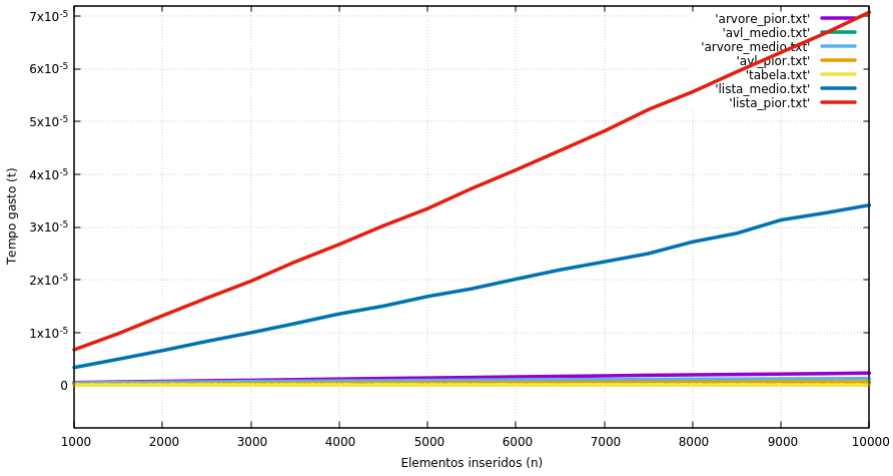


Figure 17: Tempo de execução de todos os casos de todas estruturas

Como já mostrado anteriormente neste trabalho e reafirmado na figura 17, o tempo de execução da busca em uma lista encadeada tanto no pior quanto no

médio caso é superior às demais estruturas, causando até mesmo uma dificuldade para verificar a diferença no tempo de execução das demais. Por conta disso, na ordem 5 será excluida a lista encadeada para melhor visualização.

## 3.5 Ordem 5

Nesta ordem será comparado o tempo de execução entre os casos da árvore binária, balanceada e tabela hash. O gráfico pode ser visto na figura 18.

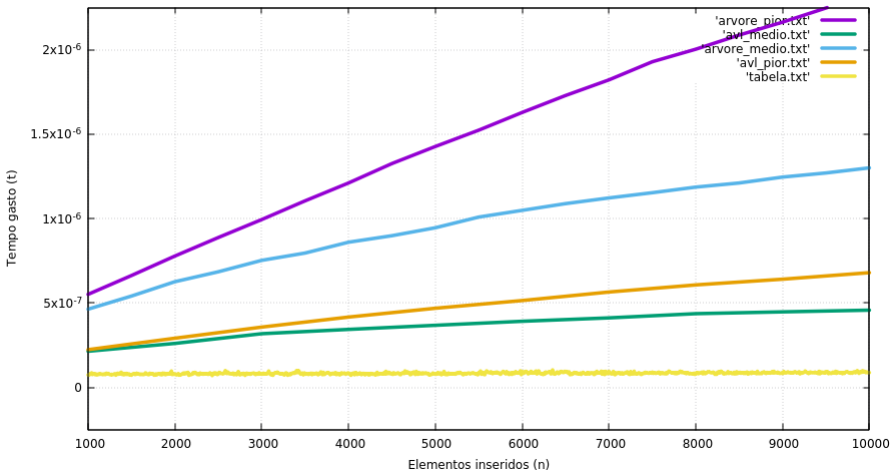


Figure 18: Tempo de execução de todos os casos de todas estruturas

Analisando o gráfico da figura 18 verifica-se que a busca em uma tabela hash contém o menor tempo de execução dentre as demais estruturas. A busca em uma árvore balanceada, em ambos os casos, tem tempo de execução superior à tabela hash mas inferior à busca em uma árvore binária, que por sua vez contém o maior tempo de execução dentre as demais, em ambos os casos.

## References


[1] LEVITIN, A.. Introduction to the Design and Analysis of Algorithms. 2. Addison Wesley. 2006

[2] CORMEM, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C.. Introduction to Algorithms. 3. The MIT Press. 2009

[3] DROZDEK, Adam. Estrutura de Dados e Algoritmos em C++. CENGAGE Learning, 2002

[4] CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a Estruturas de Dados: com técnicas de programação em C. 7a ed. Elsevier 2004.

[5] ASCENCIO, Ana Fernanda G.; ARAÚJO, Graziela S. Estruturas de Dados: Algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson, 2010.