RELATÓRIO DA DISCIPLINA ESTRUTURA DE DADOS

ALUNO: HUGO RAFAEL DE MEDEIROS FERNANDES

PROFESSOR: JOÃO PAULO DE SOUZA MEDEIROS

LABORATÓRIO ALAN TURING – PRÉDIO BSI

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – CERES/CAICÓ

Email: hugo.fernandes@live.com

RESUMO:

Utilizando os conhecimentos estudados em sala sobre algoritmo e complexidade, foi desenvolvido em laboratório e em sala de aula, uma tarefa prática utilizando estruturas de dados, para verificar seus comportamentos e determinar a ordem de seus tempos de execução.

1. INTRODUÇÃO

O objetivo deste relatório é relatar um experimento realizado em laboratório e em sala de aula pela disciplina *Estrutura de Dados*, com o objetivo de verificar, desenvolver e entender o funcionamento dos algoritmos em questão, medindo e comparando seus tempos de execução.

2. ALGORITMOS UTILIZADOS

Os algoritmos utilizados foram:

- Lista Encadeada;
- Árvore Binária;
- Árvore AVL
- Tabela Hash

3. FERRAMENTAS UTILIZADAS

As ferramentas utilizadas foram:

- Papel e Caneta;
- Computador;
- Sistema Operacional *Debian GNU/Linux*;
- *Software Gnuplot*;
- Processador de texto *Gedit*;
- Linguagem de programação *C*.

4. MÉTODO DE MEDIÇÃO E COMPARAÇÃO DOS RESULTADOS

O método de medição utilizado para capturar o tempo de execução nos algoritmos foi através da função *gettimeofday* presente na biblioteca *sys/time.h* da linguagem *C*, onde seus resultados são inseridos num arquivo de texto, por onde é gerado o gráfico do tempo de execução do algoritmo através da ferramenta *Gnuplot*.

Como a busca de informações é uma tarefa comum na ciência da computação, então daremos enfase a está, medindo e comparando os resultados para se verificar qual destas estruturas

de dados possui o melhor tempo de busca de informações.

5. REFERÊCIAS

1. Algoritmos: teoria e prática / Thomas H. Comen... [et al.]; tradução da segunda edição [americana] Vandenberg D. de Souza – Rio de Janeiro: Elsevier, 2002 – 6ª reimpressão.

6. ANEXOS

Abaixo estão os comentários a respeito dos algoritmos estudados, bem como os gráficos gerados a partir dos códigos desenvolvidos em laboratório.

1. LISTA ENCADEADA

A lista encadeada consiste em uma estrutura de dados onde o elemento anterior contêm uma referência para o seguinte. Dessa forma, cada elemento da lista encadeada conhece seu sucessor.

Em uma lista encadeada, os elementos não necessariamente precisam está organizados de forma sequencial, podendo cada elemento está em uma posição qualquer na memória.

O mais importante em uma lista encadeada é conhecer a referência para o primeiro elemento, pois a partir deste pode-se percorrer qualquer elemento na lista.

1.1 Tempo de Busca

O tempo de busca em uma lista encadeada depende de como os elementos foram inseridos e de qual elemento será buscado.

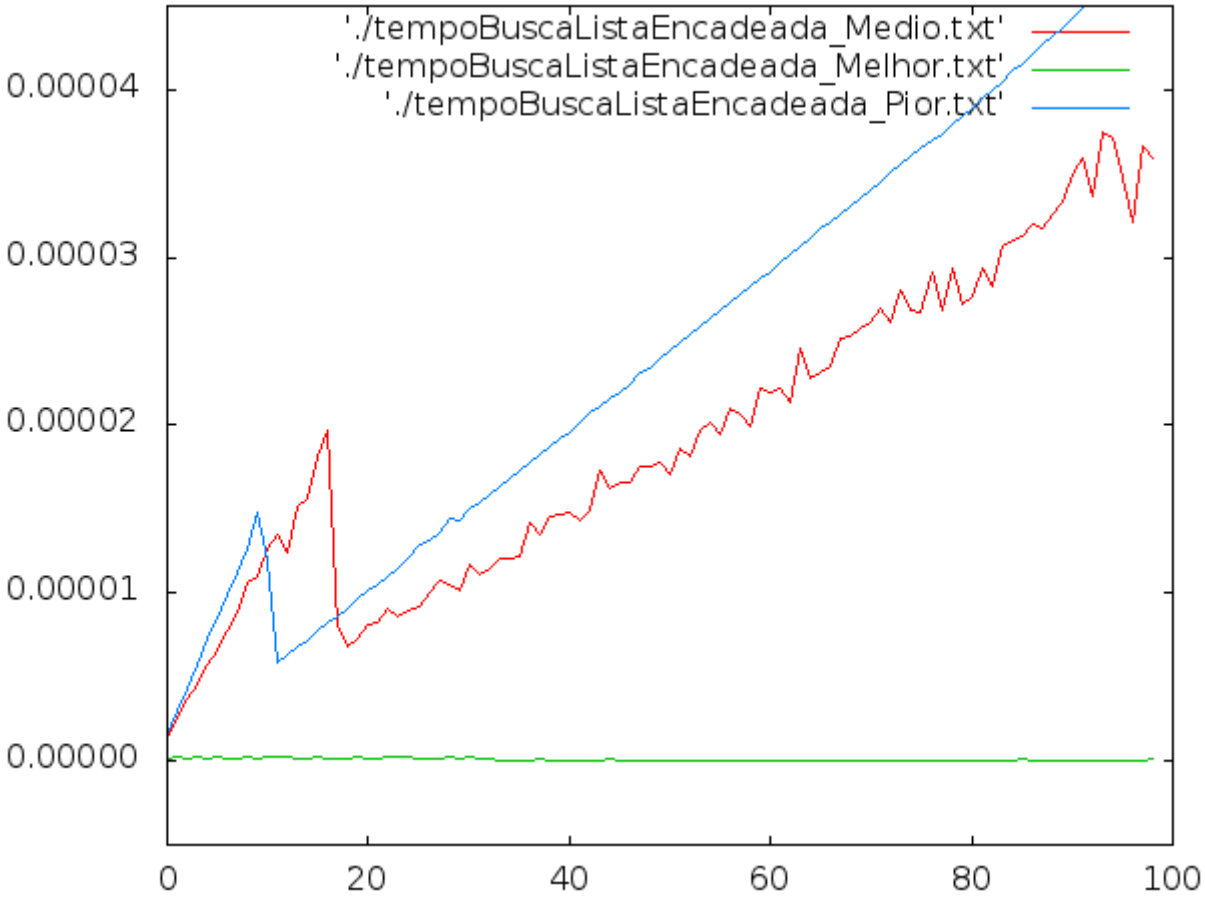
No exemplo desenvolvido, os elementos são inseridos de forma aleatória e buscados de três formas diferentes que são:

- Melhor Caso (Constante) – quando o elemento buscado é o primeiro da lista.
- Pior Caso (Linear) – quando o elemento buscado não existe na lista.
- Médio Caso (Linear) – quando o elemento buscado é aleatório, podendo existir em qualquer posição na lista, bem como não existir.

Nos três casos, inicialmente são inseridos 100 elementos em uma lista encadeada em seguida são realizadas 100 buscas nessa lista, a média aritmética dessas 100 buscas é feita e o resultado é inserido em um arquivo contendo os tempos. Esse processo é realizado repetidas vezes onde a cada repetição o número de elementos inseridos aumenta em 100, isso é feito até o número de elementos na lista atingir o valor 9900, onde o processo é encerrado.

1.2 Gráfico do Tempo de Busca

Pode-se verifica que os casos variam de tempo constante a linear.


'./tempoBuscaListaEncadeada_Medio.txt'
'./tempoBuscaListaEncadeada_Melhor.txt'
'./tempoBuscaListaEncadeada_Pior.txt'
0.00004
0.00003
0.00002
0.00001
0.00000
0
20
40
60
80
100

2. ÁRVORE BINARIA

A árvore binária consiste em uma estrutura de dados onde cada elemento é denominado como nó, o primeiro nó inserido é chamado de raiz da árvore e seus consecutivos são chamados de filhos. Cada nó possui referência para seu filho da esquerda e direita. Um elemento é inserido a esquerda de um nó, quando o valor do nó for maior que o valor do elemento. Analogamente, um elemento é inserido a direita de um nó, quando o valor do nó for menor que o valor do elemento.

Em uma árvore binária, os elementos não necessariamente precisam está organizados de forma sequencial, podendo cada elemento está em uma posição qualquer na memória.

O mais importante em uma árvore binária é conhecer a referência para a raiz da árvore, pois a partir deste pode-se percorrer qualquer elemento na mesma.

2.1 Tempo de Busca

O tempo de busca em uma árvore binária depende de como os elementos foram inseridos e de qual elemento será buscado.

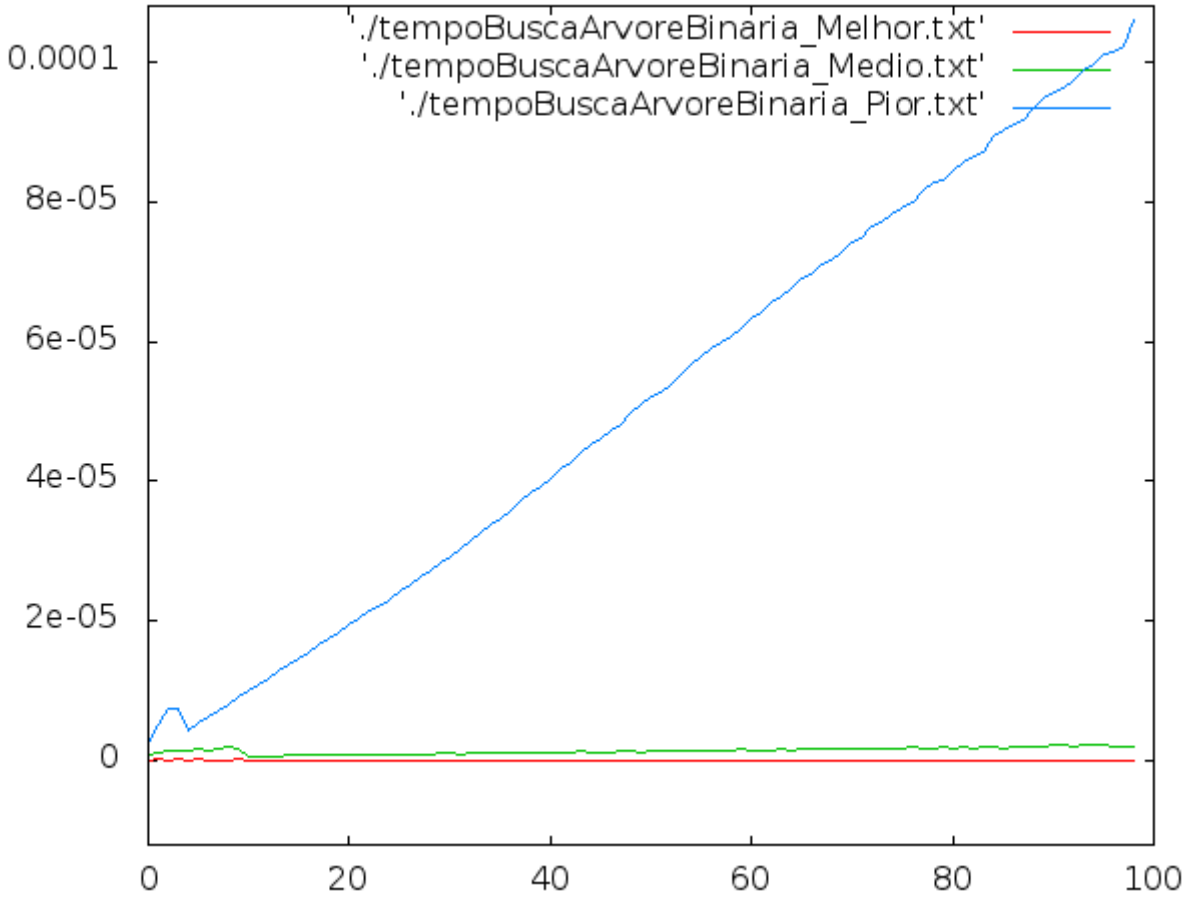
No exemplo desenvolvido, os elementos são inseridos de forma aleatória e buscados de três formas diferentes que são:

- Melhor Caso (Constante) – quando o elemento buscado for a raiz da árvore.
- Pior Caso (Linear) – quando o elemento buscado não existe na árvore.
- Médio Caso (Logaritmo) – quando o elemento buscado é aleatório, podendo existir em qualquer posição na árvore, bem como não existir.

Nos três casos, inicialmente são inseridos 100 elementos em uma árvore binária em seguida são realizadas 100 buscas nessa lista, a média aritmética dessas 100 buscas é feita e o resultado é inserido em um arquivo contendo os tempos. Esse processo é realizado repetidas vezes onde a cada repetição o número de elementos inseridos aumenta em 100, isso é feito até o número de elementos na árvore atingir o valor 9900, onde o processo é encerrado.

2.2 Gráfico do Tempo de Busca

Pode-se verificar que os casos variam de tempo constante a linear, no entanto, seu caso médio é da ordem logaritmo.


'./tempoBuscaArvoreBinaria_Melhor.txt'
'./tempoBuscaArvoreBinaria_Medio.txt'
'./tempoBuscaArvoreBinaria_Pior.txt'
0.0001
8e-05
6e-05
4e-05
2e-05
0
0
20
40
60
80
100

3. ÁRVORE AVL

A árvore AVL é uma estrutura de dados similar a árvore binária, no entanto, esta realiza alguns procedimentos para evitar um chamado desbalanceamento.

O desbalanceamento em uma árvore é um problema onde os elementos são inseridos mais em um lado da árvore do que no outro. Para se evitar esse problema procedimentos chamados de rotações são realizados com a finalidade de balancear a árvore.

Nessa estrutura, cada nó possui um valor que representa sua altura, em relação aos outros elementos. Para se verificar que uma árvore está desbalanceada existe uma regra que consiste em, cada nó verifica se a diferença da altura dos seus filhos for maior que 1, este nó está desbalanceado e precisa ser balanceado.

2.1 Tempo de Busca

O tempo de busca em uma árvore AVL depende de como os elementos foram inseridos e de qual elemento será buscado.

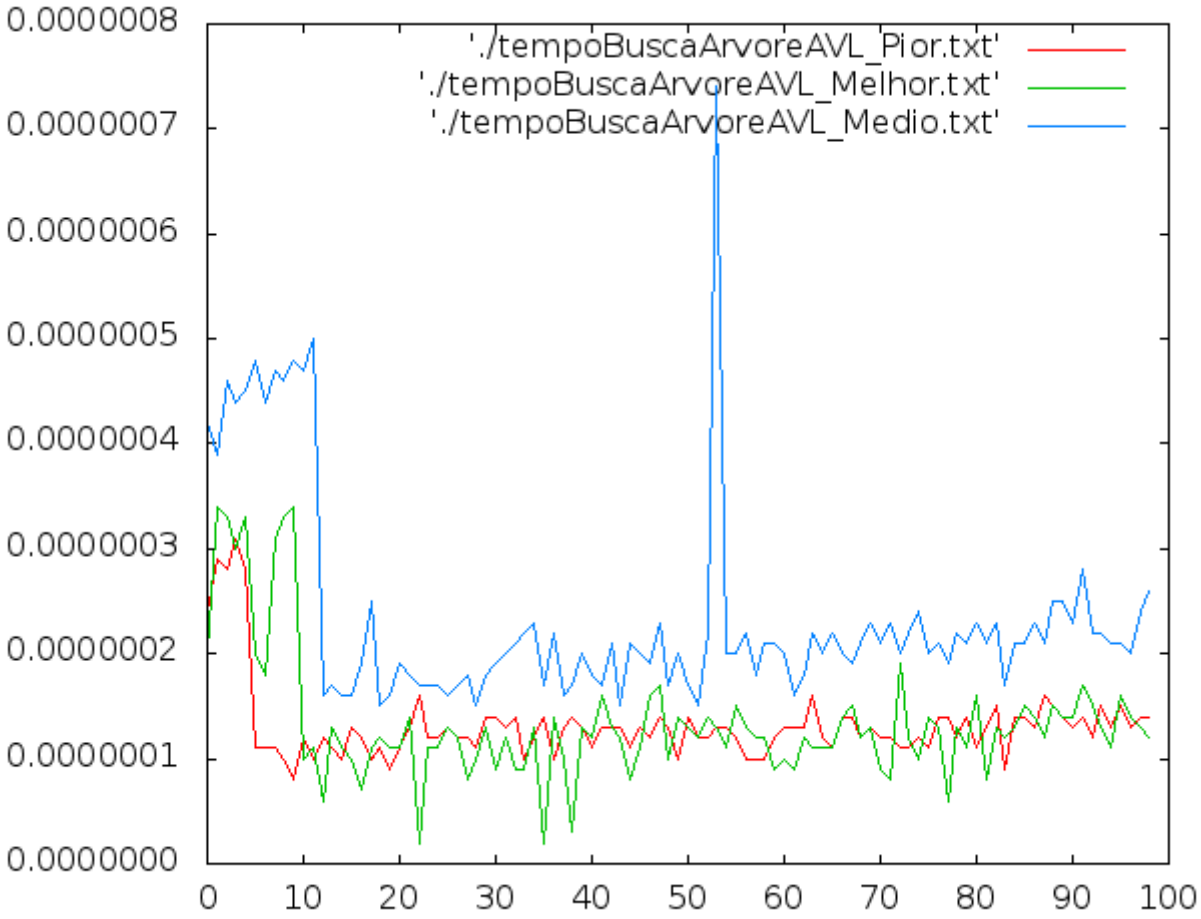
No exemplo desenvolvido, os elementos são inseridos de forma aleatória e buscados de três formas diferentes que são:

- Melhor Caso (Constante) – quando o elemento buscado for a raiz da árvore.
- Pior Caso (Logaritmo) – quando o elemento buscado não existe na árvore.
- Médio Caso (Logaritmo) – quando o elemento buscado é aleatório, podendo existir em qualquer posição na árvore, bem como não existir.

Nos três casos, inicialmente são inseridos 100 elementos em uma árvore AVL em seguida são realizadas 100 buscas nessa lista, a média aritmética dessas 100 buscas é feita e o resultado é inserido em um arquivo contendo os tempos. Esse processo é realizado repetidas vezes onde a cada repetição o número de elementos inseridos aumenta em 100, isso é feito até o número de elementos na árvore atingir o valor 9900, onde o processo é encerrado.

2.2 Gráfico do Tempo de Busca

Pode-se verificar que os casos variam de tempo constante a logaritmo.


'./tempoBuscaArvoreAVL_Pior.txt'
'./tempoBuscaArvoreAVL_Melhor.txt'
'./tempoBuscaArvoreAVL_Medio.txt'
0.0000008
0.0000007
0.0000006
0.0000005
0.0000004
0.0000003
0.0000002
0.0000001
0.0000000
0
10
20
30
40
50
60
70
80
90
100

4. TABELA HASH

A tabela hash consiste em uma estrutura de dados onde os elemento são organizados de forma parecida com a lista encadeada, no entanto, são criadas várias listas onde a referência para o inicio de cada uma delas é organizada de forma sequencial na memória, como um vetor.

Essa estrutura possui duas regras em particular. A primeira é a regra de dispersão, onde a inserção e busca de elementos obedece a lógica, resto da divisão do [valor do elemento] com o [tamanho do vetor], isso permite que na busca como na inserção a posição que o elemento ocupará obedecerá essa lógica. A segunda regra diz respeito ao tamanho do vetor, onde se a relação entre o número de elementos na tabela e o tamanho do vetor for maior que 1, o tamanho do vetor deve ser dobrado e todos os elementos reinseridos nessa nova tabela.

Em uma tabela hash, os elementos não necessariamente precisam está organizados de forma sequencial, podendo cada elemento está em uma posição qualquer na memória, no entanto, as referências para o inicio de cada lista deve ser sequencial.

O mais importante em uma tabela hash é conhecer a referência para o inicio do vetor, pois a partir deste pode-se percorrer qualquer elemento na mesma.

2.1 Tempo de Busca

O tempo de busca em uma tabela hash depende de como os elementos foram inseridos e de qual elemento será buscado.

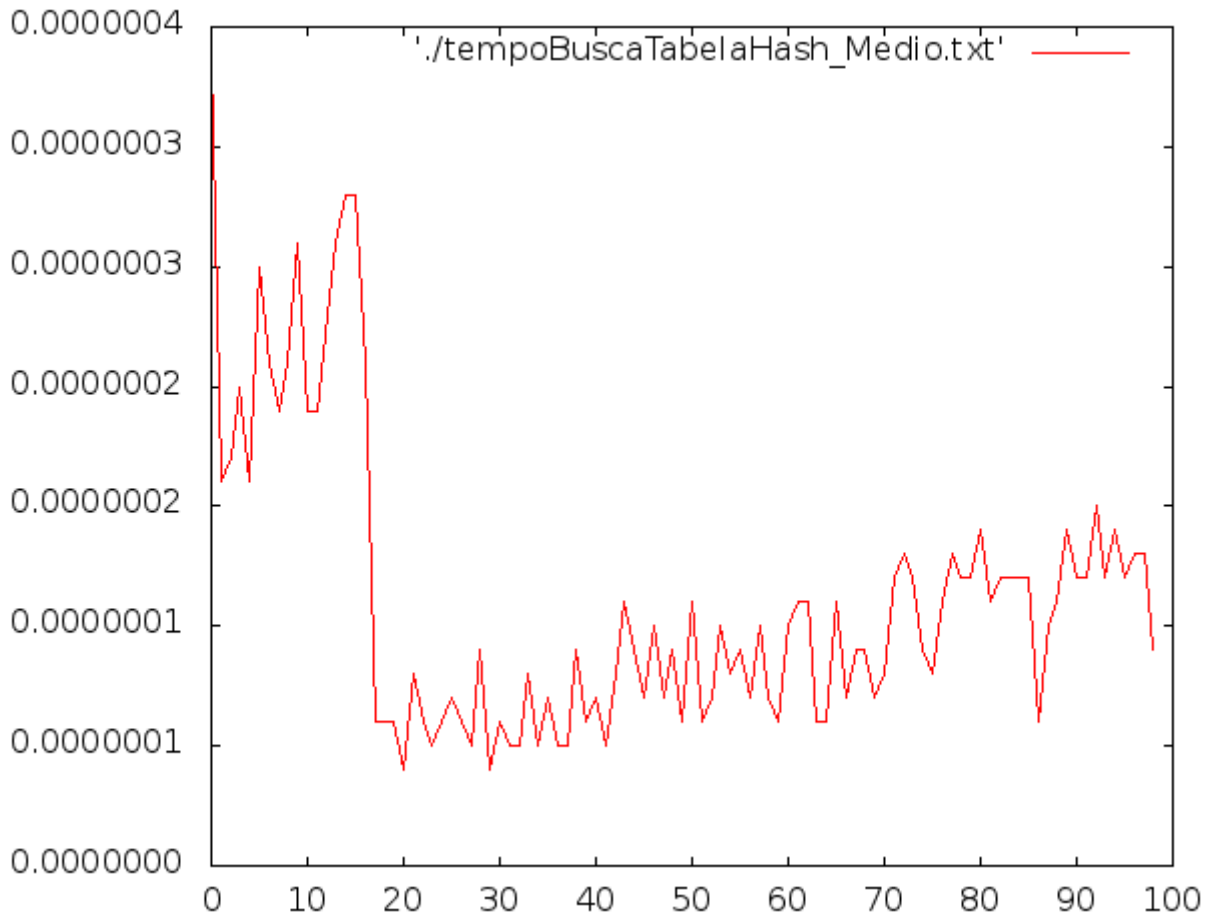
No exemplo desenvolvido, os elementos são inseridos de forma aleatória e buscados de uma forma que é:

- Médio Caso (Constante) – quando o elemento buscado é aleatório, podendo existir em qualquer posição da tabela, bem como não existir.

Nos três casos, inicialmente são inseridos 100 elementos em uma tabela hash em seguida são realizadas 100 buscas nessa tabela, a média aritmética dessas 100 buscas é feita e o resultado é inserido em um arquivo contendo os tempos. Esse processo é realizado repetidas vezes onde a cada repetição o número de elementos inseridos aumenta em 100, isso é feito até o número de elementos na árvore atingir o valor 9900, onde o processo é encerrado.

2.2 Gráfico do Tempo de Busca

Pode-se verificar que o caso médio é de ordem logaritmo.



5. COMPARAÇÃO

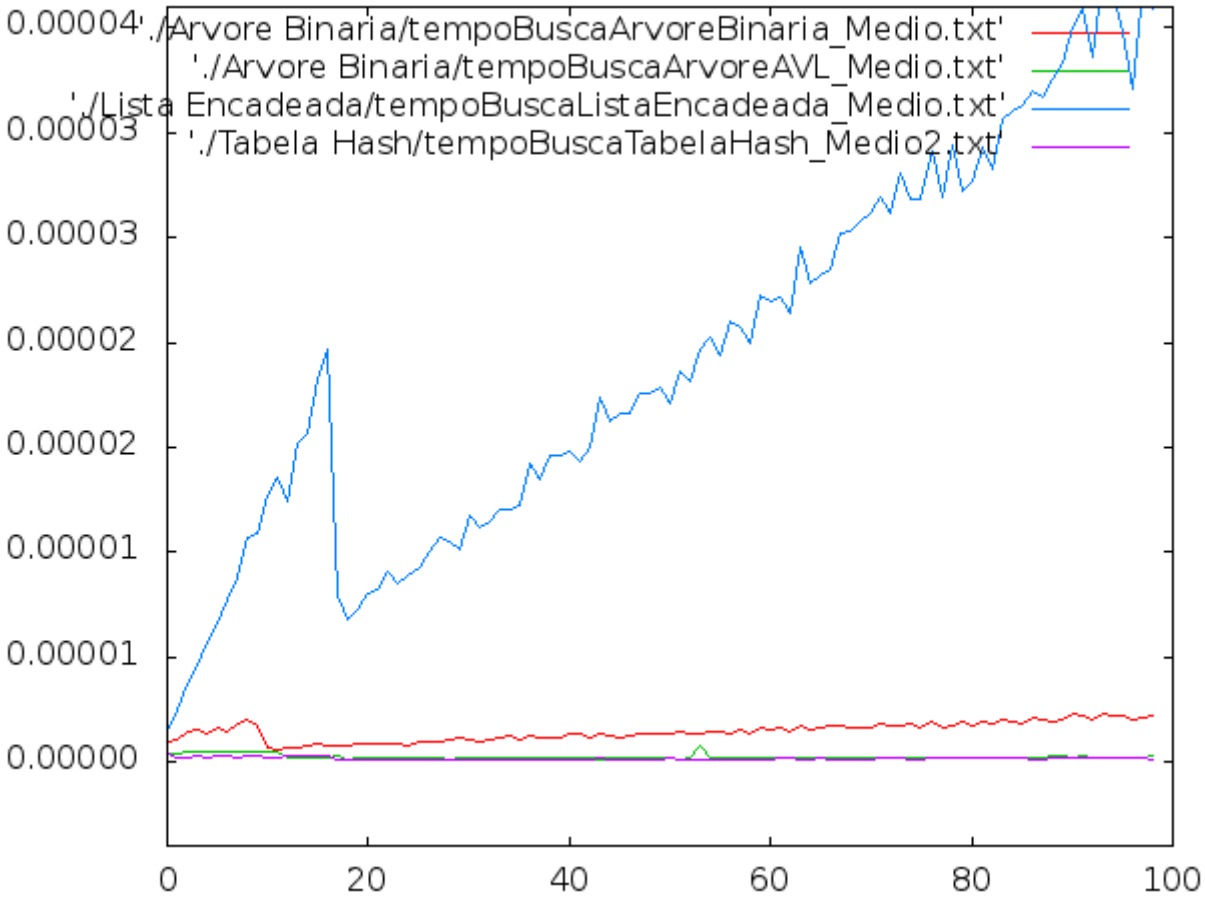
Nesse tópico é realizada uma comparação dos tempos de execução das estruturas estudadas. A melhor forma para se realizar essa comparação é verificando um gráfico contendo os tempos de execução dos casos médio dos algoritmos.

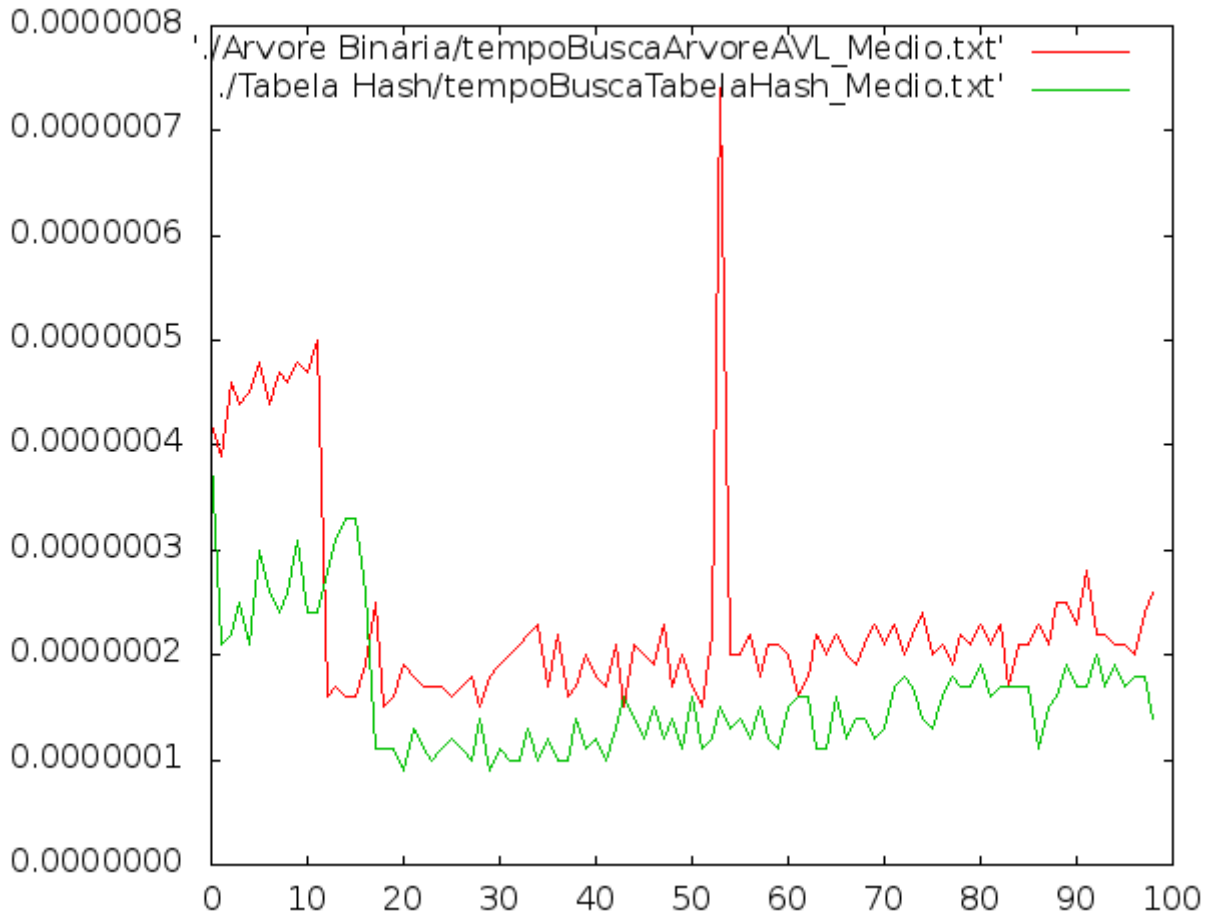
5.1 Gráfico com o tempo de execução

Pode-se verificar que os algoritmos variam de ordem logaritmo a linear, nessa ordem:

- Tabela Hash – Ordem Constante
- Árvore AVL – Ordem Logaritma
- Árvore Binária – Ordem Logaritma
- Lista Encadeada – Ordem Linear

Sendo nesse experimento a Tabela Hash o algoritmo com tempo de busca mais rápido.




'./Arvore Binaria/tempoBuscaArvoreAVL_Medio.txt'
'./Tabela Hash/tempoBuscaTabelaHash_Medio.txt'
0.0000008
0.0000007
0.0000006
0.0000005
0.0000004
0.0000003
0.0000002
0.0000001
0.0000000
0
10
20
30
40
50
60
70
80
90
100