UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO
ESTRUTURA DE DADOS BÁSICAS I
DOCENTE: SELAN RODRIGUES DOS SANTOS

ANÁLISE EMPÍRICA DE ALGORITMOS DE BUSCA

MATHEUS DE ANDRADE SILVA

MARÇO/2019
NATAL-RN

Sumário

1. INTRODUÇÃO.....3
2. MÉTODOS UTILIZADOS.....3
2.1. CARACTERÍSTICAS TÉCNICAS DO COMPUTADOR.....3
2.2. LINGUAGEM E COMPILADOR.....4
2.3. ALGORITMOS TESTADOS.....4
2.4. CENÁRIO DE TESTES.....4
3. RESULTADOS.....5
3.1. ALGORITMOS LINEARES.....5
3.2. RECURSÃO X ITERAÇÃO.....8
3.3. TAMANHO DA PARTIÇÃO E SUA INFLUÊNCIA NO DESEMPENHO.....9
3.4. ALGORÍTMOS DE CLASSES DE COMPLEXIDADE DIFERENTES.....10
3.5. PIOR CASO DA BUSCA FIBONACCI.....12
4. DISCUSSÃO.....13
4.1. ALGORITMOS LINEARES.....13
4.2. RECURSÃO X ITERAÇÃO.....13
4.3. TAMANHO DA PARTIÇÃO E SUA INFLUÊNCIA NO DESEMPENHO.....14
4.4. ALGORITMOS DE CLASSES DE COMPLEXIDADE DIFERENTES.....14
4.5. O PIOR CASO DA BUSCA FIBONACCI.....14
5. CONCLUSÃO.....14

1. INTRODUÇÃO

Este relatório visa descrever os resultados dos testes realizados com um alguns algoritmos de busca. Esses testes foram desenvolvidos e testados com um algorítmo próprio e, com ele, foram gerados alguns aquivos contendo dados sobre tempo de execução e quantidade de passos de cada algorítmo.

Foram testados os algoritmos de busca linear iterativa, busca binária iterativa, busca binária recursiva, busca ternária iterativa, busca ternária recursiva, *jump search* e busca Fibonacci.

2. MÉTODOS UTILIZADOS

2.1. CARACTERÍSTICAS TÉCNICAS DO COMPUTADOR

Para rodar o algorítmo que faz os testes, foi usado um laptop com as seguintes características:

- NOME: Dell Inspiron 5458;
- SISTEMA OPERACIONAL: Ubuntu 18.04.2 LTS 64 bits (com dualboot);
- PROCESSADOR: Intel® Core™ i5-5200U CPU @ 2.20GHz:
  - *CORES*: 2;
  - *THREADS*: 4;
- MEMÓRIA: 8GB;
- PLACA MÃE: Dell 0H23N7.

2.2. LINGUAGEM E COMPILADOR

Foi usado a linguagem C++ 11 com o compilador g++ em sua versão 7.3.0.

2.3. ALGORITMOS TESTADOS

Todas as funçoes possuiam os mesmos argumentos, sendo eles: o arranjo com os elementos, o primeiro índice da parte a ser buscada no arranjo, o último índice da parte a ser buscada no arranjo, o valor a ser buscado e um contador de passos. Os algorítimos testados foram os listados abaixo:

- Busca linear iterativa;
- Busca binária iterativa;
- Busca binária recursiva;
- Busca ternária iterativa;
- Busca ternária recursiva;
- *Jump Search*;

- Busca Fibonacci.

2.4. CENÁRIO DE TESTES

Após a implementação de cada algorítmo de busca, foi implementado um algorítmo para testa-los. O algorítmo de testes possuia um entrada para número de casos de testes e a opção de escolha dos algoritmos a ser testado. Com a entrada dos casos de teste *n*, o programa dividia *n* casos de testes diferentes entre 100 (Cem) e 500000000 (Quinhentos milhões) com intervalos iguais.

Para cada caso de teste foi rodado o algorítmo 100 vezes e feita uma média dos tempos com a finalidade de obter um tempo preciso.

Por fim, foi gerado, para cada algorítmo, dois arquivos de dados: um contendo os dados de tempo para cada caso de teste e outro contendo a quantidade de passos para cada caso de teste.

3. RESULTADOS

3.1. ALGORITMOS LINEARES

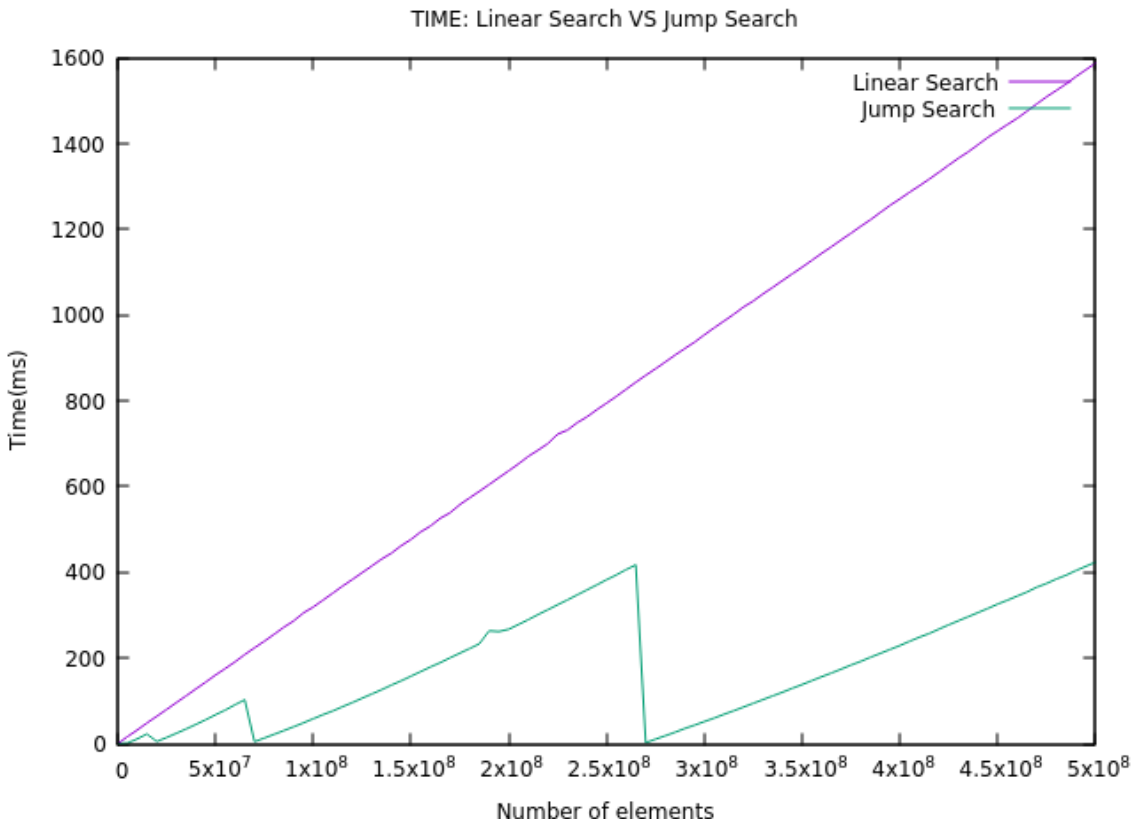

Gráfico 1: Comparação entre os tempos de execução da busca linear e *jump search*.

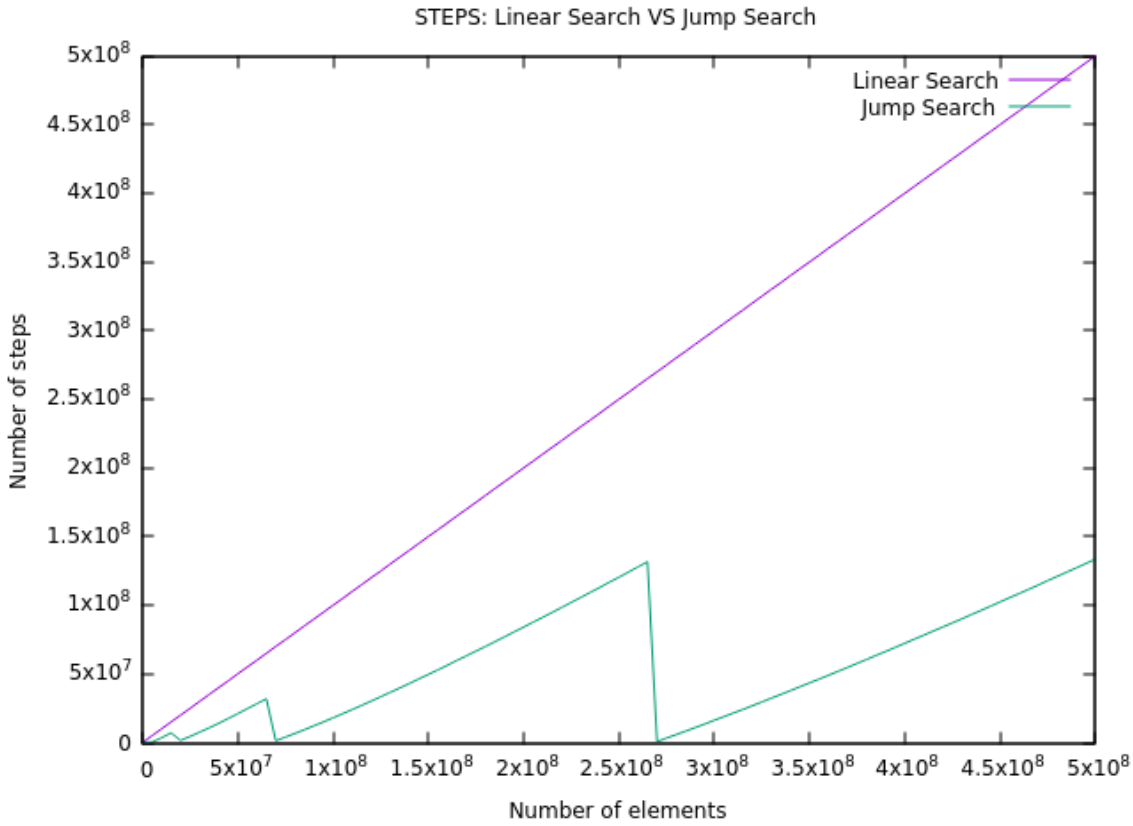

Gráfico 2: Comparação entre os números de passos da busca linear e *jump search*.

3.2. RECURSÃO X ITERAÇÃO

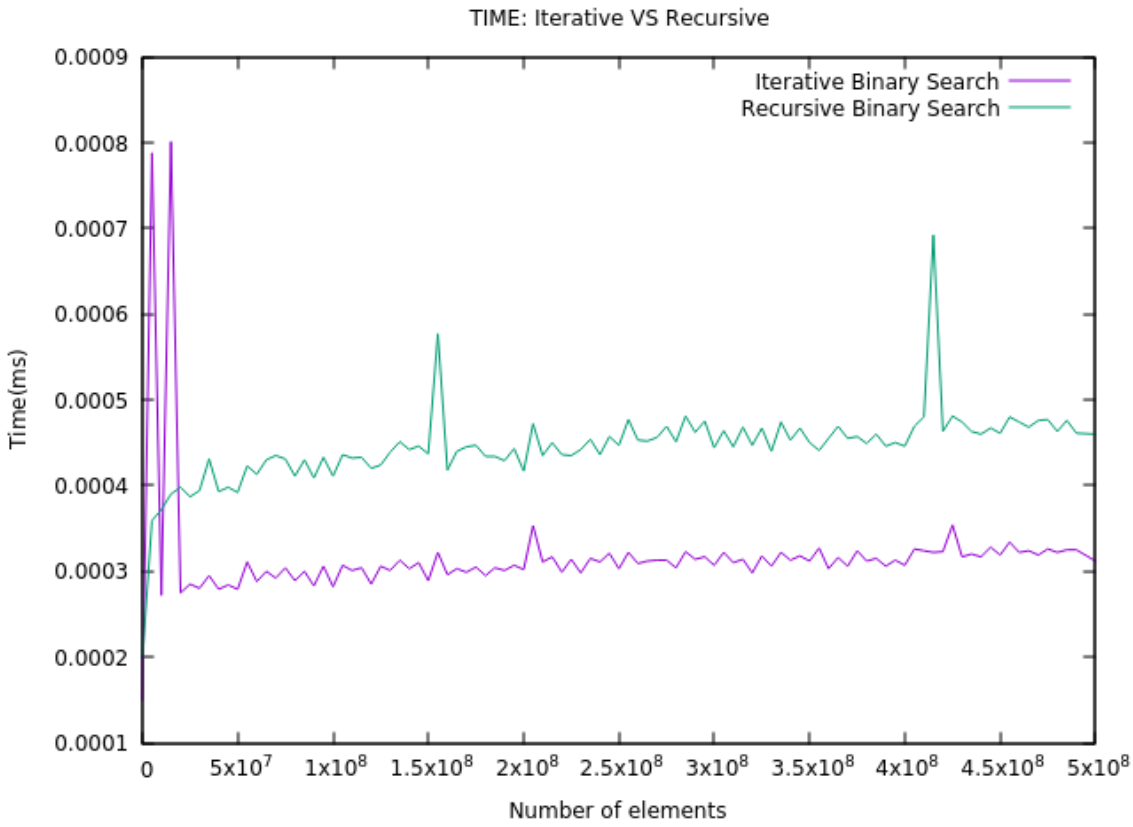

Gráfico 3: Comparação entre os tempos de execução das buscas binárias iterativa e recursiva.

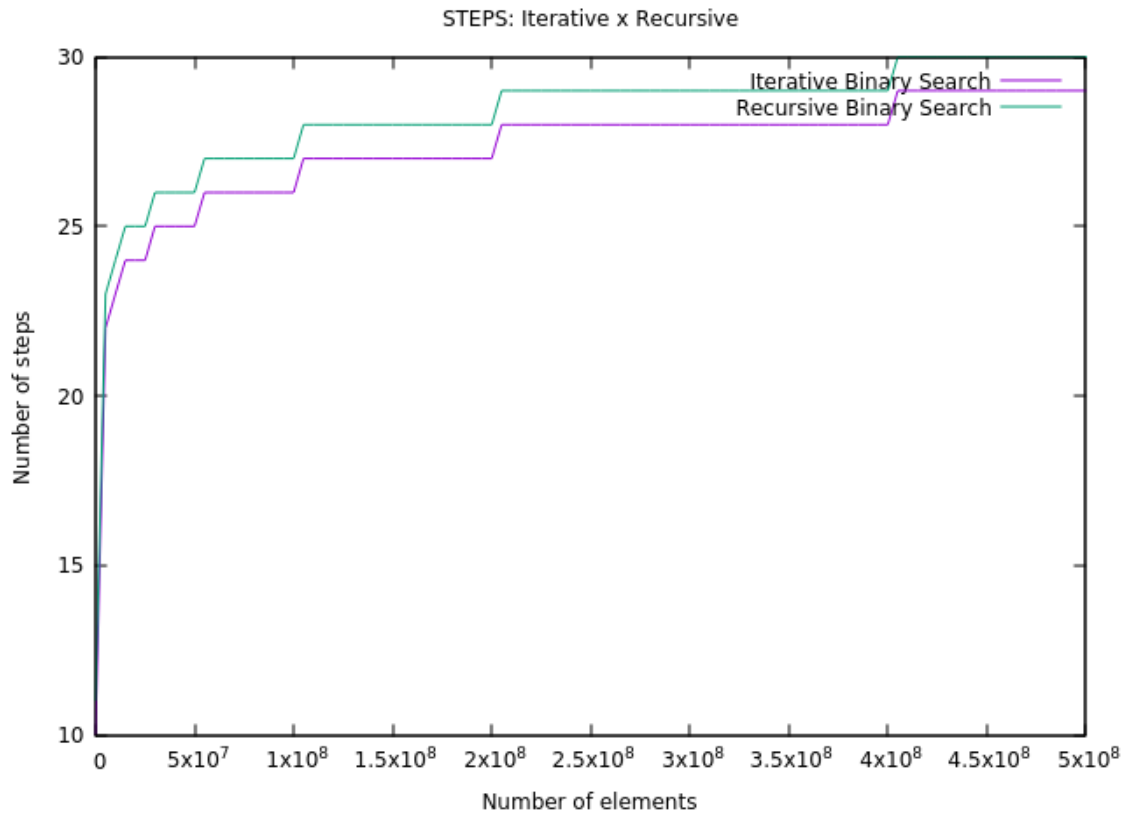

Gráfico 4: Comparação entre os números de passos das buscas binárias iterativa e recursiva.

3.3. TAMANHO DA PARTIÇÃO E SUA INFLUÊNCIA NO DESEMPENHO

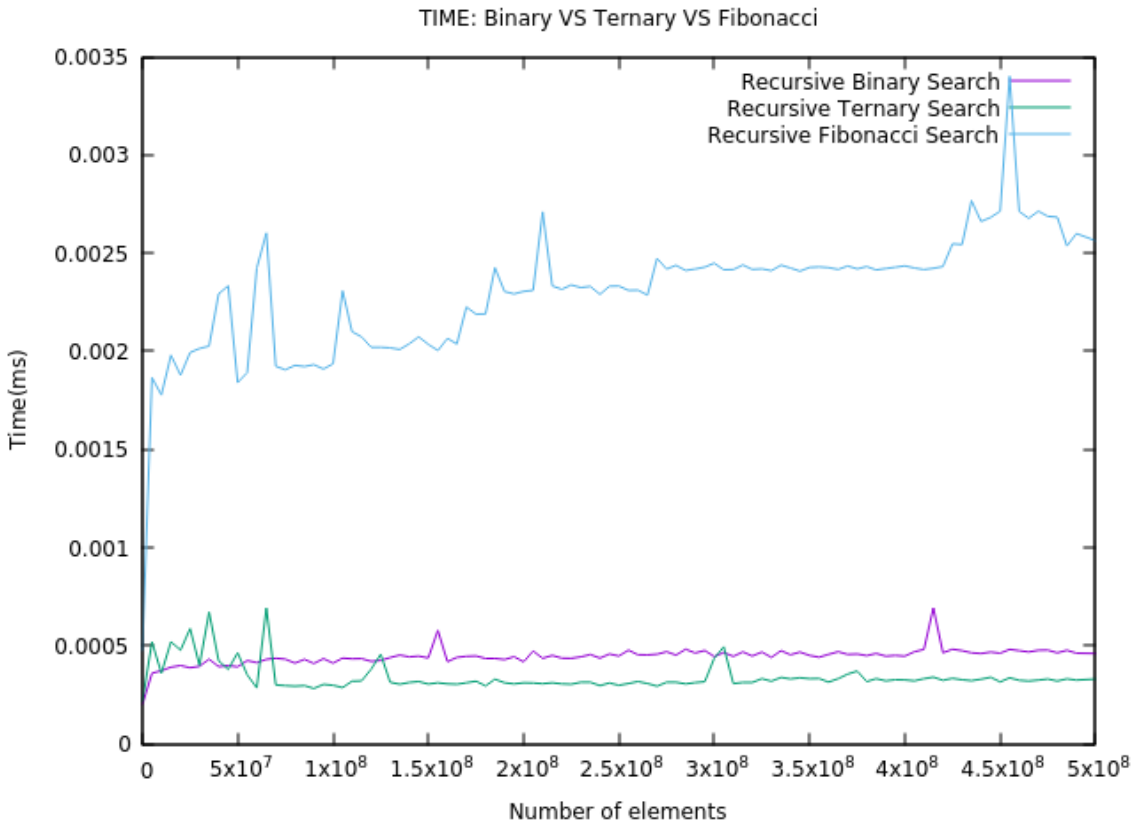

Gráfico 5: Comparação entre os tempos de execução das versões recursiva das buscas binária, ternária e Fibonacci.

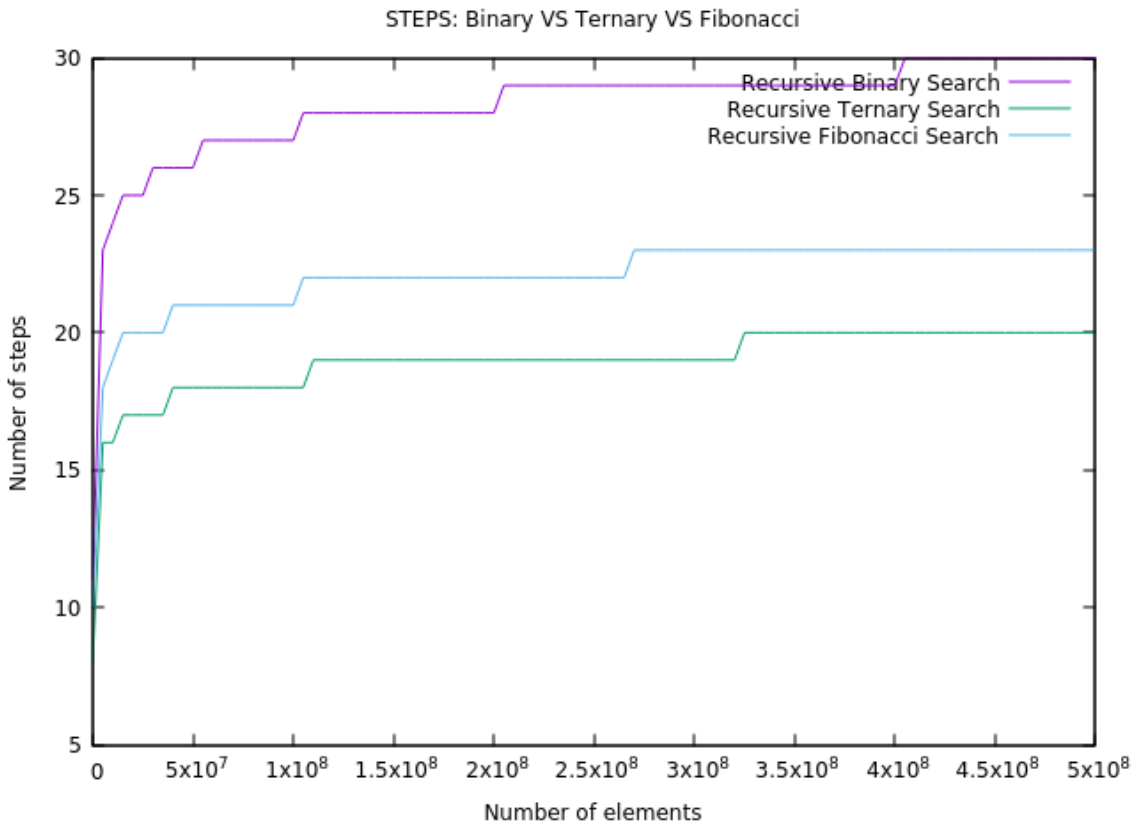

Gráfico 6: Comparação entre os números de passos das buscas binária recursiva, ternária recursiva e Fibonacci.

3.4. ALGORÍTMOS DE CLASSES DE COMPLEXIDADE DIFERENTES

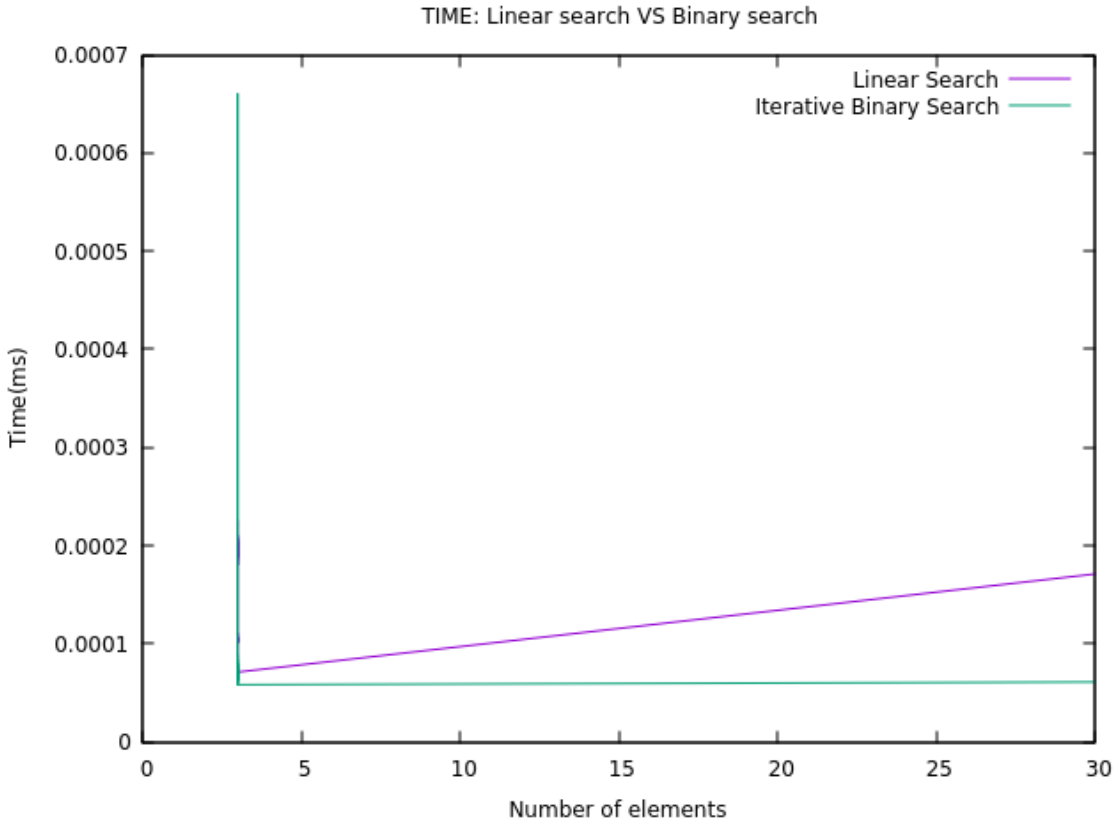

Gráfico 7: Comparação entre os tempos de execução das buscas linear e binária iterativa.

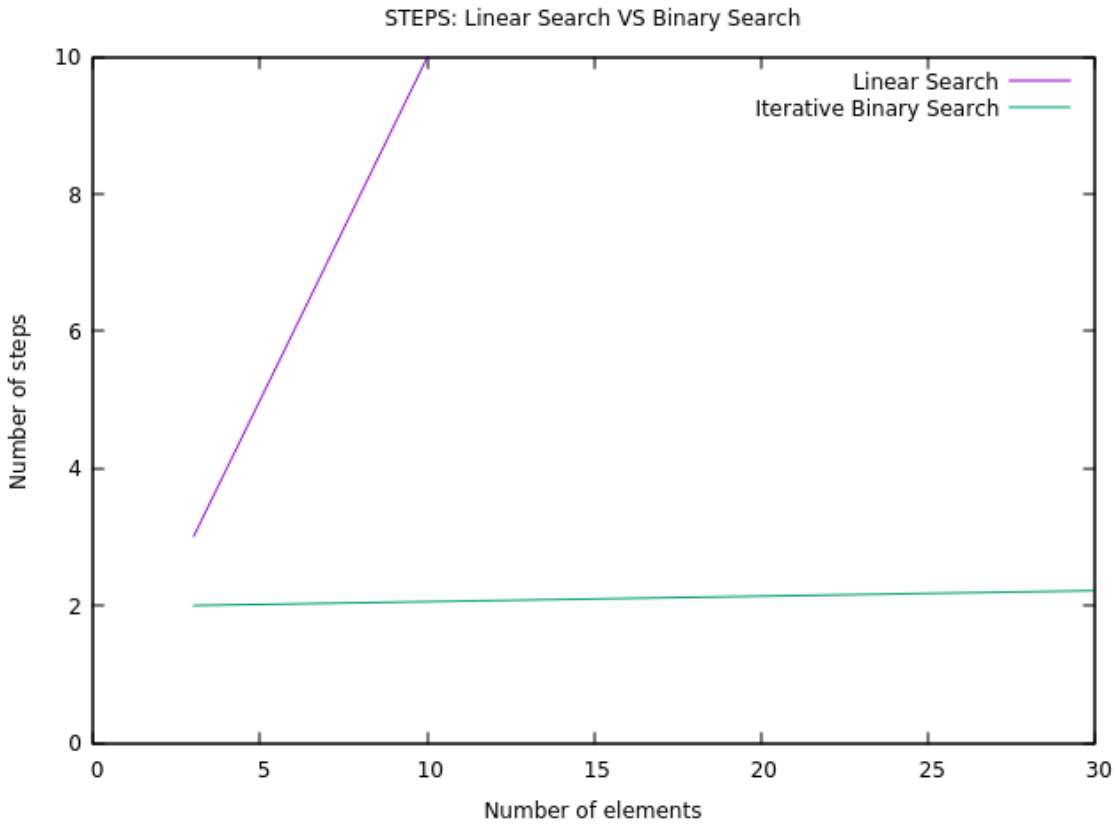

Gráfico 8: Comparação entre os números de passos da busca linear e busca binária iterativa.

3.5. PIOR CASO DA BUSCA FIBONACCI

Nos gráficos a seguir, temos o *Fibonacci Worst Case One* como o pior caso em que o elemento procurado não está no arranjo, mas possui um valor próximo dos elementos do final do arranjo e *Fibonacci Worst Case two* como o pior caso em que o elemento procurado não está no arranjo, mas possui um valor próximo dos elementos do começo do arranjo.

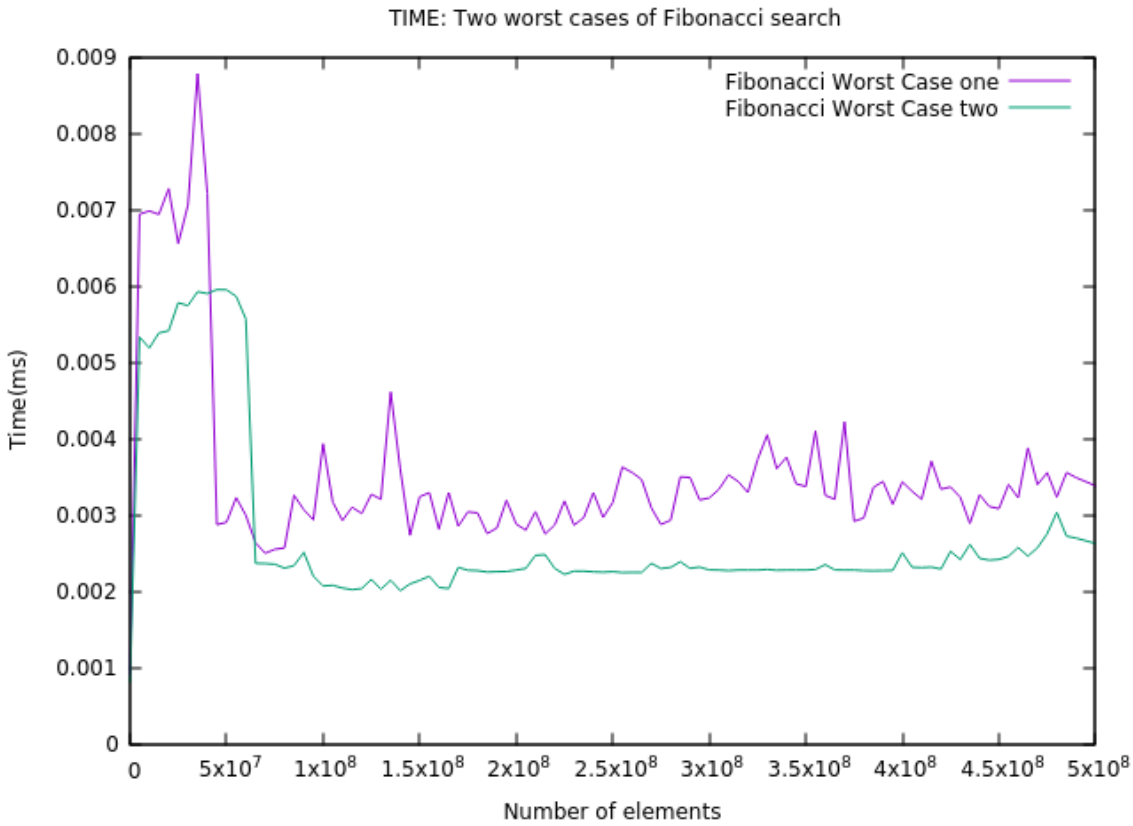

Gráfico 9: Comparação entre os tempos de execução dos dois piores casos da busca Fibonacci.

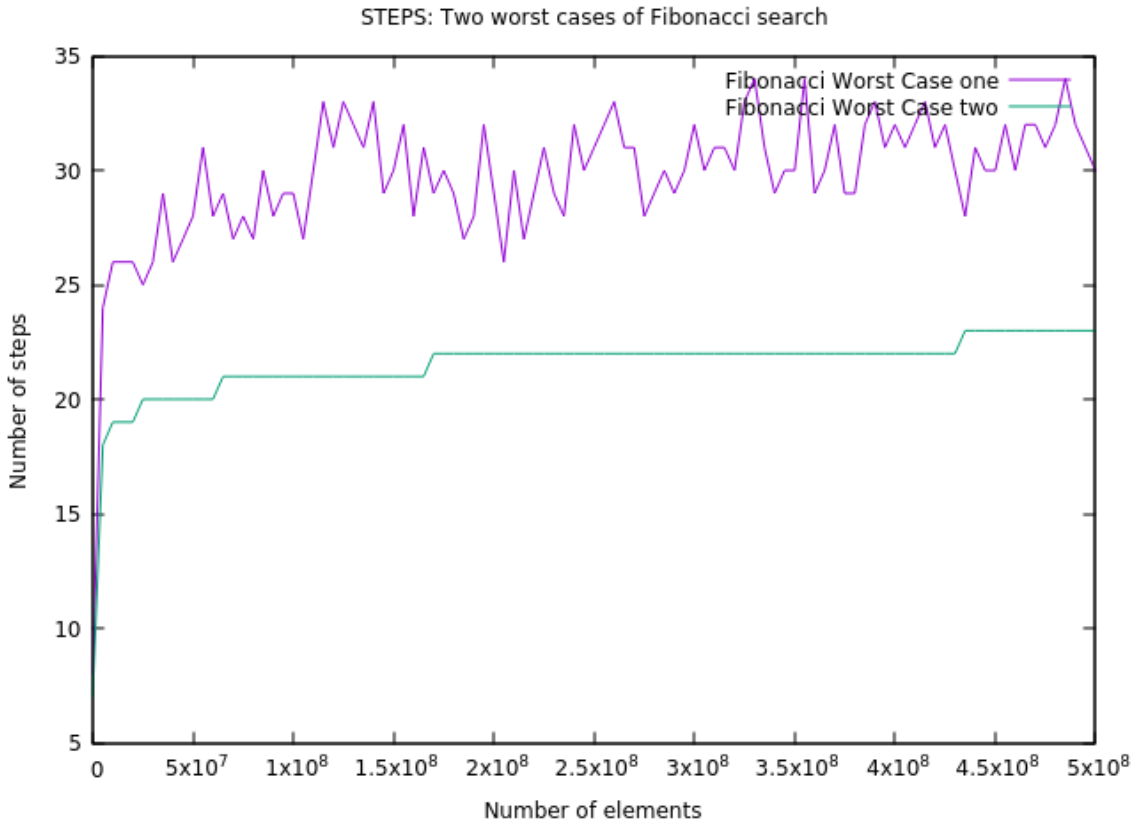

Gráfico 10: Comparação entre os números de passos dos dois piores casos da busca de Fibonacci.

4. DISCUSSÃO

4.1. ALGORITMOS LINEARES

Sabendo que o pior caso da busca linear é quando o elemento a ser buscado não está no vetor e o pior caso do *jump search* é quando o elemento a ser buscado não está no vetor e é maior que o último elemento do vetor, fazendo essa análise, concluimos que o algorítmo *jump search* se sai muito melhor do que a busca linear no pior caso (gráfico 1 e 2).

Esse resultado já é esperado, já que, pela verificação “por pulos” do *jump search*, esse consegue fazer uma busca linear em um intervalo bem menor do que a busca linear.

4.2. RECURSÃO X ITERAÇÃO

Comparando a busca binária iterativa com a busca binária recursiva, notamos um melhor desempenho na busca binária iterativa, porém com diferença pequena. Isso já era esperado, já que a versão recursiva precisa chamar ela mesma várias vezes, o que demanda um tempo a mais.

4.3. TAMANHO DA PARTIÇÃO E SUA INFLUÊNCIA NO DESEMPENHO

Em questão de tempo, a binária e ternária são muito parecidas, já a Fibonacci fica distante delas (Gráfico 5) com um tempo bem maior (mas ainda baixo, se comparado com a linear).

Nos testes realizados, as buscas binária e ternária ficam quase com o mesmo tempo de execução, sendo algumas vezes a binária mais rápida e outras a ternária.

4.4. ALGORITMOS DE CLASSES DE COMPLEXIDADE DIFERENTES

Comparando a busca binária iterativa e a busca linear, mesmo com um arranjo muito pequeno, a binária se sai melhor em todos os testes. Quanto mais o arranjo vai crescendo em quantidade de elementos, maior é a diferença de tempo e passos entre esses dois algoritmos.

Com um arranjo muito grande, nota-se na hora dos testes uma grande demora na hora de rodar o algoritmo da busca linear, o que não acontece na busca binária iterativa.

4.5. O PIOR CASO DA BUSCA FIBONACCI

Na busca Fibonacci temos dois piores casos, como foi citado em 3.5. Analisando os gráficos, notamos que o pior caso em que o elemento procurado não está no arranjo, mas possui um valor próximo dos elementos do final do arranjo leva mais tempo e executa mais passos do que o outro pior caso.

Com isso, definimos o pior caso definitivo da busca Fibonacci.

5. CONCLUSÃO

Tendo em mente os resultados obtidos nos testes, notamos que o uso de algoritmos como busca linear e *jump seach* são desnecessários, já que agora temos conhecimento de algorítmos que fazem o mesmo trabalho em um tempo muito menor, são eles: busca binária, busca ternária e busca fibonacci. Sendo a busca binária a melhor, pela fácil implementação.