**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 vistos na disciplina de pesquisa e ordenação do curso de Engenharia de Computação do Instituto Federal do Ceará, veremos como funcionam, o tempo de execução de cada um no pior caso e seus pontos positivos e negativos.

**Palavras-chave:** bubble sort, selection sort, counting sort, merge sort, algoritmo, velocidade, tempo, execução.

**1. INTRODUÇÃO**

Este artigo tem o objetivo de mostrar um comparativo entre o funcionamento de quatro métodos de ordenação implementados em sala de aula, será apresentado seus tempos de execução através da análise assintótica utilizando métodos de análise de algoritmos. Analisaremos, assim, o bubble sort e o selection sort, considerados métodos simples, e o counting sort e o merge sort, métodos mais sofisticados.

**2. DESENVOLVIMENTO**

Todos os experimentos foram realizados utilizando cada um dos algoritmos de ordenação propostos pelo Programa de Unidade Didática do Instituto Federal de Educação Ciência e Tecnologia do Ceará do curso de Engenharia de Computação. Foram escolhidos por esta equipe o método bolha, seleção, counting sort e merge sort. O propósito do trabalho será observar o tempo gasto na execução dos algoritmos quando estes ordenam uma lista grande de números. O ambiente de desenvolvimento integrado online escolhido foi o Replit que dá suporte a várias linguagens de programação e também bibliotecas gráficas, a linguagem de programação Python 3 para a construção do algoritmo e o computador Imac intel core i5 e memória de 8GB.

Quando analisa-se o tempo de execução de um algoritmo frequentemente estão interessados em sua complexidade temporal assintótica,logo,quanto tempo o algoritmo demora para executar uma operação 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.)

Todos os experimentos utilizaram a mesma função gera lista para gerar números aleatório de tamanhos 10.0000 a 50.000 para se observar o comportamento da ordenação através de gráficos.

**2.1. Bubble Sort**

O método de ordenação bubble sort ou bolha 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 que ocorre quando o vetor recebe uma lista em ordem decrescente e no melhor caso quando o vetor recebe uma lista já em ordem crescente.

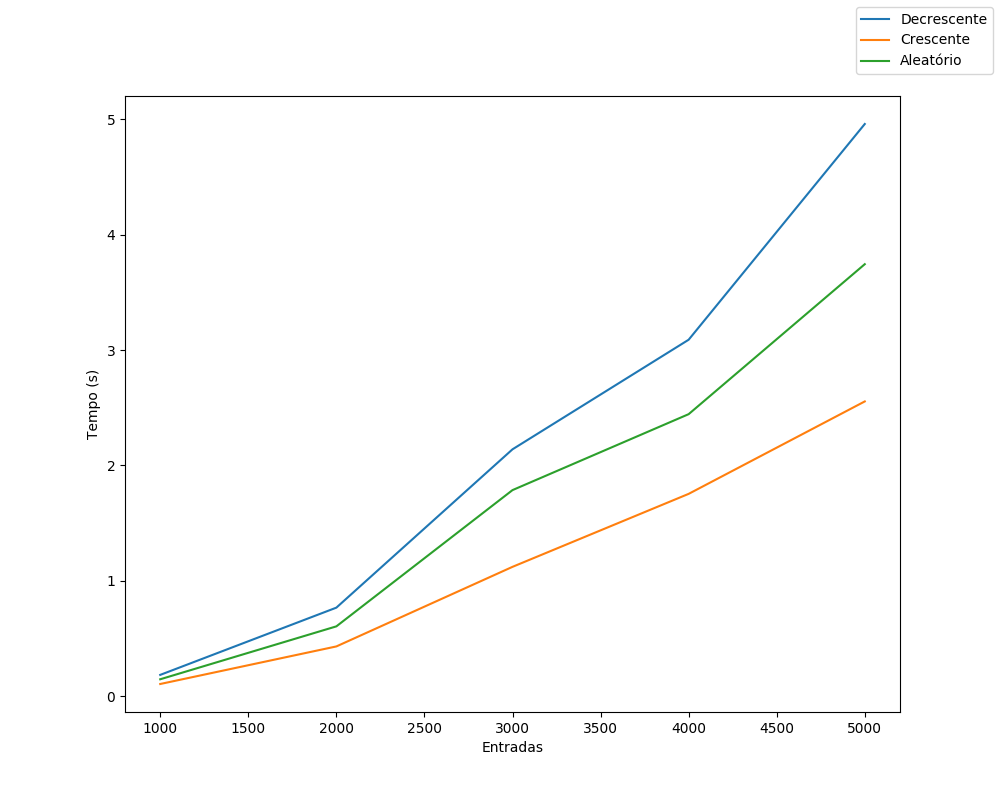
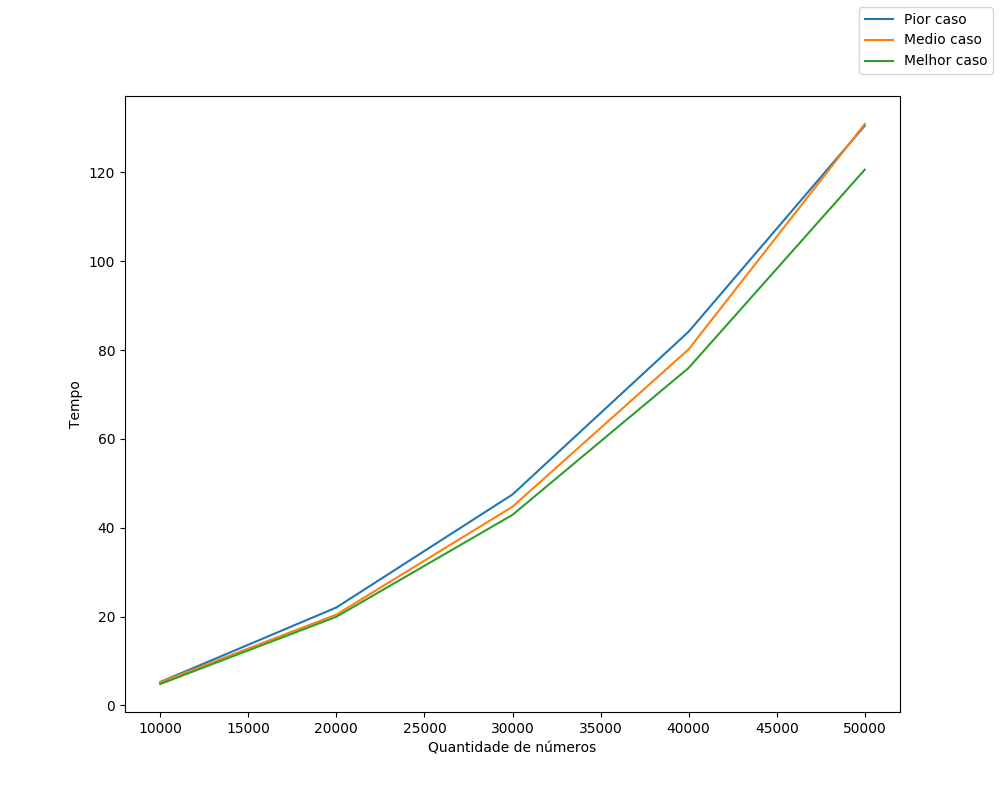
No gráfico da figura 1(a) pode-se observar que a curva de tempo de execução é exatamente uma parábola como foi descrito sendo o pior caso do bubble sort. Quando a lista está ordenada em ordem crescente, melhor caso do algoritmo, pode-se ver uma reta como o tempo de execução como foi previsto na complexidade assintótica deste**.**

Um dos pontos positivos do bubble sort é ele ser simples e de fácil implementação, porém apresenta como ponto negativo a demora da ordenação quando este recebe uma lista muito grande de números. O algoritmo não utiliza nenhum tipo de memória extra em sua implementação, sendo assim sua complexidade espacial é O(1).

**2.2. Selection Sort**

O método de seleção é considerado um dos métodos simples utilizados para ordenar listas, seu funcionamento se dá através de se percorrer o vetor a procura de um pivô, sendo ele o menor elemento entre todos da lista, após a escolha do pivô troca-se este com seu sucessor e o algoritmo continua neste processo de escolha e troca para os n-1 elementos do vetor. Como o selection sort sempre tem que comparar os elementos uns com os outros em cada iteração visando encontrar o menor, pode-se entender que não existe um melhor ou pior caso pois ele sempre tem que realizar as duas iterações do laço para os n elementos do vetor. Logo abaixo pode-se observar a implementação e o gráfico gerado pelo selectionsort utilizado no experimento, ele possui sempre o tempo de execução , como pode-se observar na figura 1(b).

Um dos pontos positivos do selection sort é ele ser um dos mais velozes para ordenar vetores pequenos, porém apresenta como ponto negativo não ser estável, fazendo sempre comparações. O algoritmo não utiliza nenhum tipo de memória extra em sua implementação. Sua complexidade espacial é O(1).

****

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. Primeiramente, ele calcula o ponto médio do vetor dado, o que fará com que ele utilize um tempo constante Θ(1). Depois ele utilizará a recursividade para resolver dois subproblemas, cada um com tamanho de n/2, o que soma 2T(n/2) para o tempo de execução. Posteriormente, ele unirá os subvetores criados em um único vetor ordenado, o que utilizará um tempo de Θ(n).

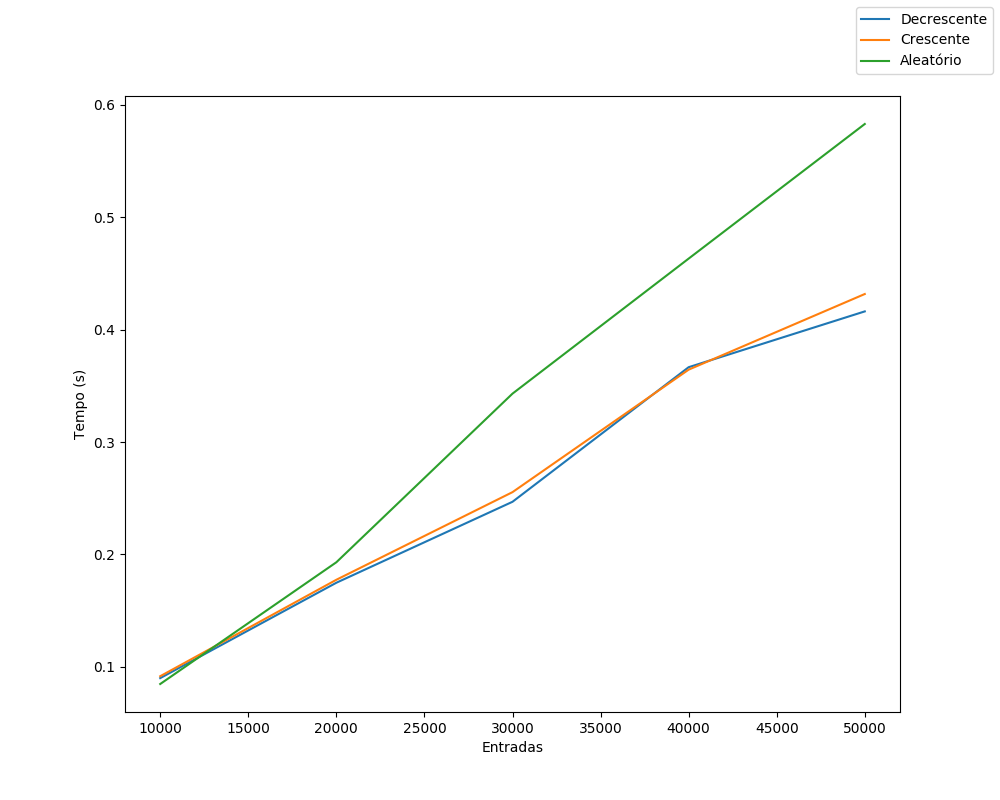
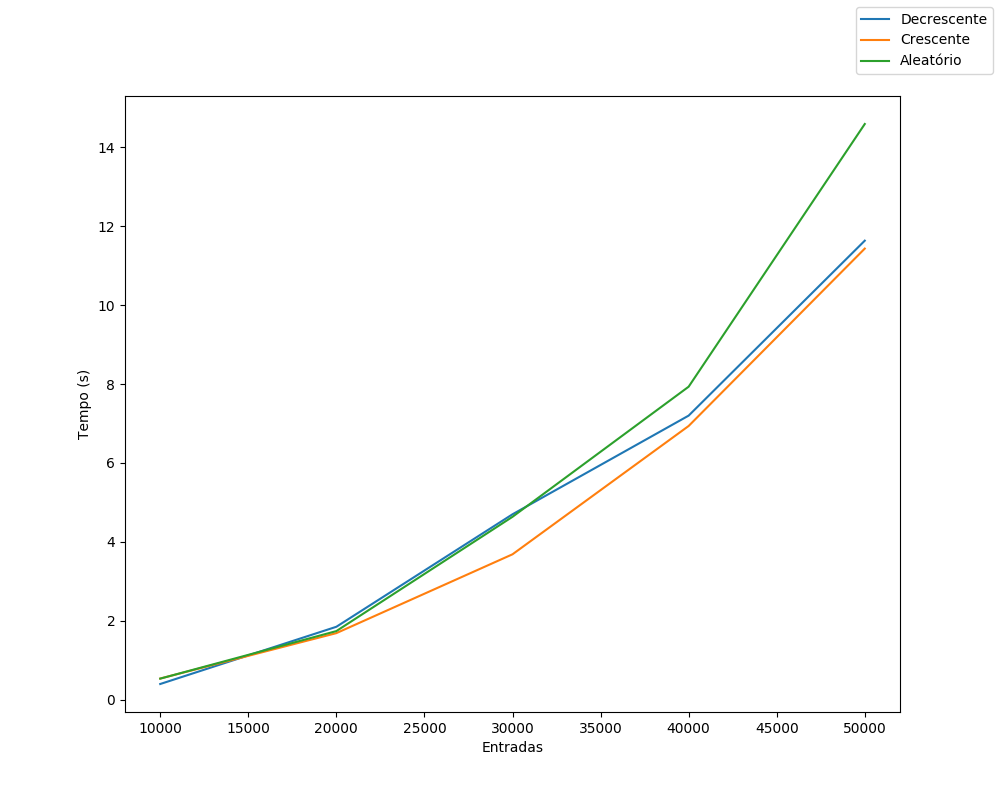
No caso deste algoritmo, podemos falar em melhor caso, quando não é necessário trocar após comparações. Em médio caso, quando há necessidade de haver troca após comparações. E, por fim, em pior caso, quando sempre é necessário fazer trocas em cada comparação. Em todos os casos citados, o algoritmo levará um tempo de Θ(n log n).

Como em todo algoritmo, o merge sort possui pontos positivos e pontos negativos. Entre as vantagens de utilizá-lo está a sua fácil implementação, útil para ordenação externa e sua aplicação com restrição de tempo. E, 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 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 um lista ordenada de forma crescente quanto para uma de forma decrescente, o tempo de execução é similar, como mostra o gráfico.



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 anteriores, se observa que cada um poderá ter vantagens para seu uso, assim como desvantagens. O bubble sort, que apesar de ser simples e de fácil implementação, possui uma lentidão que o torna pior do que os demais que foram abordados. O counting sort, por outro lado, de longe não é econômico em seu uso de memória sendo este uso maior quanto tão maior for o valor máximo do vetor a ser ordenado. No entanto, sua implementação é razoavelmente intuitiva, e seu tempo de execução, linear, em muitos casos. Por fim, o merge sort que, assim como o counting sort, apesar de ter um excessivo uso de memória ele 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

**Merge Sort.** Disponível em: <<https://pt.wikipedia.org/wiki/Merge_sort>>.Acessado em: 29 de abril de 2019

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