**UNIVERSIDADE FEDERAL DO CEARÁ – CAMPUS SOBRAL**
**CURSO DE ENGENHARIA DE COMPUTAÇÃO**

**DISCIPLINA: TÓPICOS ESPECIAIS EM COMPUTAÇÃO II**

**PROFESSORA: FISCHER JONATAS FERREIRA**

# TRABALHO ESQUENTA - Algoritmos de Busca

| ALUNO | MATRÍCULA |
| --- | --- |
| **Francisco Cassiano de Vasconcelos Souza** | **413067** |

**Sobral – CE**

**2023**

**SUMÁRIO**

## 1. INTRODUÇÃO

A busca é uma tarefa comum em muitos aspectos da vida cotidiana, e os algoritmos de busca computacional têm como objetivo automatizar e otimizar esse processo. Os algoritmos de busca são uma parte fundamental da ciência da computação que desempenham um papel crucial na resolução de uma ampla variedade de problemas. Eles são projetados para encontrar informações específicas em conjuntos de dados, facilitando a recuperação de dados relevantes e a tomada de decisões informadas. Portanto, iremos trabalhar com os principais algoritmos de busca:

- Busca Sequencial;
- Busca Binária;
- Busca Quadrática;
- Busca Cúbica;
- Busca Ternária;

A complexidade dos algoritmos de busca é um tópico importante na análise e no desenvolvimento desses algoritmos. Existem duas medidas principais de complexidade associadas a algoritmos de busca:

**Complexidade Temporal:** A complexidade temporal é frequentemente expressa em termos de notação "O" (Big O), que descreve a taxa de crescimento do tempo de execução em relação ao tamanho da entrada.

**Complexidade Espacial:** Isso se refere à quantidade de memória (espaço de armazenamento) que um algoritmo de busca utiliza à medida que processa os dados de entrada. À semelhança da complexidade temporal, a complexidade espacial é expressa em notação "O".

## 2. MÉTODO

Para a execução dos algoritmos foi escolhida a linguagem de programação Python pela facilidade de uso e por uma gama de bibliotecas disponíveis adequadas para medição de desempenho.

A máquina utilizada para a análise de desempenho dos algoritmos de busca portava um processador **Intel(R) Core(TM) i5-1145G7**, memória RAM **8,00 GB**.

Os arquivos utilizados como fonte de dados para o experimento e análise foram disponibilizados na atividade, porém podem ser gerados novos arquivos com os arquivos geradores **geraArquivoDesordenado.py** e **geraArquivoOrdenado.py.**

A análise de desempenho do algoritmo é realizada através da função disponível no arquivo **./module/valoresMetricas.py**.

```
        total_tempo = 0

        total_memoria = 0

        tracemalloc.start()

        for _ in range(iteracoes):

            inicio_tempo = time.time()

            codigo()  # Executa o trecho de código fornecido

            fim_tempo = time.time()

            tempo_gasto = fim_tempo - inicio_tempo

            total_tempo += tempo_gasto

            snapshot = tracemalloc.take_snapshot()

            memoria_usada = sum(stat.size for stat in
snapshot.statistics("lineno"))

            total_memoria += memoria_usada

        media_tempo = total_tempo / iteracoes

        media_memoria = total_memoria / iteracoes
```

O código inicia um cronômetro utilizando **time.time()** e um rastreador de alocação de memória usando **tracemalloc.start()**, executa o trecho de código pela função **codigo()** as vezes determinadas pela variável **iteracoes**, medindo o tempo gasto e uso de memória em cada execução.

## 3. DESENVOLVIMENTO

Será realizada uma análise de desempenho dos seguintes algoritmos de busca:

- Busca Sequencial v1;
- Busca Sequencial v2;
- Busca Binária;
- Busca Quadrática;
- Busca Cúbica;
- Busca Ternária;

Utilizou-se 13 instâncias de dados com tamanhos variados que foram gerados pelos algoritmos anteriormente citados, **geraArquivoDesordenado.py** e **geraArquivoOrdenado.py**. Foram usados tantos os arquivos **ordenados** quanto os **desordenados**.

### 3.1 Busca Sequencial O(n)

A busca linear é um algoritmo simples de busca em que você percorre uma lista de elementos, um por um, até encontrar o elemento desejado ou determinar que ele não está presente na lista.

#### 3.1.1 Busca Sequencial v1

Este é um exemplo simples de busca sequencial, que é fácil de entender e implementar, mas pode ser ineficiente para grandes conjuntos de dados, pois verifica cada elemento na sequência.

#### 3.1.2 Busca Sequencial v2

A principal diferença entre a v2 e v1 está no momento em que a função retorna o valor. Na v2, assim que o alvo é encontrado, a função retorna imediatamente o índice, encerrando a busca. Por outro lado, na v1, a função continua a percorrer todo o vetor mesmo depois de encontrar o alvo, atualizando o índice a cada ocorrência do alvo até o final do vetor.

#### 3.1.3 Consumo de Tempo e Memória

Nas duas versões do algoritmo de Busca Sequencial foi utilizado o tipo de busca desordenado. Para cada instância foram realizadas 100 iterações, alternando o alvo a ser encontrado.

Ambas as funções de busca sequencial têm um tempo de execução linear em relação ao tamanho do vetor. Isso significa que, à medida que o tamanho do vetor aumenta, o tempo necessário para realizar a busca aumenta proporcionalmente. A cada elemento adicional no vetor, um número constante de comparações precisa ser feito.

Para vetores pequenos ou com poucos elementos, o tempo de execução é geralmente insignificante, e ambas as funções de busca sequencial são eficazes. No entanto, à medida que o vetor cresce, a busca sequencial pode se tornar relativamente lenta. Porém a Busca Sequencial v2, tem uma ligeira eficiência em tempo em comparação com a v1.

O consumo de memória das funções de busca sequencial é constante, pois não depende do tamanho do vetor. Essas funções não alocam memória adicional com base no tamanho do vetor. Podemos notar, através do gráfico abaixo:

**Figura 01**- Consumo memória Busca Sequencial v1

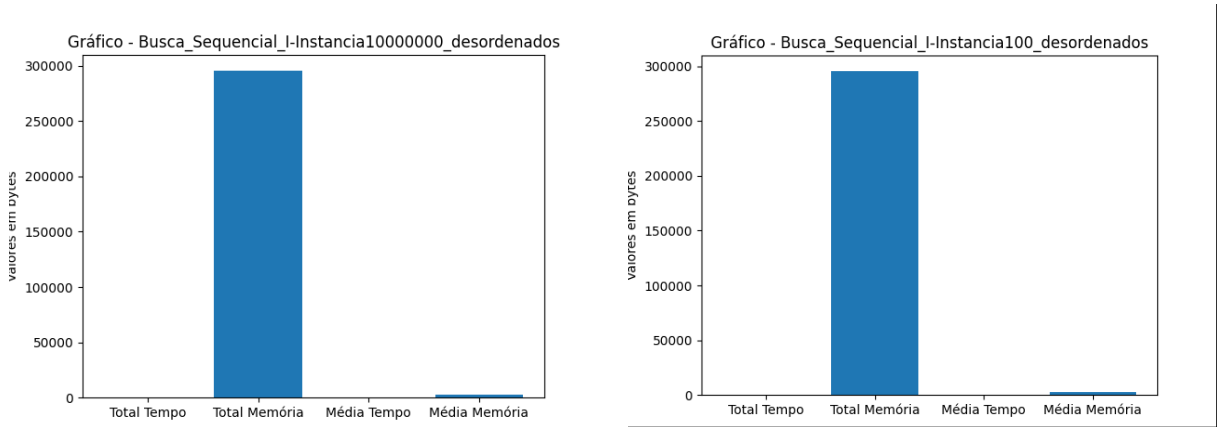
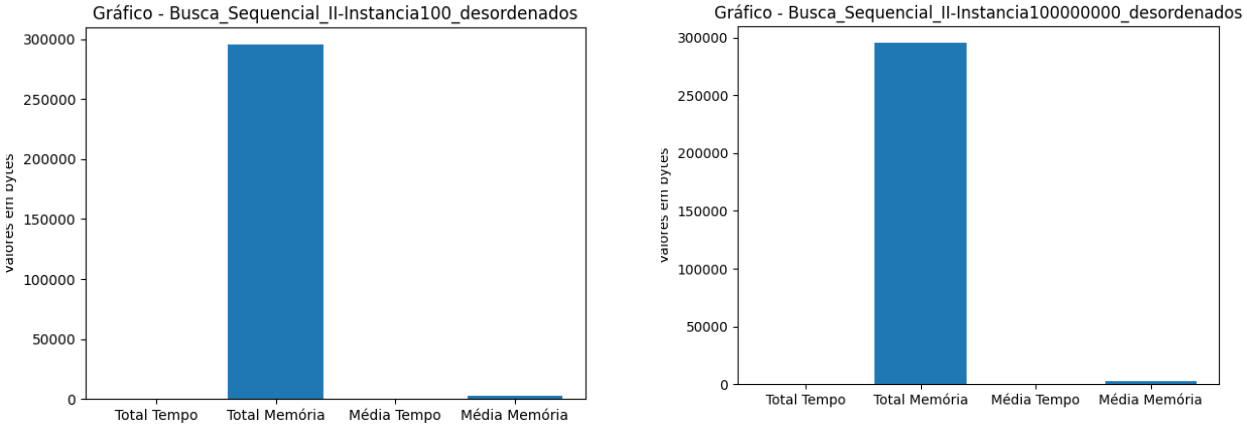



**Figura 02**- Consumo memória Busca Sequencial v2



### 3.2 Busca Binária O(log n)

A busca binária é um algoritmo de busca eficiente que é usado para encontrar um elemento específico em uma lista ordenada. Ela aproveita o fato de que a lista está ordenada e reduz drasticamente o número de comparações necessárias em comparação com a busca sequencial.

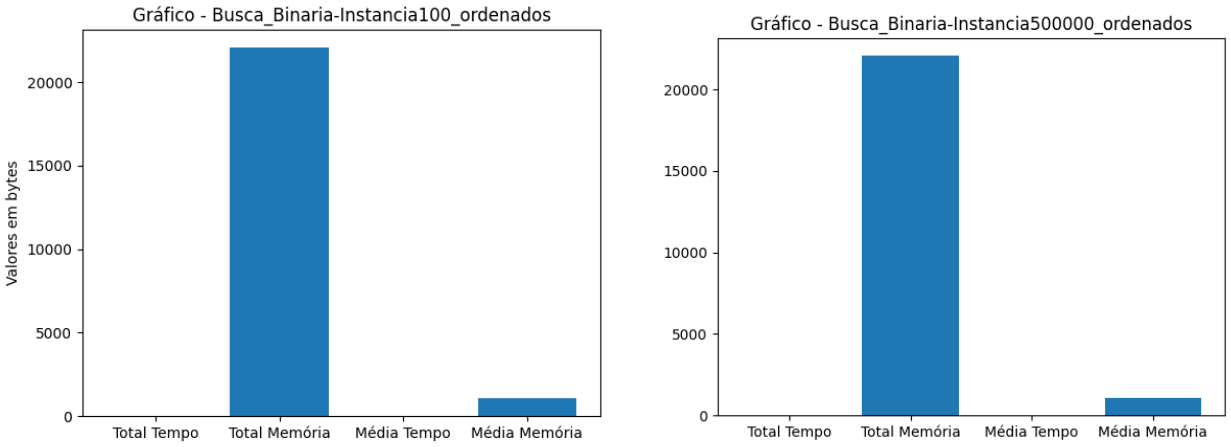
Neste algoritmo foi utilizado o tipo de lista ordenada para as 13 instâncias. Para cada instância foram realizadas 20 iterações, alternando o alvo a ser encontrado.

À medida que o tamanho do vetor aumenta, o tempo de execução cresce muito lentamente. A busca binária é significativamente mais rápida do que a busca sequencial (O(n)) para vetores grandes.

O consumo de memória é determinado principalmente pela quantidade de memória necessária para armazenar o vetor de entrada. O algoritmo de busca binária em si não requer memória adicional significativa

Portanto, o consumo de memória é essencialmente constante, independentemente do tamanho do vetor de entrada, como mostra os gráficos abaixo

**Figura 03**- Consumo memória Busca Binária



### 3.3 Busca Quadrática O(n^2)

Uma busca quadrática é um termo usado para descrever algoritmos de busca que têm um desempenho proporcional ao quadrado do tamanho dos dados de entrada. Isso significa que o tempo de execução e o consumo de memória aumentam significativamente à medida que o tamanho dos dados aumenta.

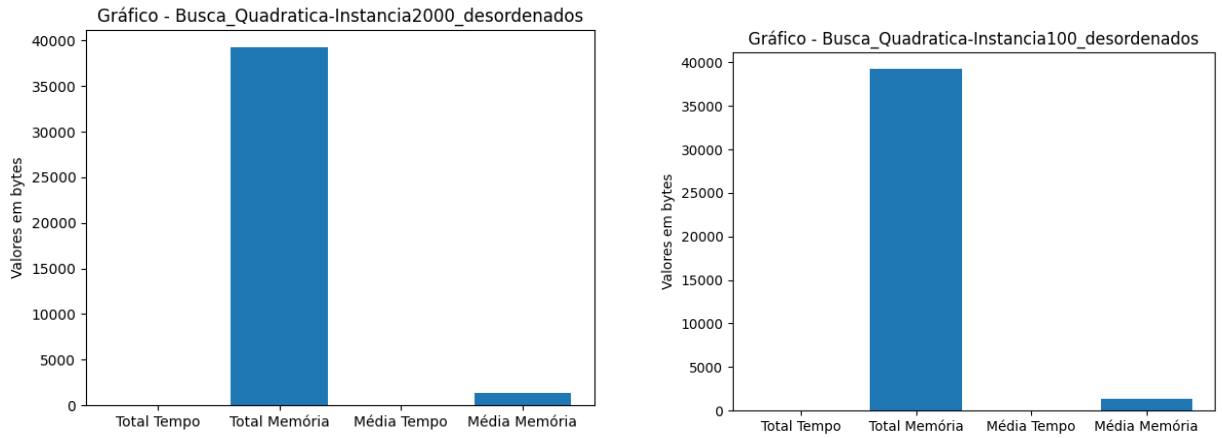
Neste algoritmo foi utilizado o tipo de lista ordenada para as 13 instâncias. Para cada instância foram realizadas 30 iterações, alternando o alvo a ser encontrado.

À medida que o tamanho do vetor aumenta, o tempo de execução cresce rapidamente. A busca quadrática é muito menos eficiente do que a busca binária, especialmente para vetores grandes.

O consumo de memória é determinado principalmente pela quantidade de memória necessária para armazenar o vetor de entrada.O algoritmo em si não requer memória adicional significativa além de algumas variáveis de controle.

Portanto, o consumo de memória é essencialmente constante e não depende do tamanho do vetor de entrada, como mostram os gráficos da figura.

**Figura 04**- Consumo memória Busca Quadrática




### 3.4 Busca Cúbica O(n^3)

A busca cúbica se refere a algoritmos de busca que têm um tempo de execução de ordem cúbica em relação ao tamanho dos dados de entrada. Isso significa que o tempo de execução aumenta com o cubo do tamanho dos dados.

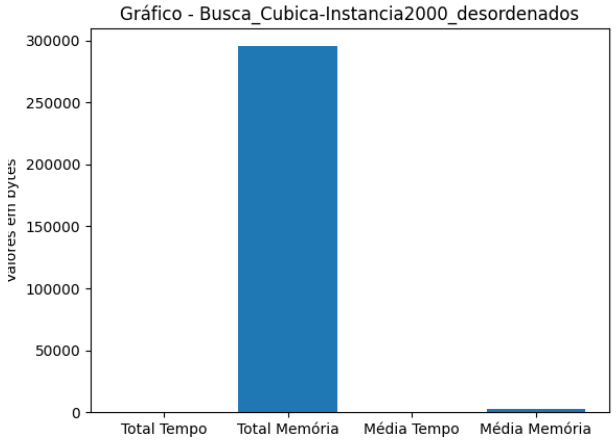
Neste algoritmo foi utilizado o tipo de lista ordenada para as 13 instâncias. Para cada instância foram realizadas 100 iterações, alternando o alvo a ser encontrado.

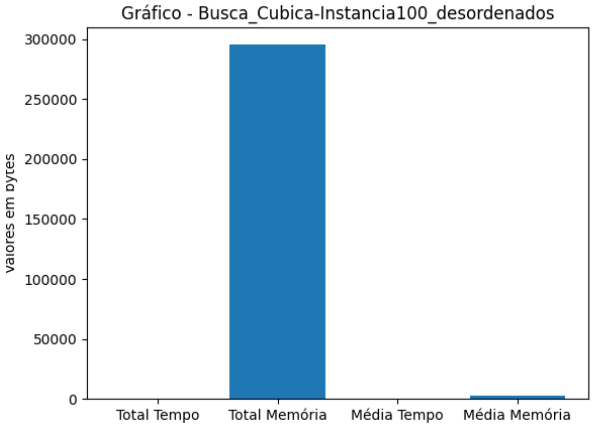
À medida que o tamanho do vetor aumenta, o tempo de execução cresce de forma cúbica. Portanto, o algoritmo se torna extremamente lento para vetores grandes.

O pior caso ocorre quando não há combinações válidas no vetor, resultando em um tempo de execução máximo.

O consumo de memória é determinado principalmente pelo tamanho do vetor de entrada. O algoritmo em si não requer memória adicional significativa. O consumo de memória é essencialmente constante e não depende do tamanho do vetor de entrada.

**Figura 05**- Consumo memória Busca Quadrática




### 3.5 Busca Ternária O(log n)

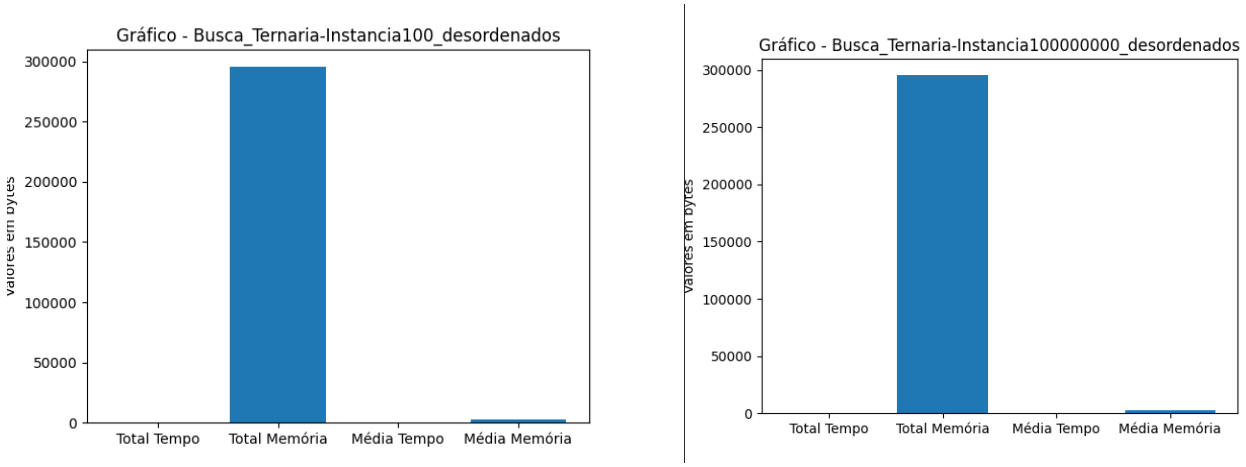
A busca ternária é um algoritmo de busca que divide repetidamente o espaço de busca em três partes iguais e é usado para encontrar o valor mínimo (ou máximo) de uma função unimodal em um intervalo.

O tempo de execução da busca ternária é logarítmico na medida em que a tolerância diminui. Quanto menor a tolerância, mais etapas de busca serão realizadas.

Pode-se observar que o algoritmo tem um desempenho muito bom para encontrar o mínimo (ou máximo) de funções unimodais, especialmente quando a função é computacionalmente cara para calcular, pois reduz rapidamente o espaço de busca.

O consumo de memória é determinado principalmente pela quantidade de memória necessária para armazenar as variáveis. O consumo de memória é essencialmente constante, independentemente do tamanho do intervalo de busca.

**Figura 06**- Consumo memória Busca Quadrática




## 4. CONCLUSÃO

Os algoritmos de busca tem um papel significativo e fundamental em abordagens computacionais na resolução de uma ampla variedade de problemas do mundo real.

A escolha do algoritmo de busca depende do contexto e dos requisitos do problema. Para vetores pequenos ou não ordenados, a busca sequencial pode ser suficiente. Para vetores ordenados, a busca binária é altamente eficiente. Os algoritmos de busca quadrática e a busca cúbica para vetores grandes tem um desempenho extremamente ruim, deve ser evitado o uso dos mesmos. A busca ternária é eficiente para encontrar mínimos/máximos de funções unimodais, mas não é uma escolha adequada para procurar elementos específicos em vetores.

Portanto, a escolha deve ser feita com base nas características do problema que você está resolvendo.

## 5. REFERÊNCIA BIBLIOGRÁFICA

[1] EDUCAÇÃO, Mundo, **Algoritmos de Busca.** Disponível em: https://panda.ime.usp.br/cc110/static/cc110/13-busca.html. Acesso em: 28 de agosto de 2023.

[2] SANTOS, Valéria, **Algoritmos de Busca Comparação de Algoritmos.** Disponível em: http://www.decom.ufop.br/romildo/2019-2/bcc702/Aula17-Busca-Comparacao-de-Algoritmos.pdf Acesso em: 01 de setembro de 2023.

[1] ACADEMY, Khan, **Busca Binária.** Disponível em: https://pt.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search . Acesso em: 01 de setembro de 2023.