Algoritmos de Busca e Ordenação

Otto Schmidt
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
os24@inf.ufpr.br

I. Introdução

Esse é um relatório sobre algoritmos de busca e ordenação. Os algoritmos de busca utilizados foram a busca sequencial e a busca binária. E os algoritmos de ordenação são Insertion Sort, Selection Sort e Merge Sort. Foram criadas versões recursivas e iterativas dos algoritmos, com exceção do Merge Sort, o qual terá somente a versão recursiva.

Foram realizados diversos testes, em ambiente Linux, para coletar o custo de comparação e medir o tempo de execução de cada algoritmo, a fim de compará-los e determinar suas características.

II. Experimentos Realizados - Busca

Os algoritmos de busca também foram testados 20 vezes, era necessário procurar um elemento na extremidade esquerda de um vetor. O algoritmo de busca sequêncial começa a busca pelo lado direto, contudo, sua versão recursiva não foi executada, já que foram utilizados vetores com mais de 100.000.000 elementos.

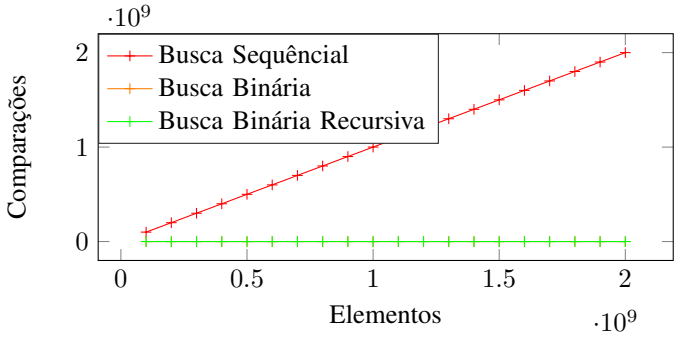

Figura 1. Gráfico com a quantidade de comparação dos algoritmos de busca

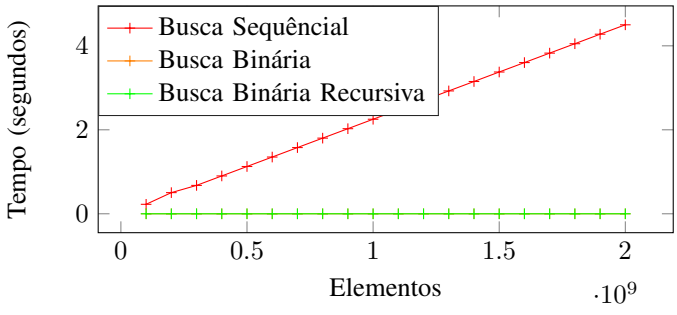

Figura 2. Gráfico com o tempo de execução dos algoritmos de busca até 2 bilhões de elementos

III. Experimentos Realizados - Ordenação

Os algoritmos de ordenação foram testados 20 vezes. Eles precisavam ordenar, de forma não decrescente, um vetor que estava ordenado de forma não crescente. Porém, os algoritmos Insertion e Selecion Sort, em suas formas recursivas, também não foram executados em vetores grandes.

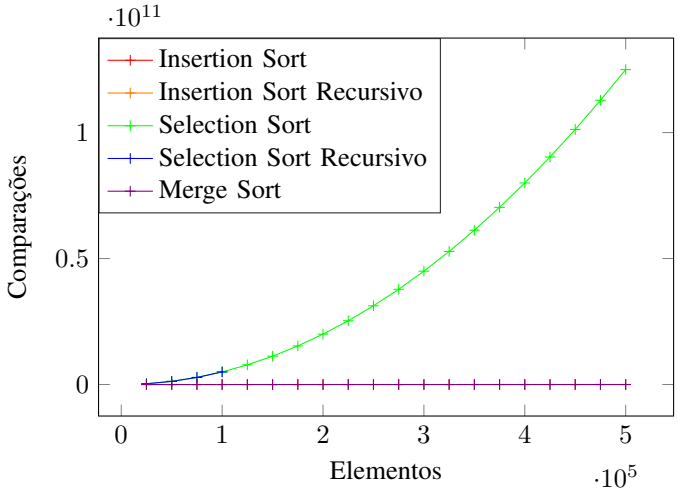

Figura 3. Gráfico com a quantidade de comparações realizadas para os algoritmos de ordenação

Na figura 3, podemos ver que a quantidade de comparações do Selection Sort cresce muito rápido, dificultando a visualização dos outros algoritmos. Portanto, na figura 4 o Selecion Sort foi removido.

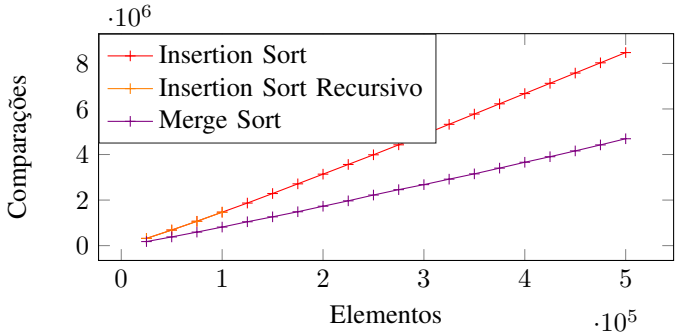

Figura 4. Gráfico com a quantidade de comparações realizadas para os algoritmos de ordenação

E, como dito anteriormente, além de medir a quantidade de comparações realizadas pelos algoritmos, também foi crono-

metrado os tempos de execução, eles estão presentes na figura 5.

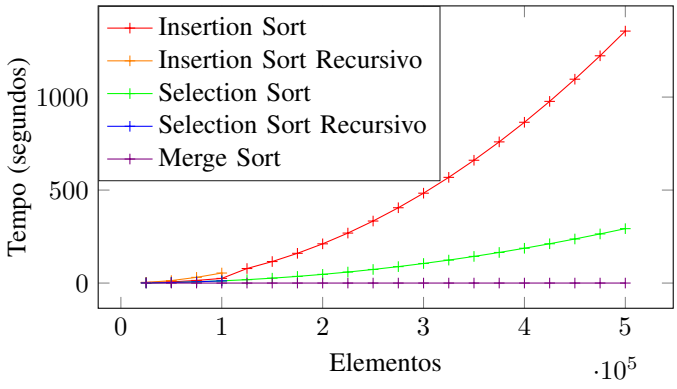

Figura 5. Gráfico com o tempo de execução dos algoritmos de ordenação até 500.000 elementos

IV. Análise Resultados

Primeiramente, somente o algoritmo Insertion Sort mostrou diferença entre os métodos recursivos e iterativos. Segundamente, os algoritmos Insertion e Selection Sort e Busca Sequêncial, em versões recursivas, não foram possíveis executá-los utilizando um vetor com centenas de milhares de elementos, já que esses algoritmos realizam muitas chamadas recursivas e acabavam consumindo todo o stack.

E é possível notar que no início do gráfico na figura 5, é possível notar que eles estão bem próximos no tempo de execução. Portanto, na figura 6 foram feitos outros 20 testes com incremento de 5000 elementos a cada iteração do experimento.

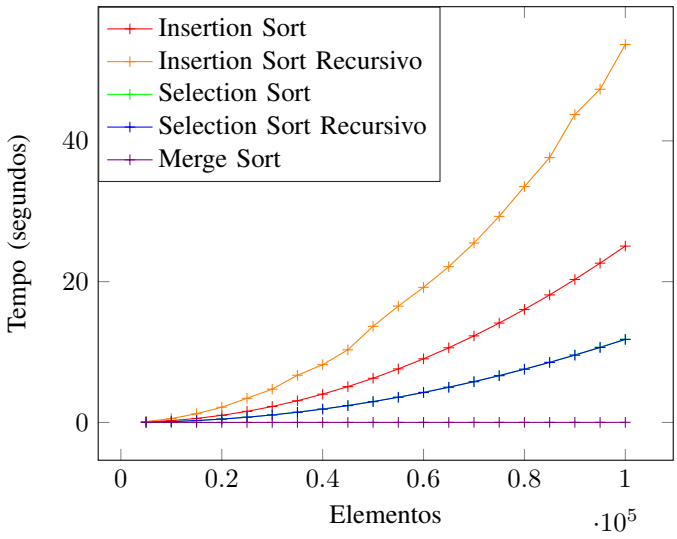

Figura 6. Gráfico com o tempo de execução dos algoritmos de ordenação até 100.000 elementos

No gráfico da figura 6, é possível notar que antes de 20.000 elementos, eles possuem tempos de execução muito similares. Ou seja, em vetores pequenos, deve-se escolher o algoritmo de ordenação dependendo da perculiaridade do computador.

Agora, em relação aos algoritmos de busca em geral, é possível perceber que a busca binária é melhor que a sequencial, já que realiza menos comparações e seu tempo de execução é praticamente instantâneo. E, sobre os algoritmos de ordenação, o Merge Sort aparenta ser melhor que todos. O Selection Sort realiza muitas comparações e é mais lento que o Merge Sort. Já o Insertion Sort realiza mais comparações que o Merge e é extremamente lento.

Contudo, para o algoritmo Merge funcionar, é necessário que ele crie um vetor cópia, portanto é necessário que o computador tenha o dobro da memória do vetor a ser ordenado. E, mesmo que o Selection Sort faça muito mais comparações que todos os algoritmos, ele ainda é mais rápido que o Insertion Sort, já que o Insertion Sort realiza muitas trocas e fica muito mais dependente na velocidade de escrita da memória, o que geralmente é mais lenta que a velocidade de leitura.

V. Conclusão

Deste modo, em relação aos algoritmos de busca, a busca binária é a melhor opção. E, sobre os algoritmos de ordenação, se você tiver memória suficiente e o vetor for muito grande, o Merge Sort é o melhor. Todavia, para vetores pequenos e equipado de pouca memória, porém rápida, o Insertion Sort, em sua versão iterativa, vai ser melhor. E caso a memória não seja rápida, o Selection Sort é a melhor opção.

E, a partir dessas peculiaridades, surgiram alguns algoritmos de ordenação híbridos, os quais tipicamente combinam algoritmos do tipo dividir e conquistar com incrementais, e.g. Merge Sort com Insertion Sort. Então ele começaria ordenar o vetor usando o Merge Sort e, quando ele chegasse num tamanho especificado, mudaria para o Insertion Sort. [1]

Referências

[1] "Algorithmic analysis and sorting - part three," Disponível em https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1138/lectures/13/Slides13.pdf (2013-07-21), Stanford University.