Um breve ensaio sobre a velocidade e o número de comparações dos algoritmos de ordenação e busca recursivos

Wilian Pereira dos Santos[1]

[1]Departamento de Informática - Universidade Federal do Paraná (UFPR)

Curitiba - PR - Brasil

Introdução

O trabalho busca comparar alguns algoritmos de ordenação em vetores em suas versões recursivas, como o Insertion Sort, Selection Sort, Merge Sort e o Quick Sort, além dos algoritmos de busca sequencial e binária, também em versões recursivas. Levando em consideração o número de comparações entre os elementos do vetor e a velocidade de execução da ordenação.

Análise

Em vetores de tamanho pequeno, os quatro algoritmos de ordenação possuem tempos de execução muito próximos. Com tamanho 10, o algoritmo Insertion Sort possui a melhor média, com 0,000001s a cada execução. Enquanto o restante leva 0,000002s para completar a operação.

O resultado se repete quando o parâmetro é o número de comparações realizadas pelos algoritmos. O Insertion sort possui a melhor média, com 22 comparações a cada execução, enquanto os outros fazem a partir de 31 comparações.

Vetores com tamanhos um pouco maiores, tamanho 100, por exemplo, começam a explicitar que os algoritmos possuem uma disparidade, ou seja, não crescem de maneira linear. As velocidades, começam a divergir, entretanto, a diferença é mais visível no número de comparações. Vide as figuras 1 e 2.

Figura 1: Velocidade dos algoritmos de ordenação em vetor de tamanho 100.

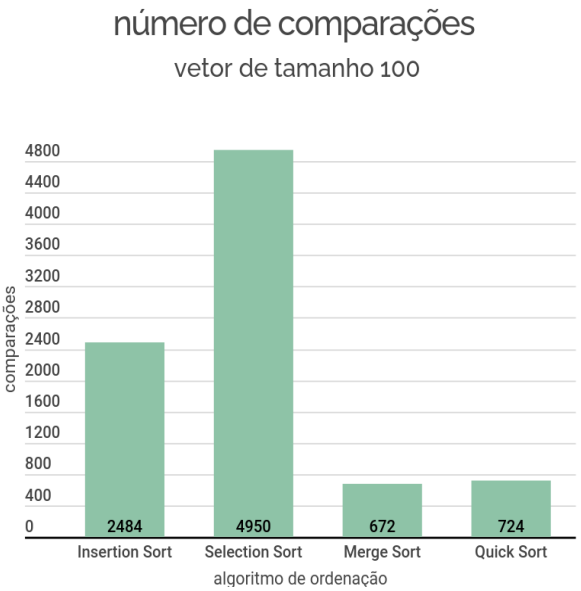


Figura 2: Comparações realizadas pelos algoritmos de ordenação em vetor de tamanho 100.

Em vetores de de tamanhos maiores, como 10.000 e 50.000, a diferença, em ambos os quesitos, cresce grotescamente. É possível ver que algoritmos mais simples, como o Insertion e Selection Sort não são boas opções para se utilizar em programas ou sistemas que trabalham com grande volume de dados. Enquanto o Merge e o Quick Sort se consagram. Veja as figuras 3 e 4.

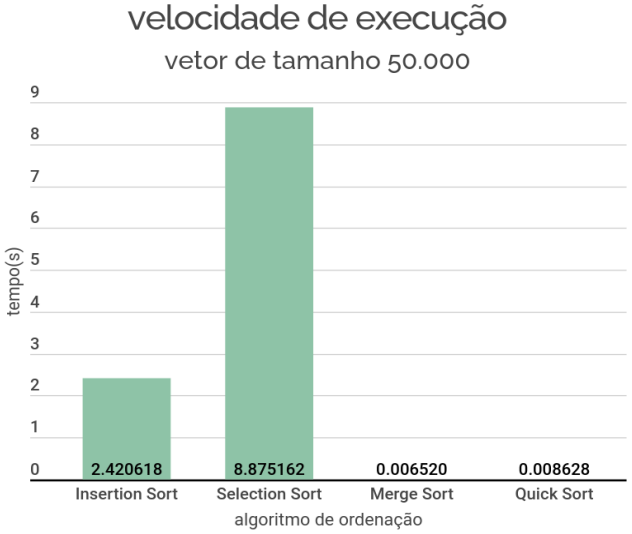


Figura 3: Velocidade de execução dos algoritmos de ordenação em vetor de tamanho 50.000.

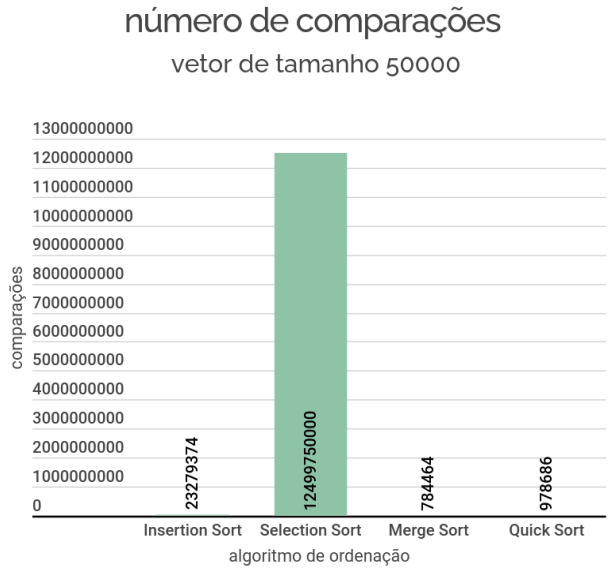


Figura 4: Comparações realizadas pelos algoritmos de ordenação em vetor de tamanho 50.000.

Os algoritmos de busca, em vetores de tamanhos menores, como tamanho 10, também realizam a operação no mesmo tempo. A diferença fica no número de comparações: a busca sequencial, caso não encontre o elemento no vetor, realiza sempre o tamanho do vetor como número de comparação. Enquanto a busca binária, também não encontrando o elemento procurado no vetor, realiza $log\ n\ (base\ 2)\ +\ 1$ comparações.

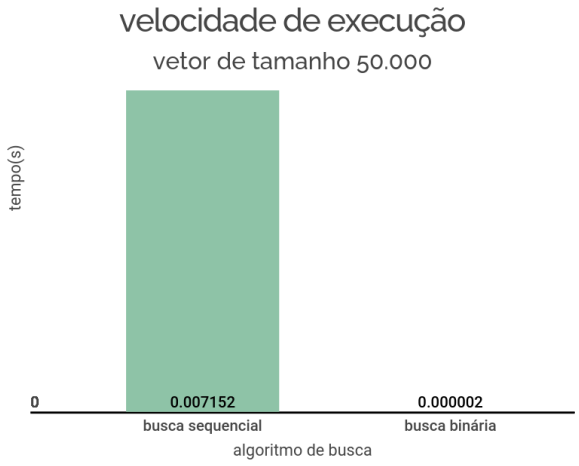
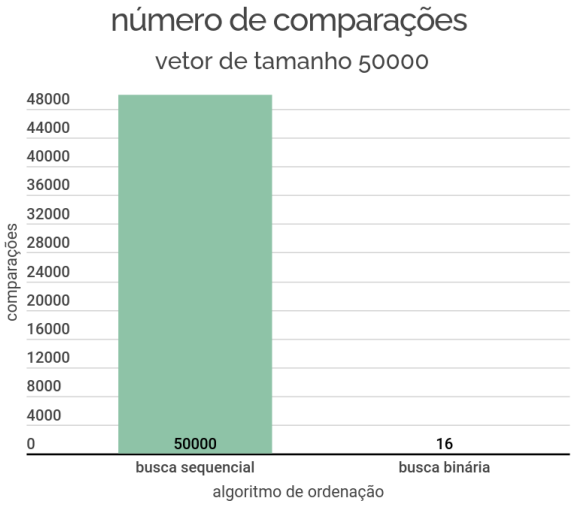


Figura 5: Velocidade de execução dos algoritmos de busca em vetor de tamanho 50.000.

Figura 6: Comparações realizadas pelos algoritmos de busca em vetor de tamanho 50.000.



A busca binária se mostra muito mais eficiente, no entanto o vetor precisa estar ordenado, e isso custa. Porém, mesmo utilizando um rápido algoritmo de ordenação, como o merge ou quick sort, ela ainda compensa muito.