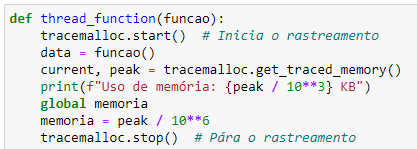
Inácio Lima de Souza Filho – 509153

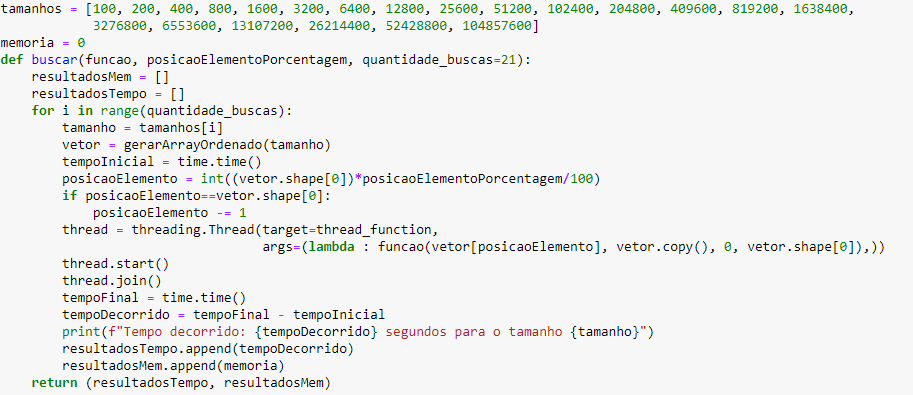
Para o trabalho foram usados 5 algoritmos de busca e para 7 casos diferentes:

1. Algoritmo 1: busca linear v1 O(n)
2. Algoritmo 1: busca linear v2 O(n)
3. Algoritmo 2: busca binária O(log n) (número procurado primeiro ou último)
4. Algoritmo 2: busca binária O(log n) (número procurado o elemento do meio)
5. Algoritmo 3: busca quadrática com contagem de repetição de elementos O(n^2)
6. Algoritmo 4: busca ternária O(log n)
7. Algoritmo 5: busca cúbica - tripla checagem O(n^3)

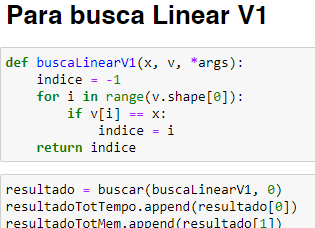
O código fornecido foi feito em Java, mas houve a conversão para python, foi usado a estrutura de array do Numpy no lugar dos vetores do Java. Para medir a memória foi utilizada a biblioteca do tracemalloc e como está sendo trabalhado com threads foi salvo em uma variável global, memoria.



Para fazer a busca foi usado um método que faz a busca para qualquer algoritmo recebido por parâmetro. Nesse algoritmo são usados vários tamanhos de vetores (os tamanhos foram mudados para melhor visualização) e feito a busca para cada tamanho, também é gerado o vetor e passado a função para executar em uma thread separada. No final se calcula o tempo total gasto e imprime na tela.



Para cada algoritmo foi desenvolvida uma função e essa função foi passada como parâmetro para ser feito a busca.



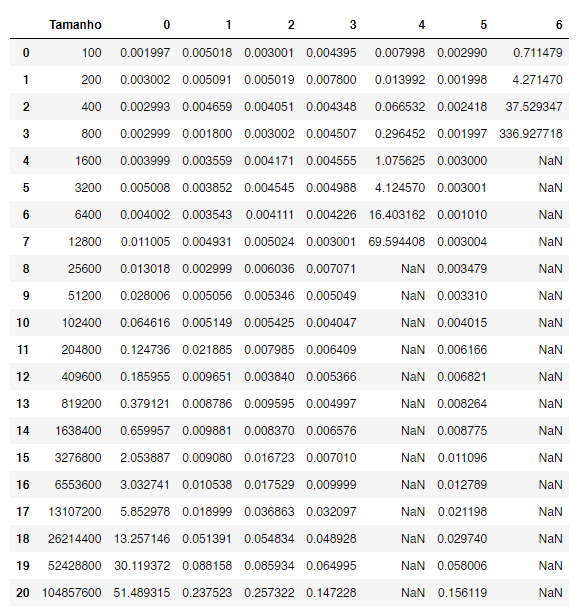
Nesse caso, o segundo parâmetro da função é a posição a ser buscada no vetor, se 0 a primeira posição, se 100 a última. Para a busca quadrática e cúbica também foi necessário limitar o tamanho do vetor, porque o tempo de processamento ficou muito grande, a limitação é o terceiro parâmetro da função.



O último passo foi passar esses resultados para tabelas:

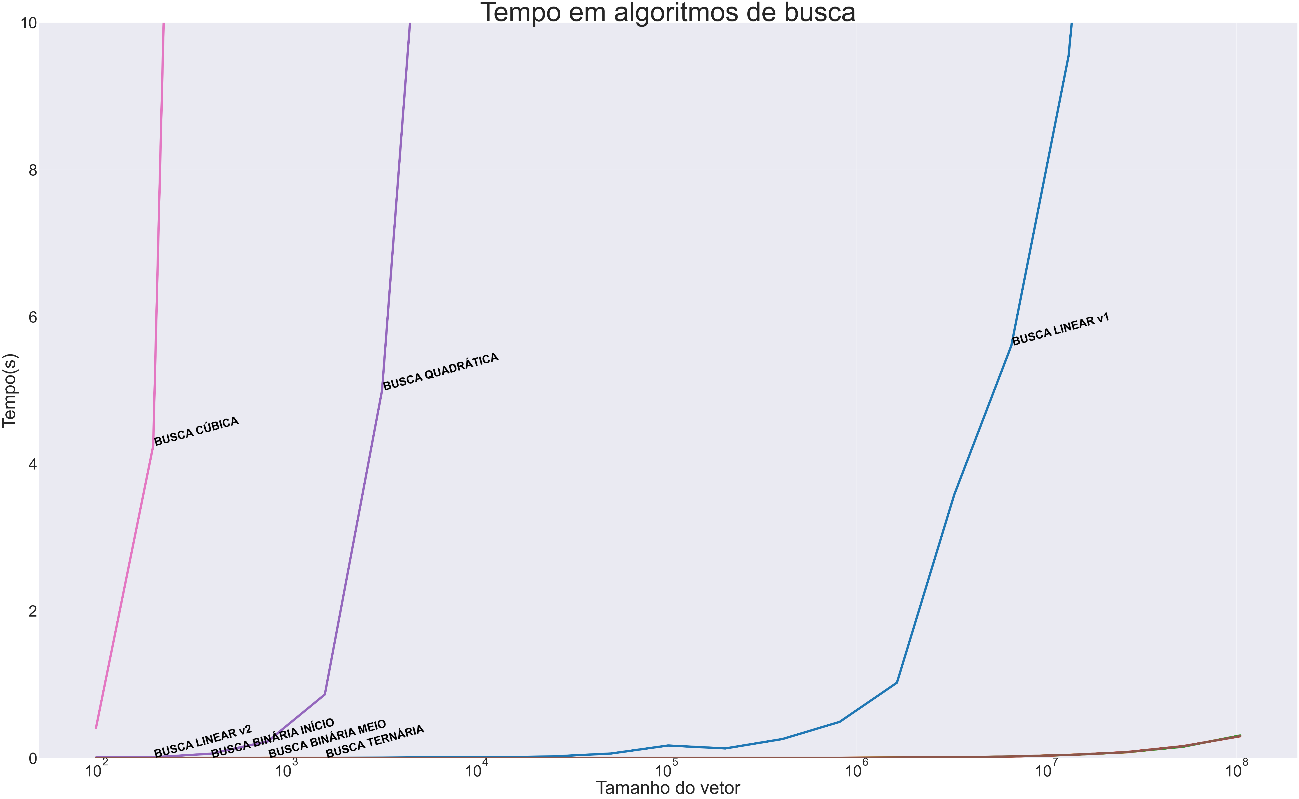


Segue o resultado:



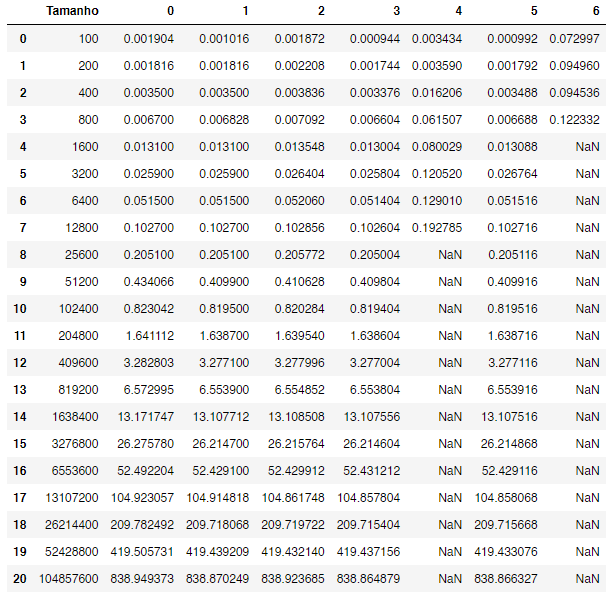
Pela a análise, os métodos mais rápidos foram a busca binária no meio do vetor e a busca ternária. Não foi possível executar os algoritmos quadráticos e cúbicos porque eles levariam muito tempo, então foi executado apenas para tamanhos menores, mas mesmo assim é possível perceber como o tempo nesses algoritmos foi muito maior.

Para plotar os gráficos foi usado o lineplot do Seaborn, como valores do cúbico e quadrático destoam muito dos outros valores foi feita uma mudança da escala. Segue o resultado:



Nesse caso é possível notar com clareza como os algoritmos de O(log n) são muito menores que os outros. Para a busca linear o gráfico ficou parecendo um exponencial, mas isso ocorre porque quando foi plotado o gráfico se usou escala logarítmica para os tamanhos.

Já para a memória também foi montado uma tabela:



Nesse caso é notável como a memória não teve muita variação entre um método e outro, apenas um pequeno uso maior para o método de busca binária e ternária. Segue o gráfico:

