Análise de Desempenho de Algoritmos de Ordenação

Rafael Passos Domingues

I. Introdução

Nesta análise, serão investigados os algoritmos de ordenação Bubble Sort, Selection Sort e Insertion Sort. O desempenho desses algoritmos será avaliado com base no número de utilizações em diferentes configurações de arrays.

II. Referencial Teórico

Incluir aqui o referencial teórico relevante para a análise de desempenho de algoritmos de ordenação. Cite artigos, livros ou outras fontes científicas que abordem o assunto.

III. Material Utilizado

Para realizar esta análise, foram utilizados dois computadores: um computador pessoal com sistema operacional Linux e um computador disponível no laboratório com sistema operacional Linux. O código foi implementado em C++ e compilado com o compilador g++.

IV. Métodos Implementados

O código em questão implementa os seguintes métodos:

- Inclusão das bibliotecas necessárias: `iostream`, `fstream`, `ctime` e `algorithm`.
- Definição das funções de classificação: `bubbleSort`, `insertionSort` e `selectionSort`.
- Definição da função `randomArrayGenerator` para gerar um array aleatório.
- Definição da função `saveResultsToFile` para salvar os resultados em um arquivo CSV.
- Implementação da função principal (`main`) que realiza as seguintes ações:
  1) Definição de constantes para determinar o tamanho inicial, tamanho final e incremento.
  2) Declaração dos arrays e variáveis necessários para armazenar os resultados.
  3) Preenchimento do array de tamanhos com valores incrementais.
  4) Execução dos algoritmos de classificação para diferentes tamanhos de arrays (aleatório, crescente e decrescente).
  5) Armazenamento do número de utilizações realizadas em cada algoritmo de classificação.
  6) Chamada da função `saveResultsToFile` para salvar os resultados em um arquivo CSV chamado "sort_usages.csv".
  7) Retorno de 0 para indicar que o programa foi executado com sucesso.

V. Resultados Obtidos

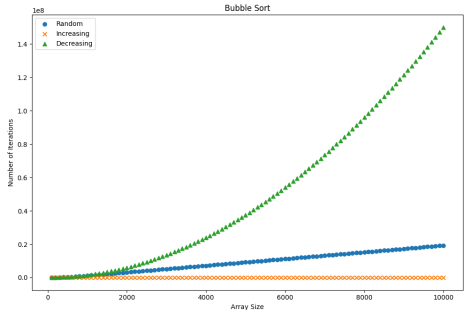

Fig. 1. Número de utilizações do Bubble Sort para diferentes tamanhos de arrays.

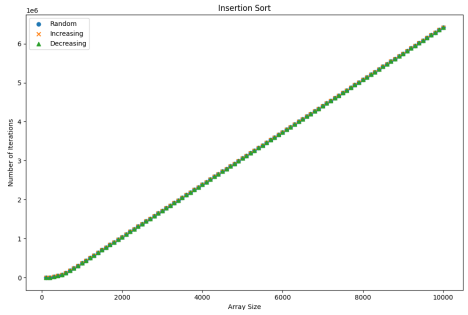

Fig. 2. Número de utilizações do Insertion Sort para diferentes tamanhos de arrays.

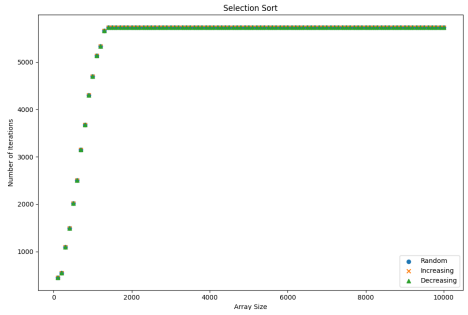

Fig. 3. Número de utilizações do Selection Sort para diferentes tamanhos de arrays.

Os resultados da tabela mostram o número de utilizações de cada algoritmo de ordenação para diferentes tamanhos de arrays. Observa-se que o Bubble Sort tem a maior contagem de utilizações, seguido pelo Insertion Sort e pelo Selection Sort.

VI. Conclusão

Com base nos resultados obtidos, conclui-se que o algoritmo de ordenação por seleção é o menos performático, pois o número de iterações cresce exponencialmente e estabiliza em um grande número de iterações. O método de inserção é o melhor, pois tem um comportamento semelhante às distintas distribuições de elementos nos vetores, enquanto que o Bubble Sort é muito performático com vetores em ordem decrescente, apresentando uma performance razoável com vetores aleatórios e diminuindo a performance em uma relação tamanho ao quadrado do vetor crescente.

VII. Referências Bibliográficas

1) Knuth, D.E. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1997.
2) Sedgewick, R. *Algorithms in C++, Parts 1-4: Fundamentals, Data Structure, Sorting, Searching*. Addison-Wesley Professional, 1997.
3) Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. *Introduction to Algorithms*. MIT Press, 2009.
4) Aho, A.V., Hopcroft, J.E., Ullman, J.D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
5) Bentley, J.L. *Programming Pearls*. Addison-Wesley Professional, 1999.