ANÁLISE DE ALGORITMOS DE ORDENAÇÃO

Maciel Freire, Anna Carolina
Graduanda em Engenharia de computação - IFCE
Silveira Araújo, Tallita Maria
Graduanda em Engenharia de computação - IFCE
Santedicola La Serra, Angelo Gabriele
Graduando em Engenharia de computação - IFCE

RESUMO

Este trabalho apresenta as características de quatro dentre os vários métodos de ordenamento estudados na disciplina de Pesquisa e Ordenação do curso de Engenharia de Computação do Instituto Federal do Ceará. Serão comparados o tempo de execução de cada um e seus pontos positivos e negativos.

Palavras-chave: bubble sort, selection sort, counting sort, merge sort, algoritmo.

1. INTRODUÇÃO

Os métodos de ordenamento são essenciais em diversas áreas da computação. Verificar qual o tipo e quando utilizar o mais adequado é essencial para que a programação seja eficiente.

Este artigo mostra a funcionalidade de quatro destes métodos, com base em seus tempos de execução e através da análise assintótica de cada um, ou seja, em quanto tempo a operação é executada no pior dos casos.

> "Isto é, estamos preocupados com o modo como o tempo de execução de um algoritmo aumenta com o tamanho da entrada no limite, à medida que o tamanho da entrada aumenta sem limitação. Em geral, um algoritmo que é assintoticamente mais eficiente será a melhor escolha para todas as entradas, exceto as muito pequenas." (CORMEN, T. H. Algoritmos Teoria e Prática. 3.ed. Página 33. Rio de Janeiro: Elsevier: Editora Ltda: 2012.)

Foram analisados o Bubble Sort e o Selection Sort, considerados métodos simples, e o Merge Sort e o Counting Sort, métodos mais sofisticados. Todos estes

são propostos pelo Programa de Unidade Didática do Instituto Federal de Educação Ciência e Tecnologia do Ceará, no curso de Engenharia de Computação.

2. DESENVOLVIMENTO

Para o estudo de cada método, foi observado o tempo de execução gasto quando estes ordenam uma lista grande de números. O ambiente de desenvolvimento integrado online escolhido foi o Repl.it, com linguagem de programação Python 3, num computador Imac de processador Intel core i5 e memória RAM de 8GB.

Todos os experimentos utilizaram-se de funções que geram listas de números crescentes, decrescentes e aleatórios, para que fosse possível observar o comportamento da ordenação através de gráficos.

2.1. Bubble Sort

O método de ordenação Bubble Sort é um algoritmo de ordenação simples, cujo objetivo é percorrer o vetor permutando repetidamente elementos adjacentes que estão fora de ordem. Seu tempo de execução no pior caso é $\theta(n^2)$, que ocorre quando o vetor recebe uma lista em ordem decrescente. No melhor caso o tempo é de $\theta(n)$, quando o vetor recebe uma lista já em ordem crescente. Assim sendo, observa-se no gráfico da figura 1(a) que a curva de tempo de execução é exatamente uma parábola em seu pior caso. Quando a lista está em ordem crescente, no melhor caso do algoritmo, pode-se observar uma reta.

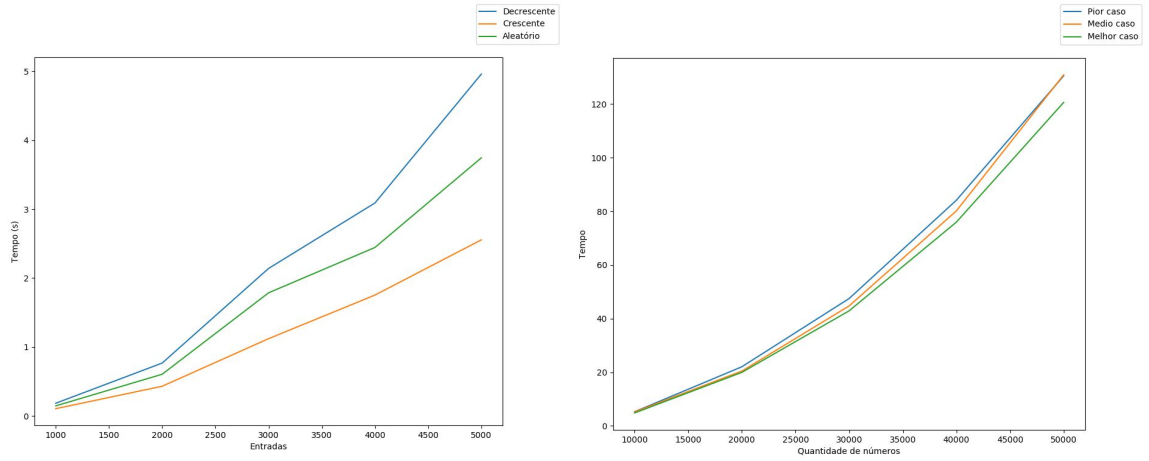
Um dos pontos positivos deste método é a sua simplicidade de implementação e a não utilização de memória extra. Por outro lado, a demora na execução é grande ao receber uma lista com muitos números.

2.2. Selection Sort

O Selection Sort é um método simples utilizado para ordenar listas. Em sua execução, o vetor é percorrido à procura de um pivô que seja o menor entre todos os elementos. Após a sua escolha, troca-se este com seu sucessor e o algoritmo continua neste processo de escolha e troca para os n-1 elementos da lista. Como o Selection Sort comparar sempre os elementos uns com os outros em cada iteração,

pode-se entender que não existe melhor ou pior caso. No gráfico da figura 1(b) pode-se observar que o tempo de execução $\theta(n^2)$ é o mesmo para todos os casos.

Este algoritmo tem a vantagem de ser um dos mais velozes para ordenar vetores pequenos e de não se utilizar de nenhum tipo de memória externa. Mas para vetores grandes não é estável, fazendo sempre $(n^2 - n)/2$ comparações.



a) Bubble Sort b) Selection Sort

Figura 1 - Casos dos métodos simples

Fonte: elaborado pelos autores (2019).

2.3. Merge Sort

O Merge Sort, também conhecido como ordenação por mistura, é um exemplo de algoritmo de comparação do tipo dividir-para-conquistar. Primeiro ele calcula o ponto médio do vetor dado, o que fará com que ele utilize um tempo constante. Depois ele usa a recursividade para ordenar dois subvetores, cada um com tamanho de $n/2$, que somam $2t(n/2)$ para o tempo de execução. Posteriormente, ele une os subvetores criados em um único vetor ordenado.

No caso deste algoritmo, podemos falar em melhor caso, quando não é necessário trocar após comparações. Em médio caso, quando há necessidade de troca. E, por fim, em pior caso, quando sempre é necessário fazer trocas a cada comparação. Em todos os casos, o algoritmo levará um tempo de $\theta(n \log n)$, como mostra a figura 2(a).

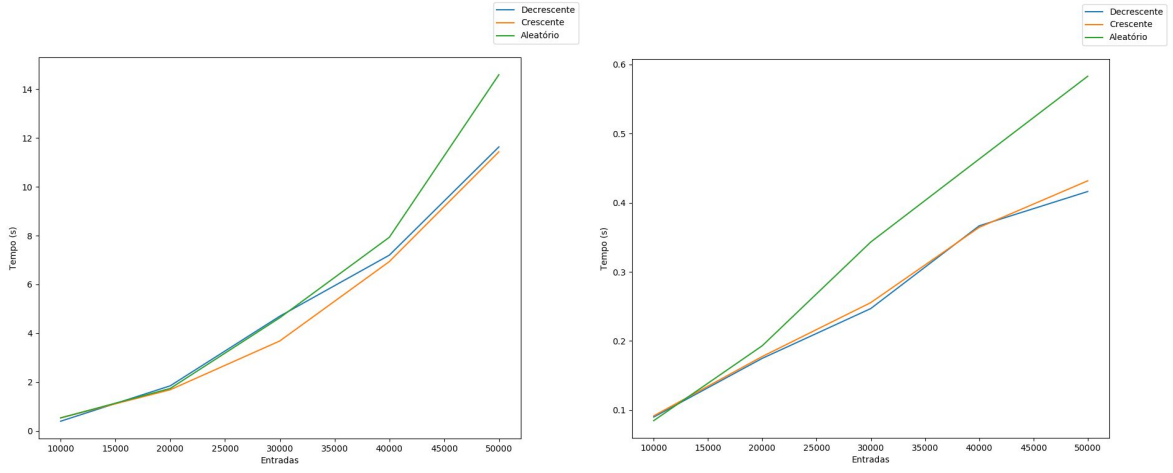
Entre as vantagens de utilizá-lo está a sua fácil implementação, útil para ordenação externa e sua aplicação com restrição de tempo. Entre suas

desvantagens, está a utilização de memória auxiliar e o seu alto consumo de memória.

2.4. Counting Sort

O counting sort é um algoritmo que ordena valores inteiros, tem complexidade linear $\theta(m+n)$, como mostrado na figura 2(b) e se utiliza de dois vetores auxiliares: o primeiro, com tamanho igual ao maior valor da entrada acrescido de 1, que guarda as informações sobre as ocorrências dos valores da lista de entrada, e o segundo, com mesmo tamanho da lista de entrada, que, ao final da execução, armazenará a lista já ordenada.

Para este algoritmo, o caso que gasta mais tempo na ordenação é o de uma lista com valores dispostos aleatoriamente. Tanto para uma lista ordenada de forma crescente quanto para uma de forma decrescente, o tempo de execução é similar, como mostra o gráfico da figura 2(b). Apesar de sua rápida execução, independente do tamanho do vetor de entrada, o uso de memória é muito alto e cresce à medida que aumenta o valor do maior elemento da lista a ser ordenada.



a) Merge Sort b) Counting Sort

Figura 2 - Casos dos métodos sofisticados

Fonte: elaborado pelos autores (2019).

3. CONSIDERAÇÕES FINAIS

Ao se comparar os quatro algoritmos analisados, observa-se que cada um traz vantagens e desvantagens ao serem aplicados. Conhecer estes fatores é

essencial para que possa ser escolhido o método mais adequado às necessidades da programação.

O Bubble Sort, apesar de ser simples e de fácil implementação, tem um tempo de execução extremamente alto quando utilizado para ordenar listas muito grandes, o que o torna pior do que os demais. O Selection Sort, apesar de ser bastante eficiente para vetores pequenos e não utilizar memória externa, quando utilizado para vetores grandes, ele é extremamente instável para ordenar. O Counting Sort não é nada econômico em seu uso de memória. No entanto, sua implementação é razoavelmente intuitiva e seu tempo de execução é linear na maioria dos casos. Por fim, o Merge Sort que, apesar de ter um excessivo uso de memória, utiliza-se de uma fácil implementação e não há restrição em relação ao uso de tempo.

4. REFERÊNCIAS BIBLIOGRÁFICAS

CORMEN, T. H. **Algoritmos Teoria e Prática**. 3.ed. Rio de Janeiro: Elsevier: Editora Ltda: 2012.

Selection Sort. Disponível em: <https://pt.wikipedia.org/wiki/Selection_sort>. Acessado em: 27 de abril de 2019

Bubble Sort. Disponível em: <https://pt.wikipedia.org/wiki/Bubble_sort>. Acessado em: 27 de abril de 2019

Counting Sort. Disponível em: <https://felipepriuli.wordpress.com/2013/01/08/counting-sort/>. Acessado em: 28 de abril de 2019

Algoritmo Merge Sort. Disponível em: <https://www.academia.edu/6092282/Algoritmo_MergeSort>. Acessado em: 29 de abril de 2019