Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

JANETO ERICK DA COSTA LIMA

JÚLIO CÉSAR DA COSTA MORAIS

**ESTRUTURA DE DADOS BASICA I: ANÁLISE EMPÍRICA**

Natal – RN

2019

**Sumário**

[**1.** **INTRODUÇÃO** 3](#_Toc4160976)

[**2.** **MÉTODO** 4](#_Toc4160977)

[**2.1.** **Materiais** 4](#_Toc4160978)

[*2.1.1.* *Algoritmos implementados* 4](#_Toc4160982)

[**2.2.** **Metodologia** 6](#_Toc4160983)

[**3.** **RESULTADOS E DISCUSSÃO** 7](#_Toc4160984)

[**3.1.** **Algoritmos Lineares** 7](#_Toc4160985)

[**3.2.** **Recursão x Iteração** 8](#_Toc4160986)

[**3.3.** **Tamanho da Partição e sua Inﬂuência no Desempenho** 8](#_Toc4160987)

[**3.4.** **Algoritmos de Classes de Complexidade Diferentes** 9](#_Toc4160994)

[**3.5.** **O Pior Caso da Busca Fibonacci** 9](#_Toc4160996)

[**4.** **CONCLUSÃO** 10](#_Toc4160997)

# **INTRODUÇÃO**

Quando apenas executamos um algoritmo e o utilizamos, não há noção alguma por parte de quem programa se o algoritmo está utilizando muita memória do computador (mais que a necessária), ou demorando muito tempo para ser executado. Para isso, utilizando da análise empírica, vamos medir a eficiência dos algoritmos que são programados de formas diferentes, como por exemplo um recursivo, e um iterativo, e através disso, saber qual o mais apropriado a ser usado.

No trabalho em questão, foram feitos diversos testes com algoritmos de busca, mais especificamente em 7 diferentes, a busca binária recursiva e iterativa, busca linear iterativa, busca ternária iterativa e recursiva, busca Jump Search, e por fim, busca Fibonacci. Todos os algoritmos foram testados no mesmo computador, e submetidos a testes iguais, para assim, medirmos a eficiência dos mesmos.

# **MÉTODO**

## **Materiais**

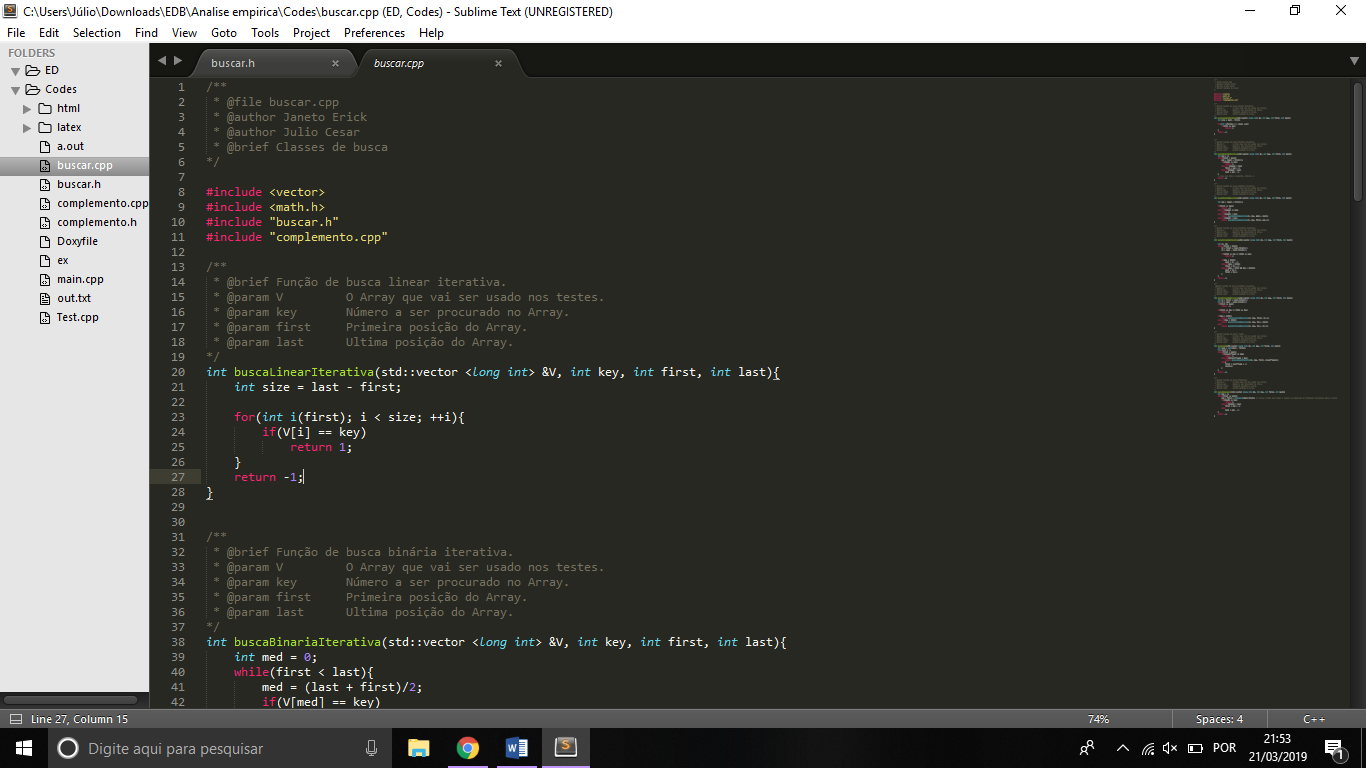
Os materiais utilizados para a realização do projeto foram:

* Processador: Intel Core i7-3537U CPU @ 2.00 GHz x 4;
* Memória RAM: 8GB de memória RAM;
* HD: 1TB de HD total, com 856.6 GB disponíveis;
* Linguagem de programação: C++ 11;
* Sistema Operacional: Linux Mint 18.1 Serena 64-bits;
* Compilador: GNU Compiler Collection (G++), na versão 5.4.0;
* Aplicativo utilizado para programar: Visual Studio Code;
* Ferramenta para gerar gráficos: GNUPLOT



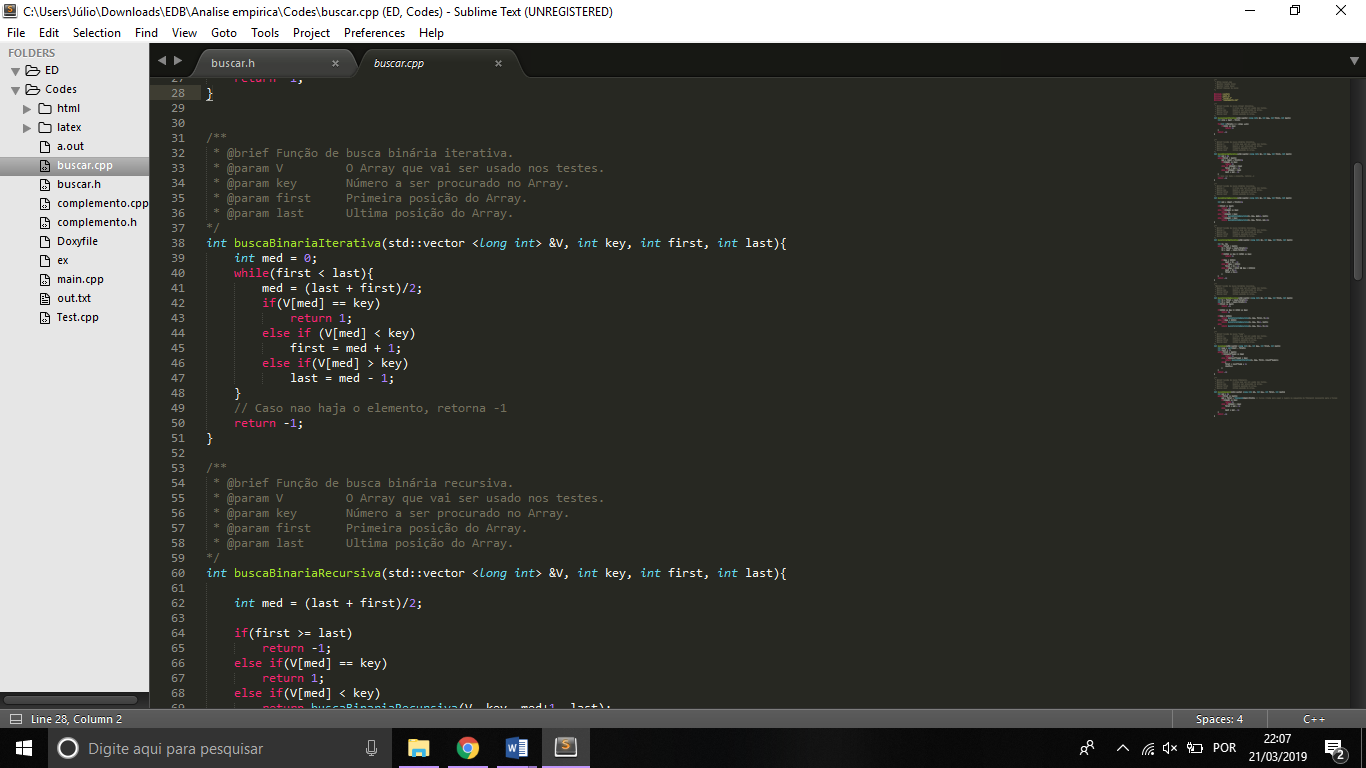
### *Algoritmos implementados*

Figura 1 – Busca Linear Iterativa



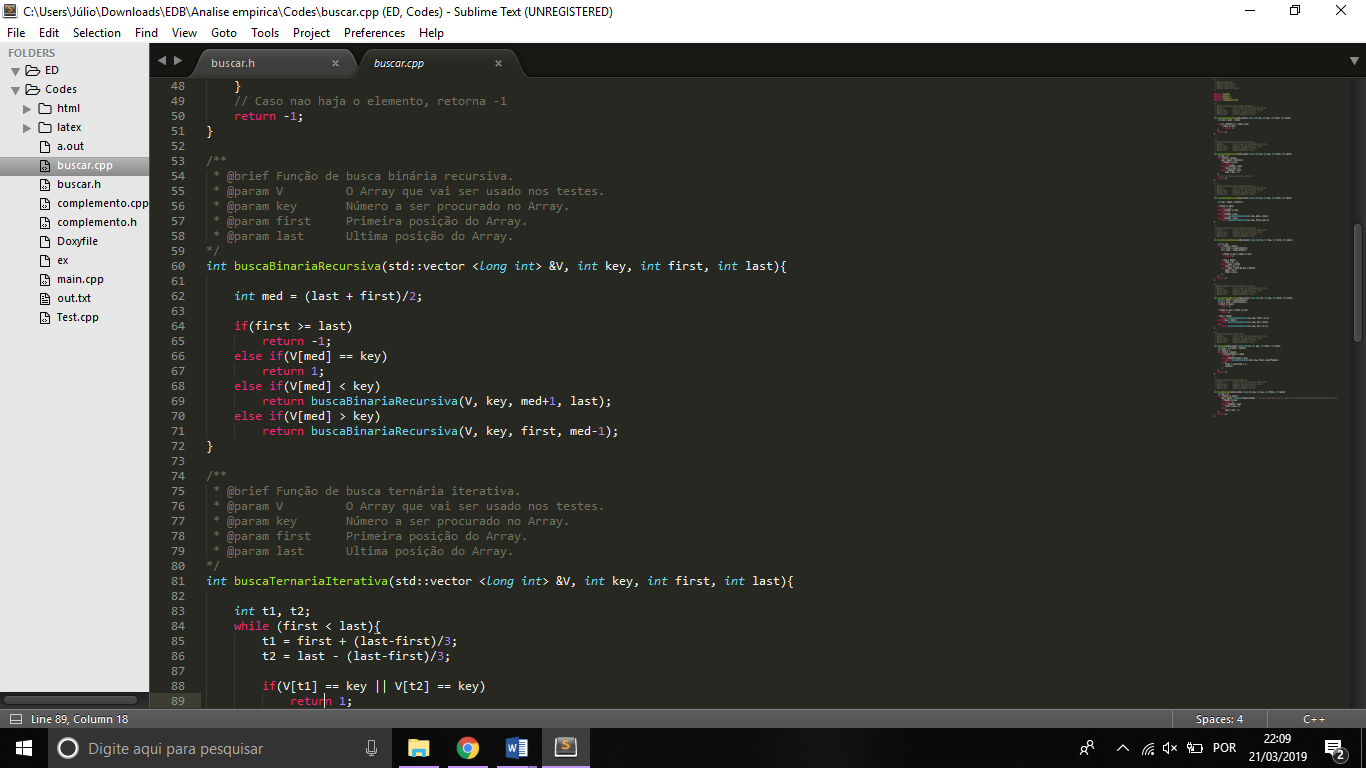
Fonte – Autoria Própria

Figura 2 – Busca Binária Iterativa

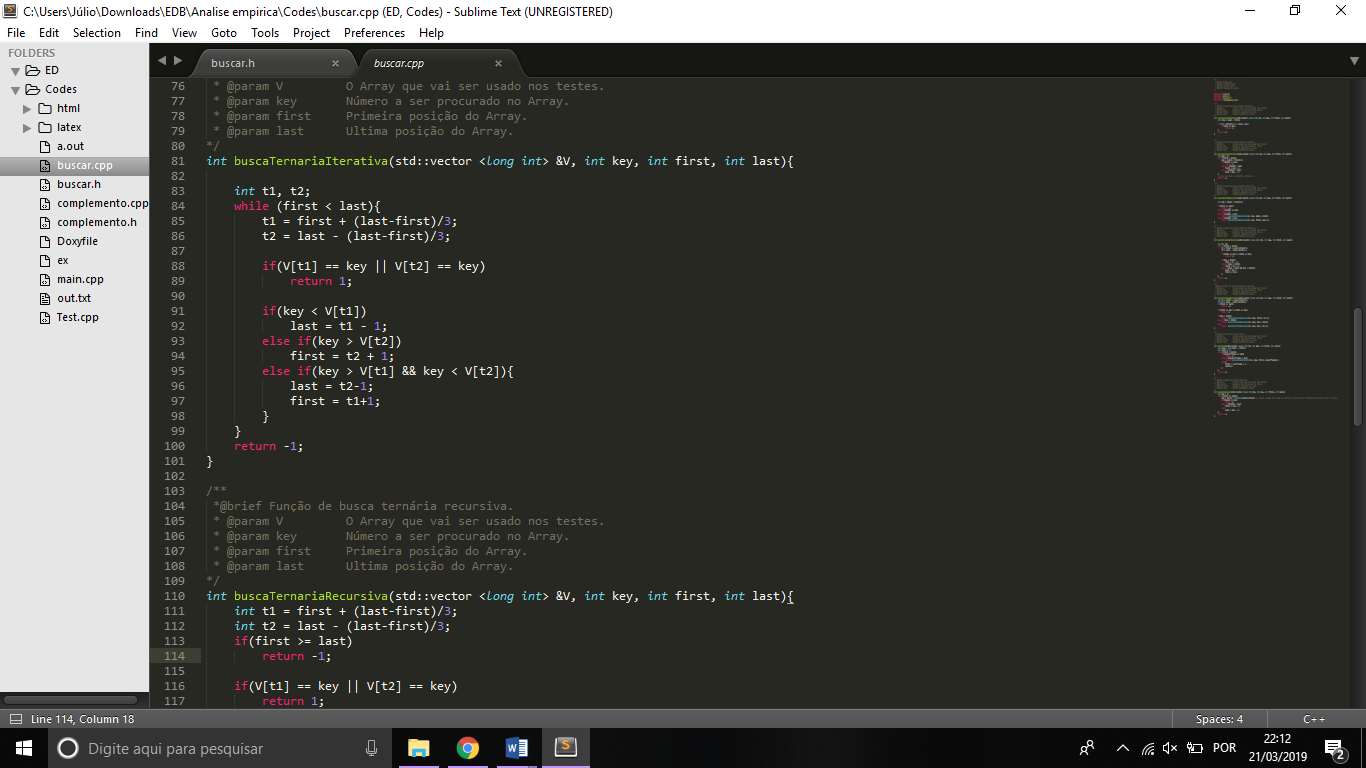


Fonte – Autoria Própria

Figura 3 – Busca Binária Recursiva

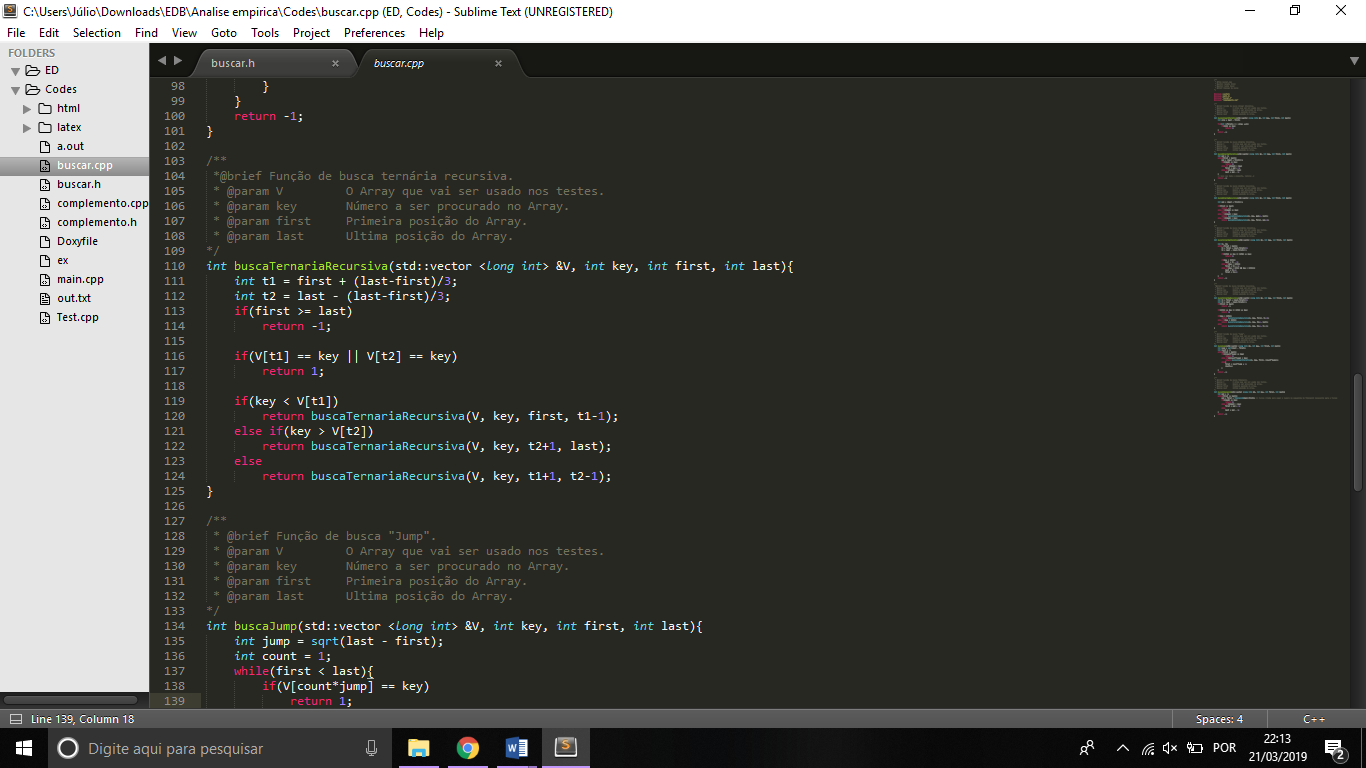


Fonte – Autoria Própria

Figura 4 – Busca Ternária Iterativa

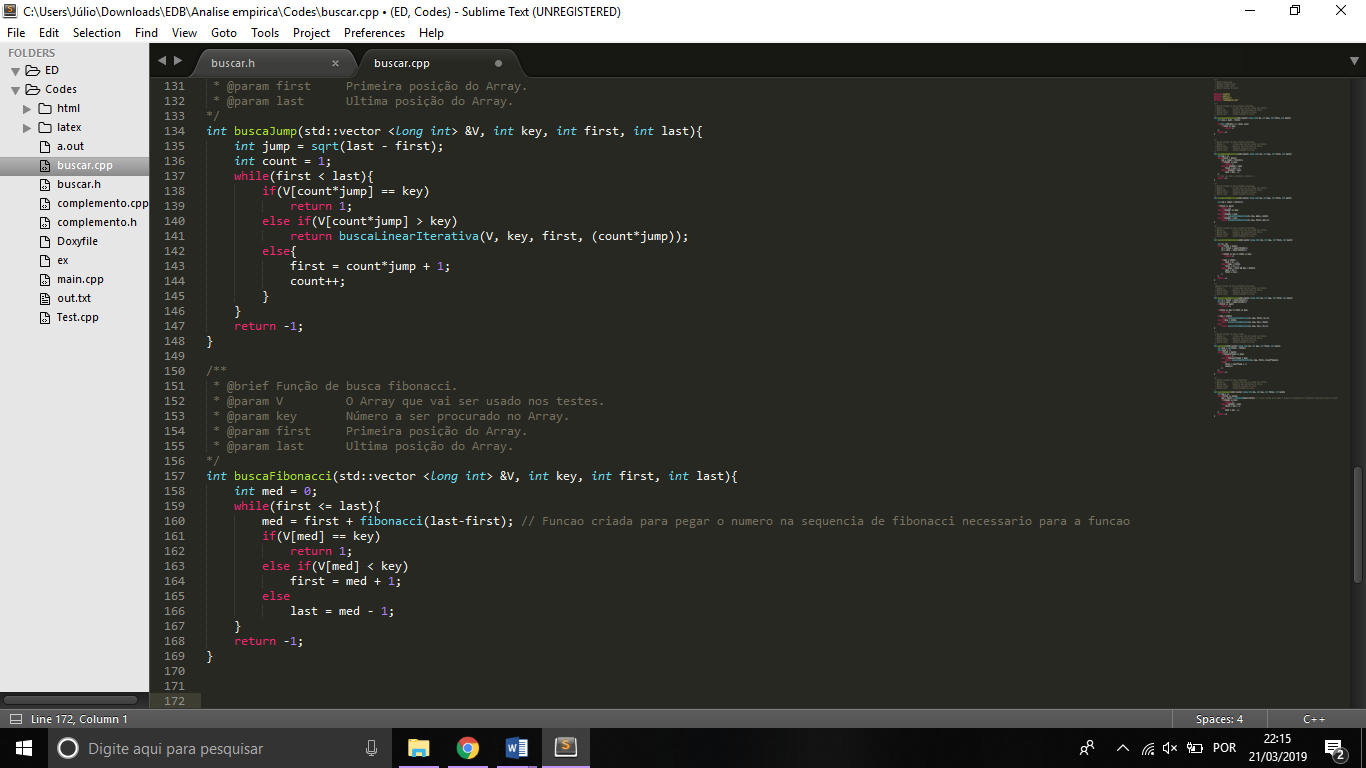
Fonte – Autoria Própria

Figura 5 – Busca Ternária Recursiva



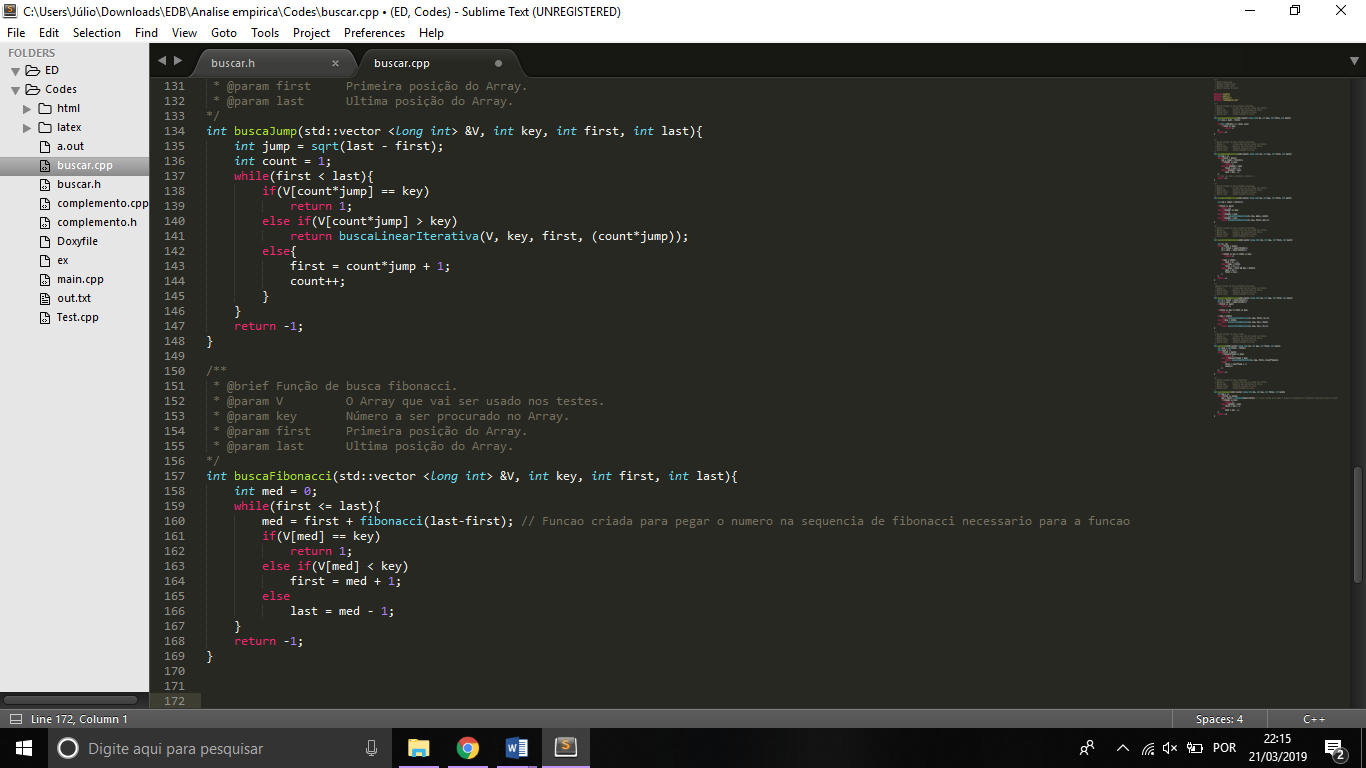
Fonte – Autoria Própria

Figura 6 – Busca Jump



Fonte – Autoria Própria

Figura 7 – Busca Fibonacci



Fonte – Autoria Própria

* Cenários considerados: Foram considerados os cenários de melhor caso e pior caso para todos os 7 algoritmos.

## **Metodologia**

Inicialmente, são gerados dados de forma sequencial e definida. O vetor tem seu tamanho definido por uma variável que o eleva ao expoente de 2.

Todos os algoritmos que foram testados no projeto tiveram de ser comparados pelo tempo de execução, para medir a eficiência. Esse tempo era medido através da biblioteca std::chrono, e cada algoritmo era medido a cada nova mudança de tamanho do vetor, que aumentava exponencialmente. Por fim, para fazer as comparações pedidas no roteiro do projeto, utilizamos do gnuplot, uma ferramenta que nos permitiu gerar gráficos, para que a comparação ficasse mais clara, e assim, analisarmos qual algoritmo é mais eficiente.

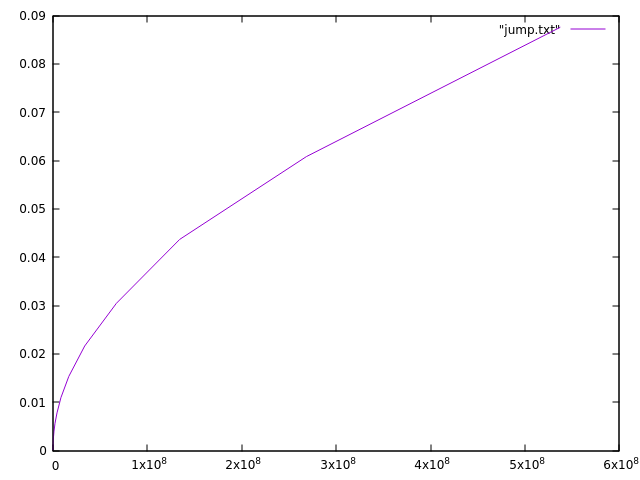
# **RESULTADOS E DISCUSSÃO**

Nessa seção, vamos apresentar os gráficos de comparação de desempenho de acordo com o que foi proposto, e responder os questionamentos feitos.

## **Algoritmos Lineares**

Vejamos abaixo o gráfico 1, que mostra a comparação entre o pior caso da busca Jump Search e da busca Linear Iterativa.

Grafico 1 – Pior caso Jump Search x Linear Iterativa

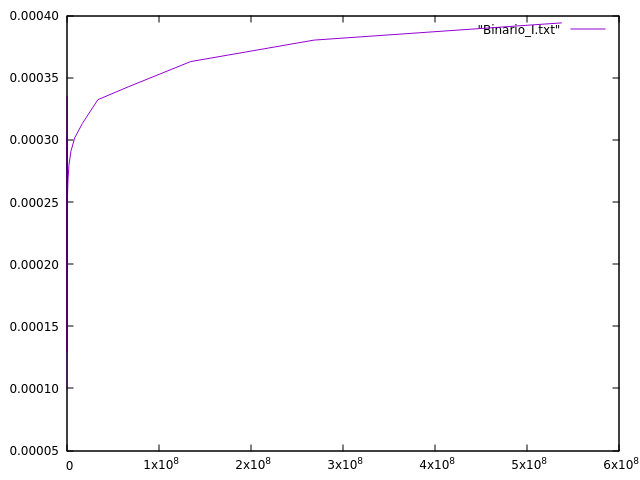


Fonte – Autoria Própria

## **Recursão x Iteração**

No gráfico dois, a comparação entre o algoritmo de busca binária em suas duas formas, Iterativa e Recursiva

Grafico 2 – busca Binária Iterativa x Recursiva

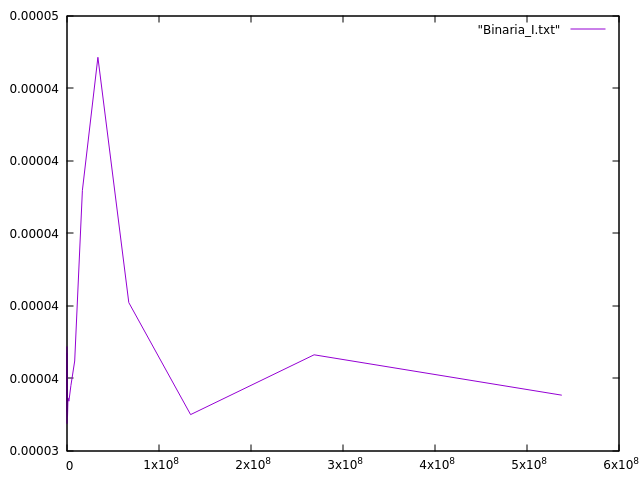


Fonte – Autoria Própria

## **Tamanho da Partição e sua Inﬂuência no Desempenho**

Vejamos a seguir o gráfico comparativo das buscas binárias, ternárias e Fibonacci, apenas nas versões iterativas.

Gráfico 3 – Binária x Ternária x Fibonacci iterativas



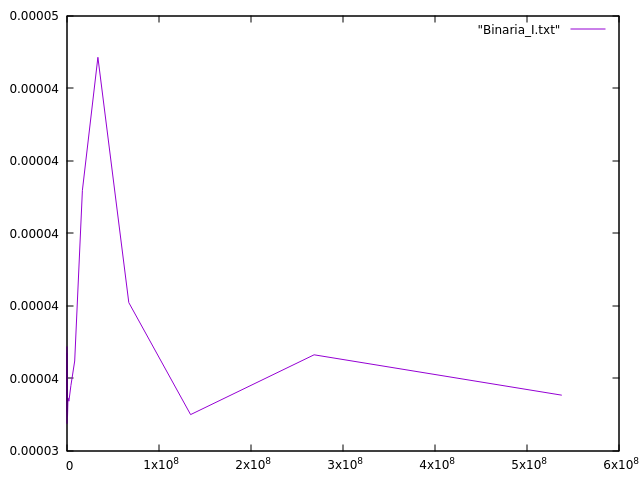
Fonte – Autoria Própria



## **Algoritmos de Classes de Complexidade Diferentes**

Vejamos a seguir no gráfico 4, a comparação de desempenho da busca binária Iterativa e da busca linear iterativa.

Grafico 4 – Busca Binária Iterativa x Busca Linear Iterativa



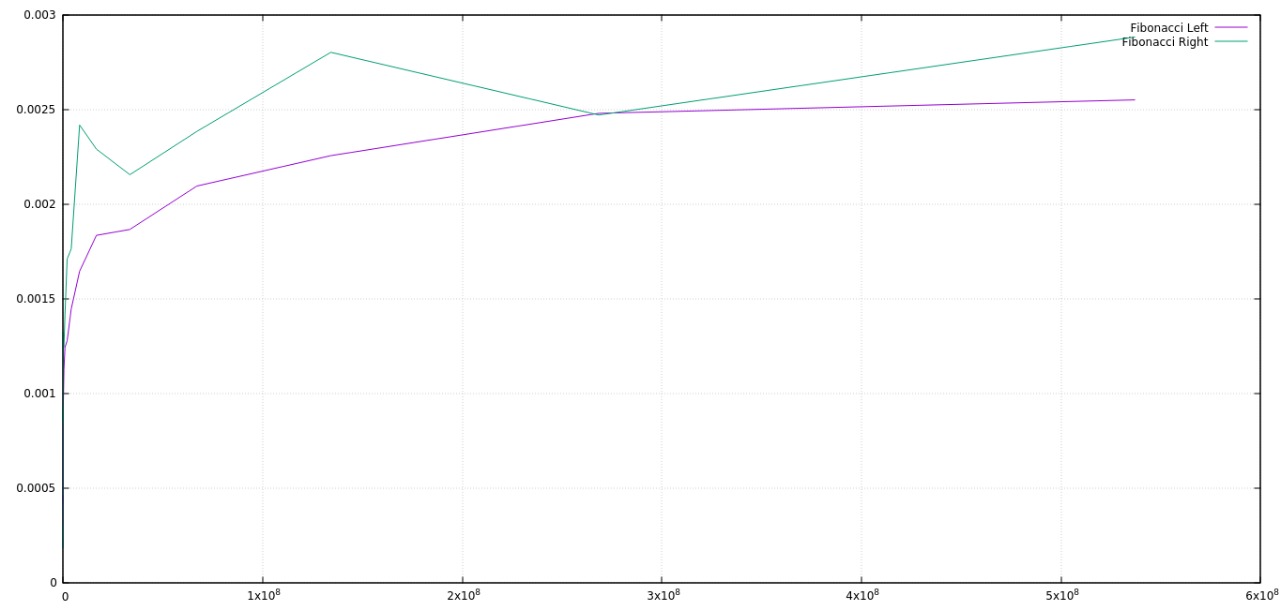
Fonte – Autoria Própria



## **O Pior Caso da Busca Fibonacci**

Por fim, no gráfico 5, o pior caso de Fibonacci sendo comparado, sendo um vindo pela direita, e o outro pela esquerda.

Grafico 5 – Fibonacci Left x Fibonacci Right



Fonte – Autoria Própria

# **CONCLUSÃO**

Diante do que foi apresentado no relatório, podemos observar o porquê da análise empírica dos algoritmos, e da realização deste projeto em questão serem importantes, visto que através da pratica, ocorre um aprofundamento de tudo o que foi visto durante as aulas, além de aprender de vez a medir a eficiência dos programas os quais fazemos, considerando casos diferentes para cada algoritmo, e, sendo assim, saber qual o melhor para se utilizar.