# Relatório de desempenho

*Algoritmos de ordenação: Shellsort vs. Heapsort*

***Aluna:*** *Izabela Garcia Tavares Pinheiro Pérez*

1. **Método de escolha e iteração dos h's (Shell Sort)**

   Foi utilizada a sequência de Sedgewick para definir os intervalos. Essa sequência é criada recursivamente através da fórmula:

   $$sedgewick(i) = 9 * (2\hat{}i - 2\hat{}(i/2)) + 1$$

   Onde i é o índice da sequência, iniciando em 0, ela é baseada em potências de 2 e números ímpares, e é conhecida por gerar um conjunto eficiente de *"gaps"* para o Shell Sort.

   Apesar de ser uma sequência infinita, foi utilizado apenas um pequeno intervalo para a execução dos testes, mostrada abaixo:

   $$\{1, 5, 19, 41, 109, 209, 505, 929, 2161, 3905, 8929, 16001, 36289, 64769, 146305, 260609\}$$

   Ao usar essa sequência, o Shellsort se beneficiou de uma melhor eficiência em relação à sua ordem de complexidade **média** *O(n log n)*, chegando próximo a *O(n)* na maioria dos casos testados.

   Portanto, isso contribuiu para a melhor eficiência do Shellsort em relação ao tempo de execução quando comparado ao Heapsort que é sempre *O(n log n)* , que pode ser observada nos resultados dos testes mais adiante.

2. **Tamanho dos vetores**

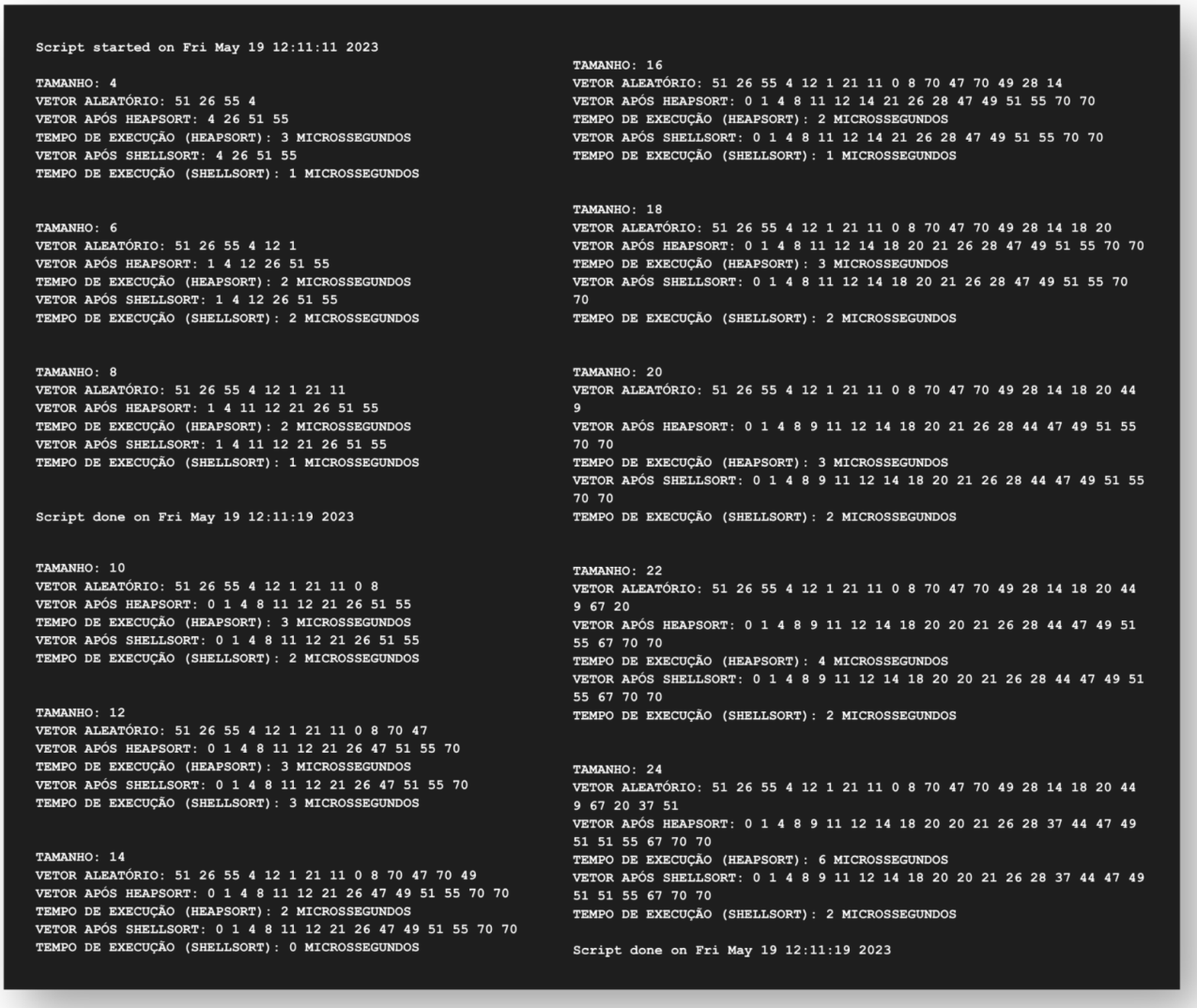
   O intervalo de tamanhos utilizado para testar a ordenação de vetores nos métodos de ordenação são os números pares de 4 a 24. A inicialização de tais vetores é feita com números inteiros de 0 a 80, utilizando a função *rand()*.

   Além disso, a leitura desses tamanhos é feita dentro de um arquivo txt contendo o intervalo desejado, ou seja, é possível alterá-lo para testar com outros tamanhos de vetores.

3. **Resultado dos testes (saída no terminal)**

   Em todos os testes foram impressas saídas no terminal indicando o tamanho dos vetores, a inicialização aleatória dos valores do vetor a ser testado, a saída após a ordenação dos dois métodos e por fim o tempo de execução de cada em microssegundos.

Veja a imagem abaixo com as saídas:



```
Script started on Fri May 19 12:11:11 2023

TAMANHO: 4
VETOR ALEATÓRIO: 51 26 55 4
VETOR APÓS HEAPSORT: 4 26 51 55
TEMPO DE EXECUÇÃO (HEAPSORT): 3 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 4 26 51 55
TEMPO DE EXECUÇÃO (SHELLSORT): 1 MICROSSEGUNDOS


TAMANHO: 6
VETOR ALEATÓRIO: 51 26 55 4 12 1
VETOR APÓS HEAPSORT: 1 4 12 26 51 55
TEMPO DE EXECUÇÃO (HEAPSORT): 2 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 1 4 12 26 51 55
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS


TAMANHO: 8
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11
VETOR APÓS HEAPSORT: 1 4 11 12 21 26 51 55
TEMPO DE EXECUÇÃO (HEAPSORT): 2 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 1 4 11 12 21 26 51 55
TEMPO DE EXECUÇÃO (SHELLSORT): 1 MICROSSEGUNDOS


Script done on Fri May 19 12:11:19 2023


TAMANHO: 10
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8
VETOR APÓS HEAPSORT: 0 1 4 8 11 12 21 26 51 55
TEMPO DE EXECUÇÃO (HEAPSORT): 3 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 11 12 21 26 51 55
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS


TAMANHO: 12
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47
VETOR APÓS HEAPSORT: 0 1 4 8 11 12 21 26 47 51 55 70
TEMPO DE EXECUÇÃO (HEAPSORT): 3 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 11 12 21 26 47 51 55 70
TEMPO DE EXECUÇÃO (SHELLSORT): 3 MICROSSEGUNDOS


TAMANHO: 14
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49
VETOR APÓS HEAPSORT: 0 1 4 8 11 12 21 26 47 49 51 55 70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 2 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 11 12 21 26 47 49 51 55 70 70
TEMPO DE EXECUÇÃO (SHELLSORT): 0 MICROSSEGUNDOS


TAMANHO: 16
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49 28 14
VETOR APÓS HEAPSORT: 0 1 4 8 11 12 14 21 26 28 47 49 51 55 70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 2 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 11 12 14 21 26 28 47 49 51 55 70 70
TEMPO DE EXECUÇÃO (SHELLSORT): 1 MICROSSEGUNDOS


TAMANHO: 18
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49 28 14 18 20
VETOR APÓS HEAPSORT: 0 1 4 8 11 12 14 18 20 21 26 28 47 49 51 55 70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 3 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 11 12 14 18 20 21 26 28 47 49 51 55 70
70
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS


TAMANHO: 20
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49 28 14 18 20 44
9
VETOR APÓS HEAPSORT: 0 1 4 8 9 11 12 14 18 20 21 26 28 44 47 49 51 55
70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 3 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 9 11 12 14 18 20 21 26 28 44 47 49 51 55
70 70
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS


TAMANHO: 22
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49 28 14 18 20 44
9 67 20
VETOR APÓS HEAPSORT: 0 1 4 8 9 11 12 14 18 20 20 21 26 28 44 47 49 51
55 67 70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 4 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 9 11 12 14 18 20 20 21 26 28 44 47 49 51
55 67 70 70
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS


TAMANHO: 24
VETOR ALEATÓRIO: 51 26 55 4 12 1 21 11 0 8 70 47 70 49 28 14 18 20 44
9 67 20 37 51
VETOR APÓS HEAPSORT: 0 1 4 8 9 11 12 14 18 20 20 21 26 28 37 44 47 49
51 51 55 67 70 70
TEMPO DE EXECUÇÃO (HEAPSORT): 6 MICROSSEGUNDOS
VETOR APÓS SHELLSORT: 0 1 4 8 9 11 12 14 18 20 20 21 26 28 37 44 47 49
51 51 55 67 70 70
TEMPO DE EXECUÇÃO (SHELLSORT): 2 MICROSSEGUNDOS

Script done on Fri May 19 12:11:19 2023
```

## 4. Análise das saídas

Para todos os tamanhos testados, tanto o Heapsort quanto o Shellsort apresentaram resultados corretos, ou seja, os vetores foram ordenados corretamente.

No caso do Heapsort, podemos observar que o tempo de execução se manteve em torno de 2 a 4 microssegundos para todos os tamanhos de vetores. Isso indica que o Heapsort possui uma eficiência bastante consistente e independente do tamanho do vetor. O que era esperado visto que o Heapsort tem sempre complexidade de tempo constante.

Por outro lado, o Shellsort apresentou tempos de execução variados, mas em geral mais baixos do que o Heapsort. Para a maioria dos tamanhos de vetor testados, seu tempo de execução ficou em torno de 1 a 2 microssegundos.

Em resumo, tanto o Heapsort quanto a implementação do Shellsort feita se mostraram algoritmos eficientes de ordenação, mas o Shellsort apresentou um desempenho ligeiramente melhor em termos de tempo de execução nos casos específicos testados.

No entanto, é importante ressaltar que o desempenho de ambos algoritmos pode variar dependendo do conjunto de dados e do tamanho do vetor. É sempre recomendável realizar testes em diferentes cenários para avaliar a eficiência dos algoritmos de ordenação em cada caso, visto que a complexidade do Shellsort não é bem definida e pode ser maior que a do Heapsort dependendo do caso.