Aula Prática 7 - HeapSort e ShellSort
Nome: Bernardo Dutra Lemos
Matrícula: 2022043949

1.Planejar uma bateria de testes com diferentes valores de tamanho para os vetores a serem gerados pelo programa, assim como uma forma de analisar a performance do método de maneira significativa para o conjunto de valores de h's escolhidos.

Os testes serão realizados em vetores de tamanho 10^5, 10^6 e 10^7 para cada tamanho serão realizados 5 testes com vetores gerados de forma aleatória. Em cada um dos tamanhos testados os tempos de clock serão salvos e será calculada a média aritmética desses valores e comparados com a média dos tempos da ordenação desses mesmos vetores com o algoritmo **heapsort.**

O valor de H escolhido será de valor inicial sendo o maior múltiplo de 2 que é maior que o tamanho e atualizado a uma taxa de divisão por 2, 2.5 e 3. Pois acredito que quanto mais vezes eu rodar o insertion sort para subvetores, mais ordenado o vetor estará quando H=1.

2.Comparação entre os tempos de ordenação do shellsort e do heapsort para tamanhos de 10^5, 10^6 e 10^7.

A forma que os valores foram calculados foi através da média aritmética do tempo ordenação de 10 vetores aleatórios de tamanho N.

- **N = 10^5:**
  - H /= 2

    ```
    Heap: 0.0251103 segundos
    Shell: 0.0329939 segundos
    ```
  - H /= 2.5

    ```
    Heap: 0.0254035 segundos
    Shell: 0.0232675 segundos
    ```
  - H /= 3

    ```
    Heap: 0.0264221 segundos
    Shell: 0.0274311 segundos
    ```
- **N = 10^6:**
  - H /= 2

    ```
    Heap: 0.355322 segundos
    Shell: 0.459824 segundos
    ```
  - H /= 2.5

    ```
    Heap: 0.356782 segundos
    Shell: 0.310395 segundos
    ```

- H /= 3

```
Heap: 0.371053 segundos
Shell: 0.393802 segundos
```

- **N = 10^7:**
  - H /= 2

```
Heap: 5.66386 segundos
Shell: 6.8794 segundos
```

  - H /= 2.5

```
Heap: 5.39613 segundos
Shell: 3.70414 segundos
```

  - H /= 3

```
Heap: 5.27581 segundos
Shell: 6.07849 segundos
```

De acordo com os testes realizados, dos testados o melhor H encontrado é o com valor de 2.5, onde é melhor até mesmo que o heapsort pros casos testados. Um fato interessante é que foi preciso implementar um if na hora de atualizar o H, pois algumas vezes o loop saia antes de realizar o insertion sort para H = 1, nesses casos o vetor não ficava ordenado.

3.Conclusões finais:

Quando a taxa de atualização é do H ser dividido por 2.5, temos que é a melhor escolha e esta é até melhor que o heapsort na maioria dos casos. É um tanto quanto surpreendente que para esse valor de h é consideravelmente melhor que o heapsort, que é considerado um dos melhores métodos de ordenação já feitos. Mas acredito que seja por que o método de implementação do heapsort não é o mais otimizado e usa recursão, enquanto o shellsort usa um algoritmo iterativo.