

Relatório de análise Algoritmos de Ordenação

João Guilherme Squaris Merlin ¹

¹PUCPR - Pontifícia Universidade Católica do Paraná (PUCPR)
R. Imac. Conceição, 1155 - Prado Velho, Curitiba - PR, 80215-901

Resumo. *O relatório visa fornecer insights sobre o desempenho de algoritmos de (BubbleSort, QuickSort e HeapSort), analisando a média de tempo (em nanosegundos) de execução, trocas e iterações com vetores de tamanhos variados (50, 500, 1000, 5000 e 10000) preenchidos com números aleatórios.).*

1. Função de Ordenação BubbleSort

O algoritmo de ordenação Bubble sort, é uma técnica de ordenação simples. Sua abordagem envolve iterar várias vezes por um conjunto de elementos, durante cada iteração, movendo o maior elemento para a posição correta. Essa movimentação é análoga à maneira como as bolhas em um tanque de água se deslocam para alcançar seu próprio nível, originando assim o nome do algoritmo.

A complexidade deste algoritmo é de

$$O(n^2)$$

exemplificado na figura 2 em casos de vetores não ordenados, como o esperado, esse é o resultado obtido (Figura 1) com a média dos tempos refletindo a complexidade do algoritmo.

Já para as trocas efetuadas e média de iterações, o resultado obtido (Figura 3 e Figura 4 respectivamente), é coerente com o que se espera desse algoritmo

A média do número de trocas foi, respectivamente para cada vetor: 45; 976,6; 2.136,80; 15.514,80; 3,58E+04.

O número de iterações foi, respectivamente para cada vetor: 1225; 1,25E+05; 5,00E+05; 1,25E+07; 5,00E+07.

2. Função de Ordenação Shell Sort

O Shell sort, criado por Donald Shell em 1959 e publicado pela Universidade de Cincinnati, é um algoritmo de classificação altamente eficiente com complexidade quadrática. Ele aprimora o método de inserção direta, abordando a lista a ser ordenada em vários segmentos e aplicando o método de inserção direta a cada um deles. Isso envolve repetidamente dividir o grupo maior em grupos menores, resultando em uma ordenação eficiente.

Novamente, a média dos tempos de execução nos entregou um resultado esperado.

Entretanto, é importante ressaltar que há fontes afirmando que esse algoritmo não tem complexidade definida, mas essas fontes não terão peso para essa análise.

Os gráficos de trocas (Figura 6) e iterações (Figura 7) seguem como o esperado pela complexidade do algoritmo, com uma crescente exponencial.

Média de trocas respectivamente para cada vetor ordenado por Shell Sort: 127,4; 3.020,40; 7.325,40; 60.389,80; 140.942,60

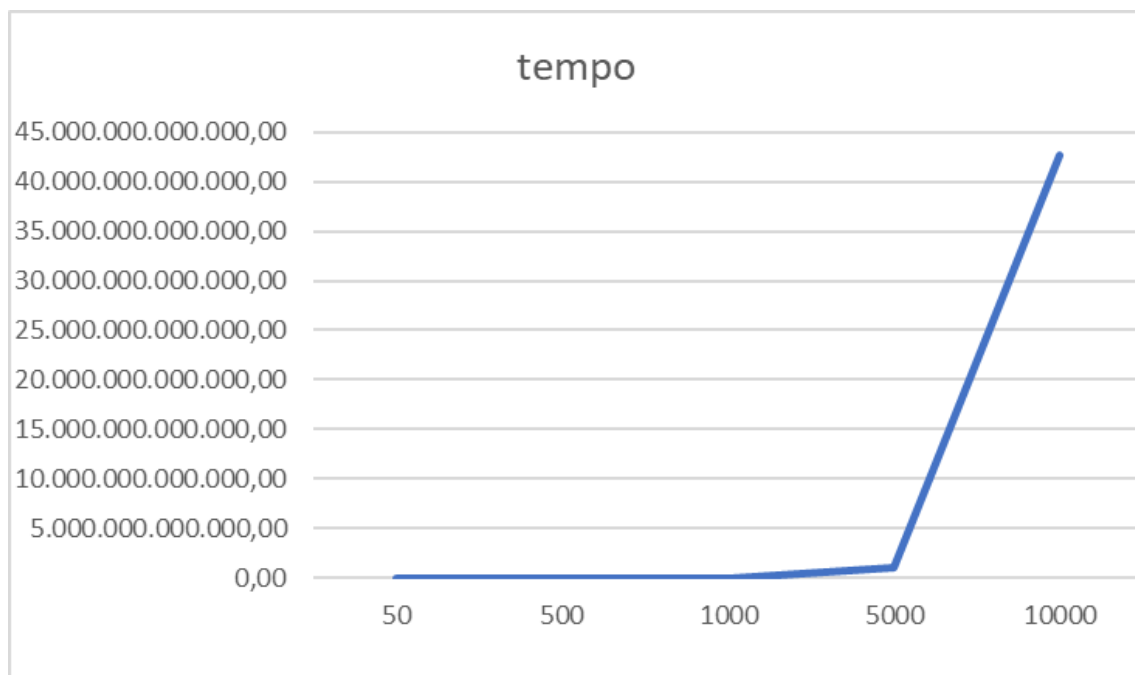


Figure 1. Média de tempo para algoritmo Bubble Sort

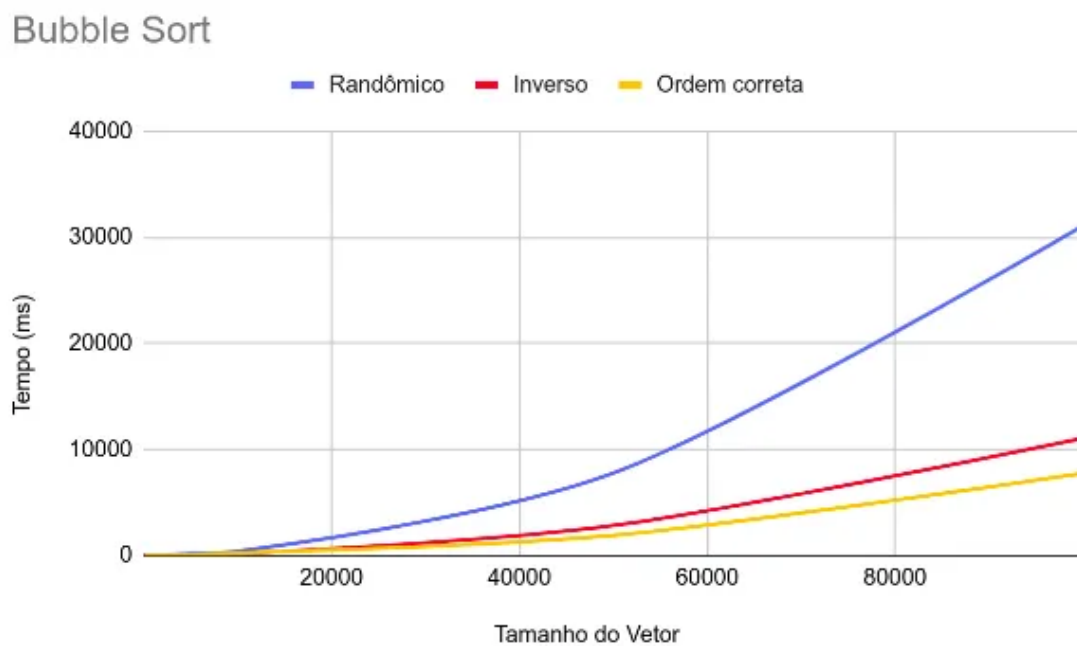


Figure 2. Resultado esperado para a média de tempo do algoritmo Bubble Sort

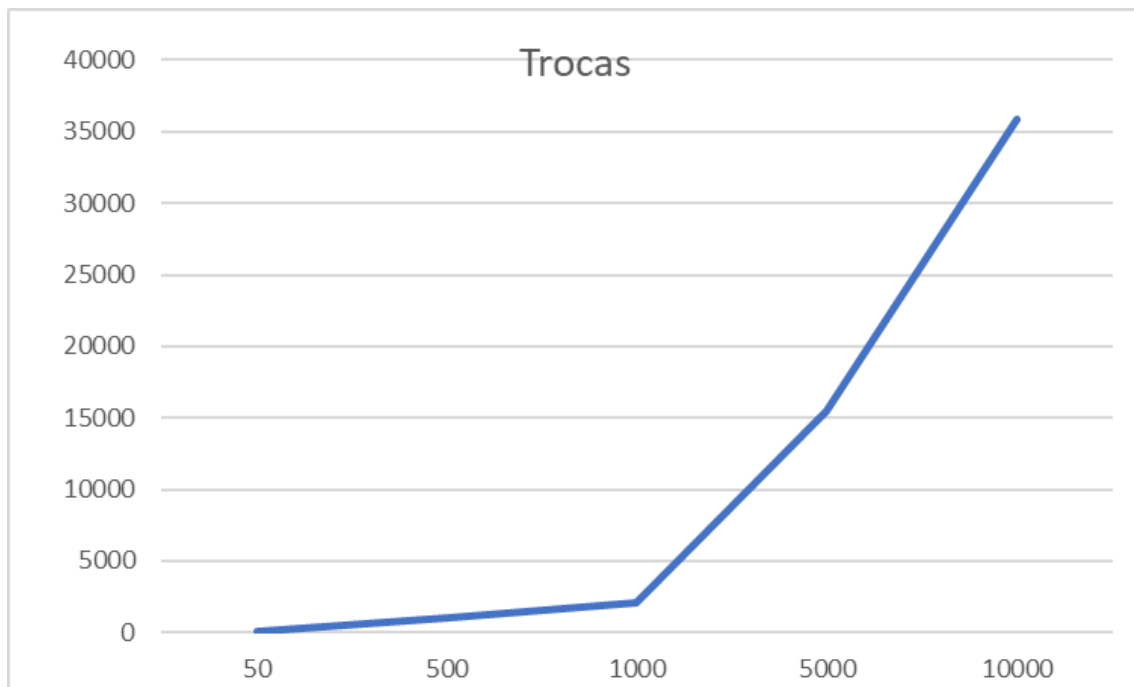


Figure 3. Gráfico de Iterações no algoritmo Bubble Sort

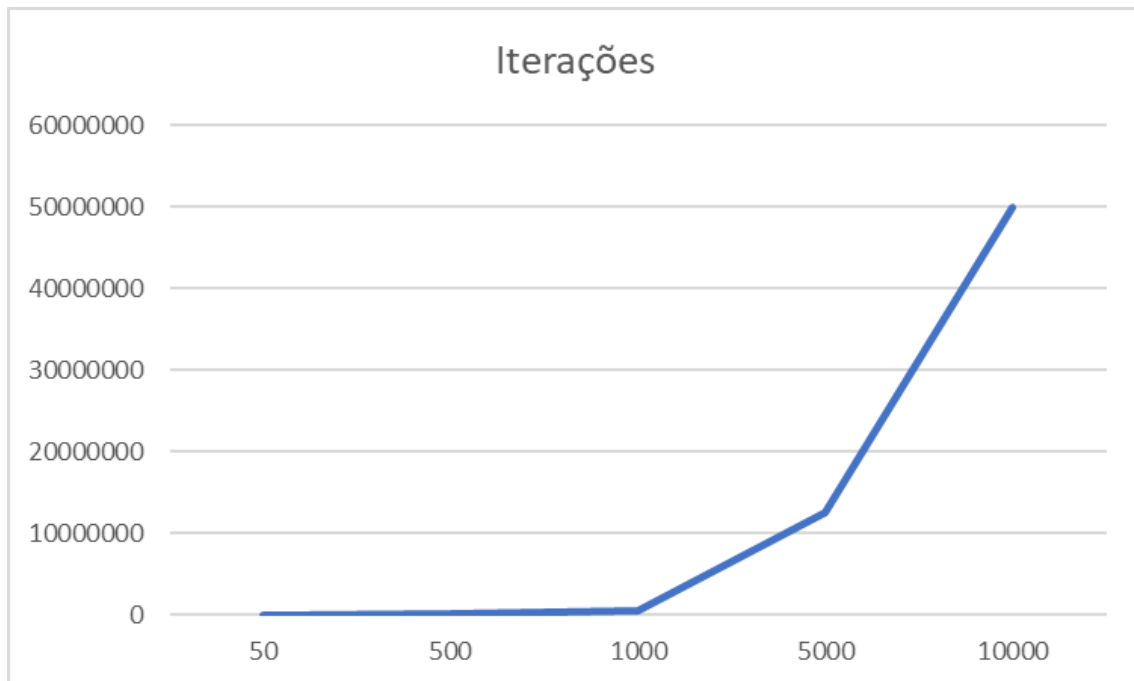


Figure 4. Gráfico de Iterações no algoritmo Bubble Sort

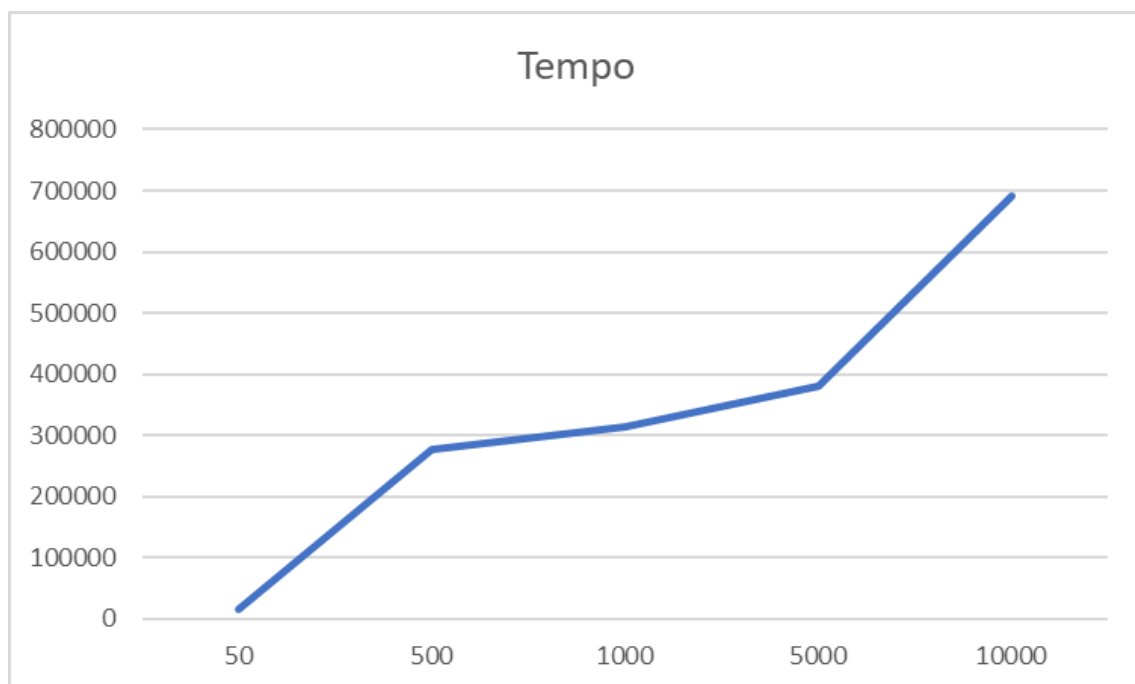


Figure 5. Média de tempo para algoritmo Shell Sort

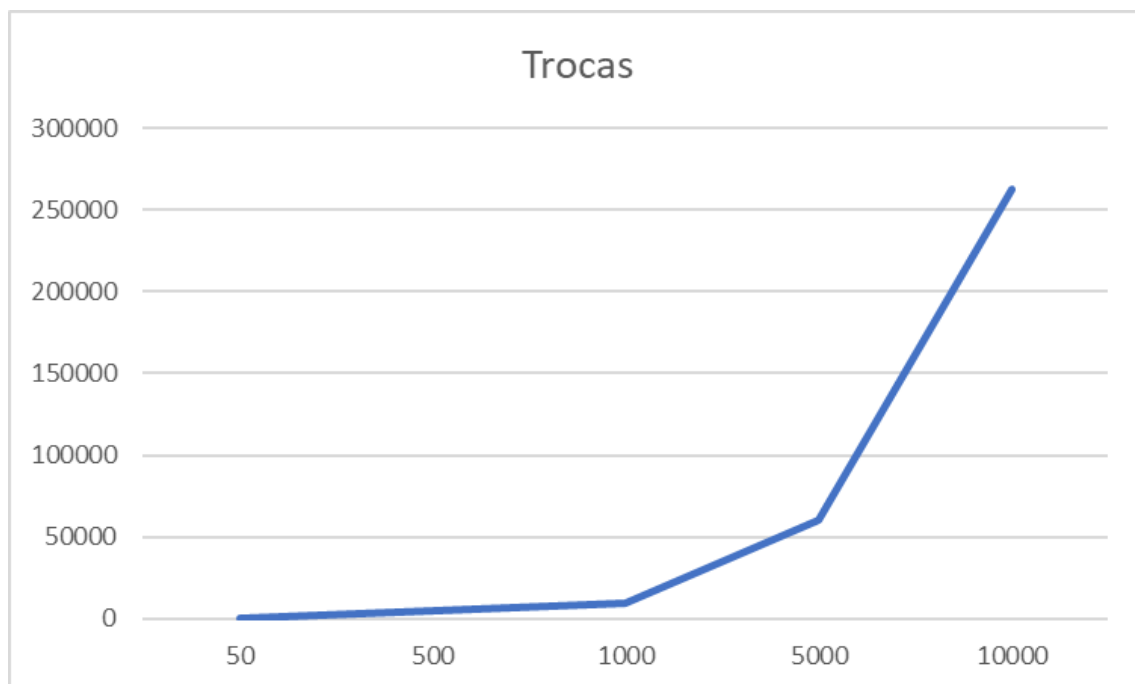


Figure 6. Gráfico de Iterações no algoritmo Shell Sort

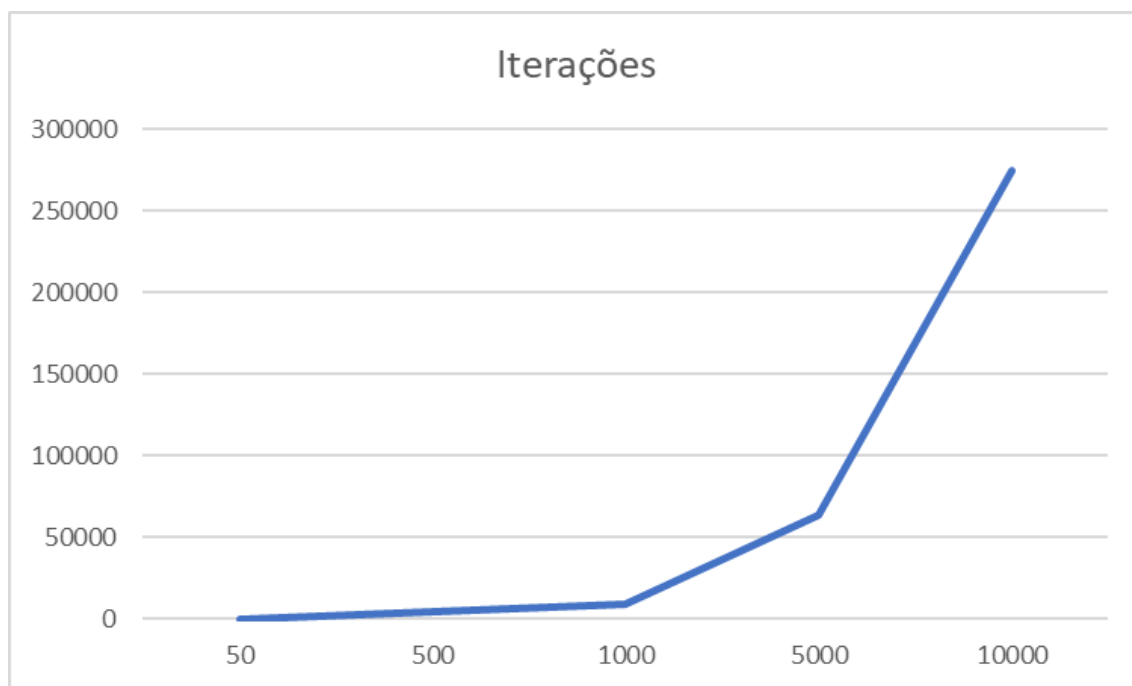


Figure 7. Gráfico de Iterações no algoritmo Shell Sort

3. Função de Ordenação Heap Sort

O algoritmo possui excelente desempenho em conjuntos aleatórios, uso eficiente de memória e desempenho quase constante em cenários médio e pior cenários. Isso contrasta com algoritmos de ordenação rápida que podem ter desempenho ruim no pior cenário. O heapsort opera no lugar e tem complexidade de $O(n \log n)$ no pior caso para ordenar n elementos.

Gráfico registrando a média de tempo do algoritmo (Figura 8), um resultado inusitado para o vetor de tamanho 5000.

Quanto as médias de trocas (Figura 9) e iterações (Figura 10, o resultado foi o correto de acordo com a complexidade $O(n \log n)$).

Média de trocas respectivamente para cada vetor ordenado por Heap Sort: 263,8; 4.318,60; 9.657,60; 60.605; 262.389,60.

Número de Iterações:

4. Considerações finais e resultado obtido

Em suma, o resultado obtido foi coerente com o que foi estudado e com o que era esperado, com exceção de algumas dúvidas nas médias dos tempos de execução, o resto dos dados obedeceu a respectiva complexidade de cada algoritmo e agregou positivamente para construção de conhecimento.

Entretanto, as dificuldade desse trabalho é similar ao do outro, além da falta de tempo, cobrar o relatório no formato latex foi uma dificuldade a mais, (temo que a formatação deste documento também tenha ficado errada).

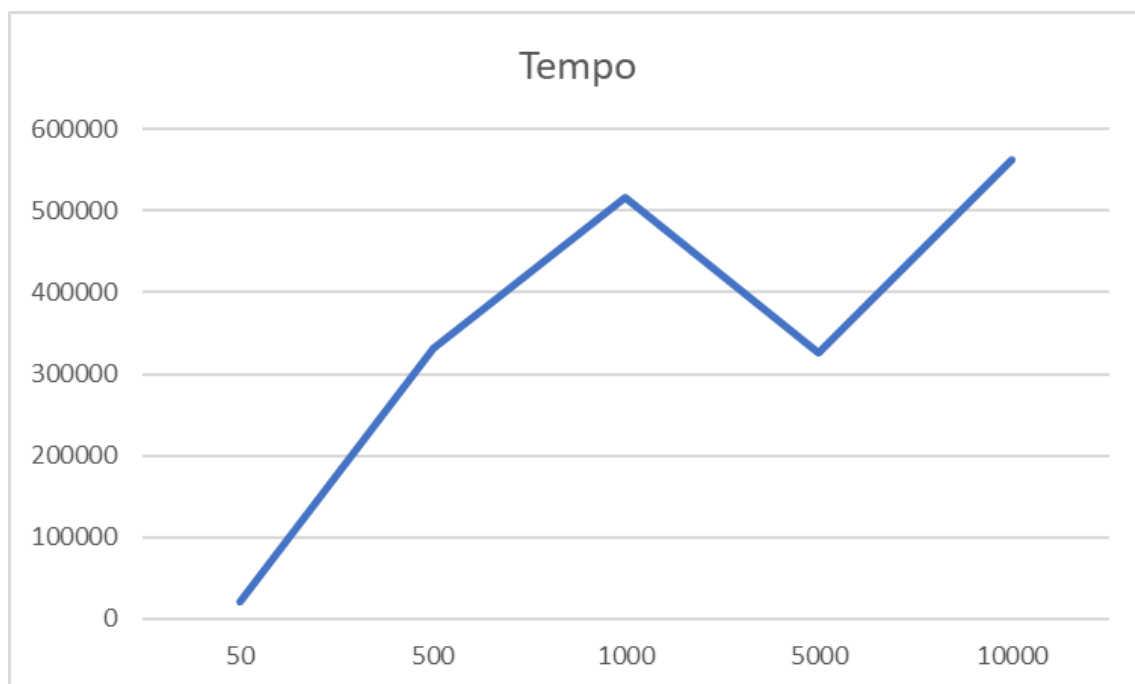


Figure 8. Média de tempo para algoritmo Heap Sort

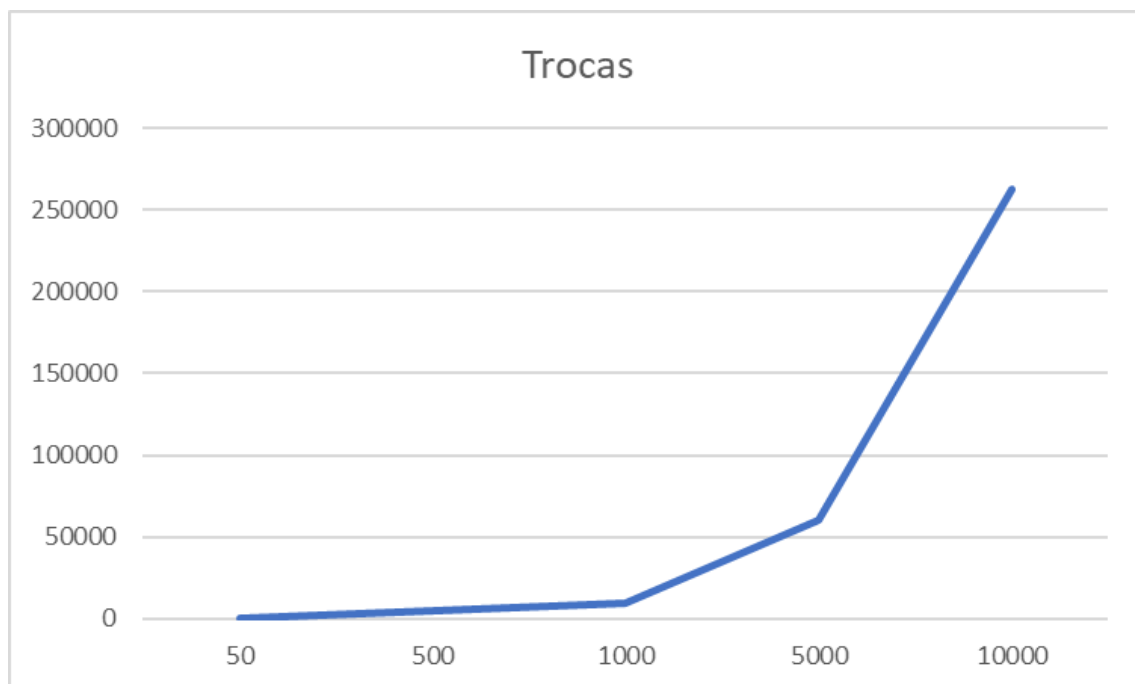


Figure 9. Gráfico de Trocas no algoritmo Heap Sort

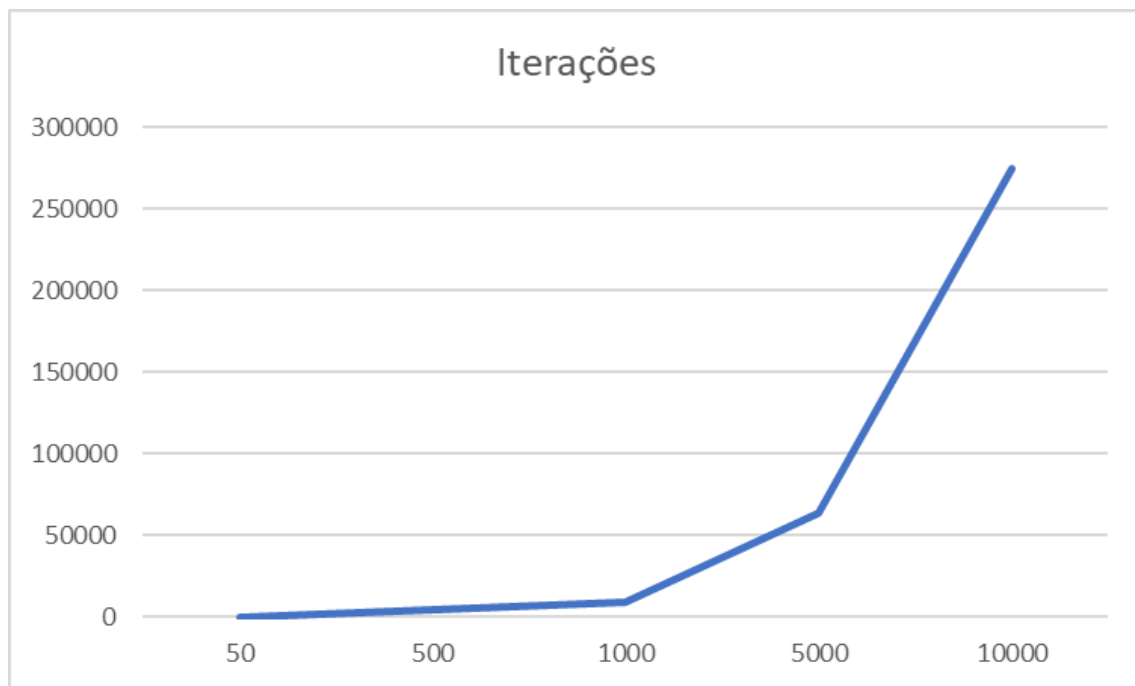


Figure 10. Gráfico de Iterações no algoritmo Heap Sort

5. Referencias

Wikipedia. Shell Sort. Disponível em: https://pt.wikipedia.org/wiki/Shell_sort. Acesso em : 07 de novembro de 2023.

Wikipedia. Heapsort. Disponível em: <https://pt.wikipedia.org/wiki/Heapsort>. Acesso em: 07 de novembro de 2023.

Wikipedia. Bubble Sort. Disponível em: https://pt.wikipedia.org/wiki/Bubble_sort. Acesso em : 07 de novembro de 2023.

Autor(es). Eficiência de Algoritmos: Ordenando com Bubble Sort, Selection Sort e Random Sort. Turing Talks, Disponível em: <https://medium.com/turing-talks/eficiencia-de-algoritmos-ordenando-com-bubble-sort-selection-sort-e-random-sort-382e04b2f523>. Acesso em: 07 de novembro de 2023.

Título da Página. Instituto de Matemática e Estatística da Universidade de São Paulo, Disponível em: https://www.ime.usp.br/cef/mac122-2000/eps/ep3-destaques/alex_murakami/Ep3.html. Acesso em : 07 de novembro de 2023.

Autor(es) (se disponível). Conheça os Principais Algoritmos de Ordenação. TreinaWeb, Disponível em: <https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao>. Acesso em: 07 de novembro de 2023.

References