Componente Curricular: Laboratório de Estrutura de Dados

Professor: Fábio

Aluno(a): Alessia Bianca Araújo

Análise Comparativa de Algoritmos de Busca

* Para o desenvolvimento desta atividade foi utilizado os seguintes componentes:

Notebook Samsung

Java 17(JBR 17.0.9)

Intellij IDEA Ultimate 2023.3

* Introdução:

Neste relatório, serão apresentados os resultados de uma análise comparativa dos algoritmos de ordenação Insertion Sort, Selection Sort e Bubble Sort. Utilizei o ambiente de desenvolvimento Intellij IDEA Ultimate para implementar e testar os algoritmos em Java, e foram avaliadas métricas como o número de trocas de posição no array, o número de comparações realizadas e o tempo de execução de cada algoritmo.

O objetivo foi comparar o desempenho de cada algoritmo em diferentes cenários, levando em consideração suas características e complexidades. Cada algoritmo foi implementado em Java de acordo com sua descrição clássica. Foi garantido que os algoritmos realizassem as operações necessárias para contabilizar o número de trocas de posição, o número de comparações e o tempo de execução.

Análises dos algoritmos de busca:

* Insertion Sort:

O Insertion Sort é um algoritmo simples e intuitivo que ordena elementos em um array por meio de inserções sequenciais, funciona bem para pequenos conjuntos de dados ou quando o array já está quase ordenado. O algoritmo é estável, o que significa que preserva a ordem relativa de elementos com chaves iguais.Ele é in-place, o que significa que a ordenação é realizada no próprio array de entrada, sem necessidade de alocação de memória adicional. Em relação ao seu desempenho o número de trocas de posição pode variar de zero a n^2 dependendo da entrada, com um melhor desempenho em arrays quase ordenados.

O número de comparações realizadas é moderado a alto, afetando o desempenho, especialmente em conjuntos de dados desordenados, o tempo de execução é geralmente eficiente para conjuntos de dados pequenos, mas pode ser significativamente afetado pelo tamanho do array em casos de conjuntos de dados grandes e desordenados. É frequentemente usado em situações onde a simplicidade e a facilidade de implementação são mais importantes do que o desempenho absoluto, como em algoritmos de ordenação interna de bibliotecas de linguagens de programação.

* Selection Sort:

O Selection Sort é um algoritmo direto que ordena elementos selecionando repetidamente o menor elemento restante e colocando-o na posição correta. Ele é in-place, mas não é estável, o que significa que a ordem relativa de elementos iguais pode ser alterada durante a ordenação. O algoritmo é simples de entender e implementar, tornando-se uma escolha comum para fins educacionais.

Em relação ao desempenho número de trocas de posição é sempre n−1, tornando-o determinístico e consistente independentemente da entrada. O número de comparações realizadas é n×(n−1)/2, o que o torna menos eficiente do que o Insertion Sort em conjuntos de dados quase ordenados. O tempo de execução é estável, mas não muito eficiente, especialmente para grandes conjuntos de dados, devido à sua complexidade de tempo quadrática.

É adequado para situações onde a simplicidade é priorizada e o tamanho do conjunto de dados é relativamente pequeno, como classificações em ambientes de aprendizado e prototipagem de software.

* Bubble Sort:

Características:

O Bubble Sort é um algoritmo simples que compara e troca elementos adjacentes repetidamente até que o array esteja ordenado. Ele é in-place e estável, mas é considerado o menos eficiente dos três algoritmos devido à sua complexidade de tempo quadrática em todos os casos.

Em relação ao desempenho o número de trocas de posição pode variar de zero a , dependendo da entrada, com uma tendência a realizar mais trocas em conjuntos de dados desordenados.

O número de comparações realizadas é semelhante ao número de trocas, variando de zero a n×(n−1), o tempo de execução é geralmente o mais lento dos três algoritmos, tornando-o inadequado para grandes conjuntos de dados.

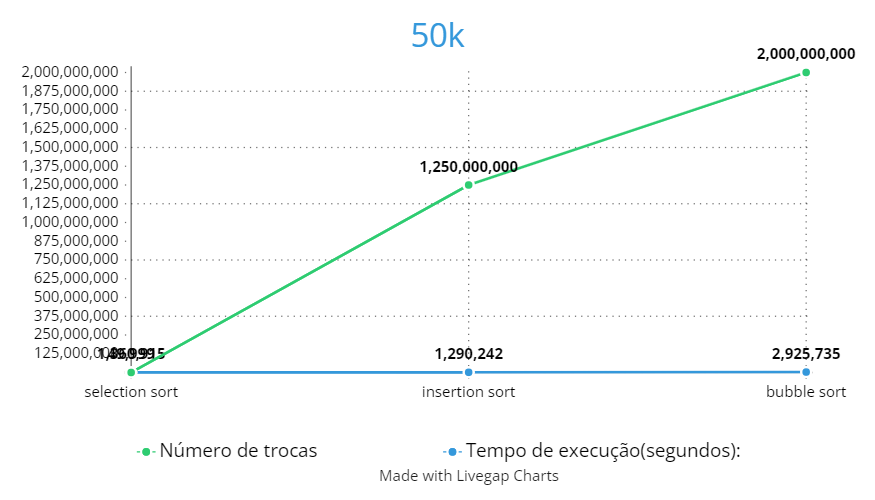
É raramente usado em aplicações do mundo real devido à sua baixa eficiência, mas pode ser útil em situações onde a simplicidade e a compreensibilidade do código são mais importantes do que o desempenho absoluto.

Conclusão:

Cada algoritmo de ordenação tem suas próprias características e é adequado para diferentes cenários e necessidades. O Insertion Sort é preferível para pequenos conjuntos de dados ou quando o array está quase ordenado, enquanto o Selection Sort é simples e consistente, mas menos eficiente. O Bubble Sort, embora seja o mais simples de entender, é o menos eficiente e raramente é usado na prática, exceto para fins educacionais ou em conjuntos de dados muito pequenos. A escolha do algoritmo de ordenação adequado depende das especificidades do problema, das restrições de desempenho e das características do conjunto de dados.

* Alguns dos testes:

Com 50k



Com 100k

