

# INF01124 - Classificação e Pesquisa de Dados - Laboratório 1

Importante: Esta tarefa pode ser feita em duplas!

## 1 ShellSort

Este laboratório tem por objetivo implementar o algoritmo de ShellSort e testá-lo com diferentes sequências. O primeiro passo é implementar o algoritmo de ShellSort, de forma que ele possa facilmente trocar a sequência utilizada. Neste laboratório iremos testar as três sequências definidas abaixo:

1. SHELL: potências de 2:

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, ...

2. KNUTH:

1, 4, 13, 40, 121, 364, 1093, 3280, 9841, 29524, 88573, 265720, 797161, 2391484, ...

3. CIURA: A sequência proposta por Ciura [1] (<https://oeis.org/A102549>), que é composta dos seguintes números <sup>1</sup>:

1, 4, 10, 23, 57, 132, 301, 701, 1577, 3548, 7983, 17961, 40412, 90927, 204585, 460316, 1035711, ...

Uma vez implementado o algoritmo vamos testá-lo com diferentes vetores de entrada, usando as diferentes sequências. Para isto, o programa de vocês deve ler os vetores de um arquivo da entrada padrão, e gravar resultados na saída padrão<sup>2</sup>. A entrada padrão deve ser redirecionado para um arquivo disponibilizado chamado **entrada1.txt**, contendo vários vetores a serem ordenados. Cada vetor é descrito em 1 linha. O primeiro número da linha descreve o tamanho  $n$  do vetor, seguido por  $n$  números do vetor.

Um exemplo que ilustra o formato do arquivo de entrada segue:

```
16 16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5
16 3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7
```

A saída gerada pelo programa de vocês vai imprimir o vetor parcialmente ordenado após cada etapa do algoritmo de ShellSort, correspondendo a um incremento. O formato da saída deve seguir um formato exato. Para a entrada acima, a saída a ser gerada é:

```
16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5 SEQ=SHELL
15 13 11 1 7 3 9 5 16 14 12 2 8 4 10 6 INCR=8
7 3 9 1 8 4 10 2 15 13 11 5 16 14 12 6 INCR=4
7 1 8 2 9 3 10 4 11 5 12 6 15 13 16 14 INCR=2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1
16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5 SEQ=KNUTH
3 10 5 1 8 4 9 6 15 13 11 2 7 16 14 12 INCR=13
3 4 5 1 7 10 9 2 8 13 11 6 15 16 14 12 INCR=4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1
16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5 SEQ=CIURA
```

<sup>1</sup>Para números maiores que 701, os termos dessa sequência são gerados pela formula  $h_k = \lfloor 2.25h_{k-1} \rfloor$ .

<sup>2</sup><https://tinyurl.com/2btt5dyx>

```

11 2 7 1 8 4 9 6 15 13 16 14 12 3 10 5 INCR=10
8 2 7 1 11 3 9 5 12 4 10 6 15 13 16 14 INCR=4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1
3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7 SEQ=SHELL
3 9 5 15 13 11 1 7 4 10 6 16 14 12 2 8 INCR=8
3 9 1 7 4 10 2 8 13 11 5 15 14 12 6 16 INCR=4
1 7 2 8 3 9 4 10 5 11 6 12 13 15 14 16 INCR=2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1
3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7 SEQ=KNUTH
3 2 5 16 14 12 1 8 4 9 6 15 13 11 10 7 INCR=13
3 2 1 7 4 9 5 8 13 11 6 15 14 12 10 16 INCR=4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1
3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7 SEQ=CIURA
3 10 5 11 2 7 1 8 4 9 6 15 13 16 14 12 INCR=10
2 7 1 8 3 9 5 11 4 10 6 12 13 16 14 15 INCR=4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 INCR=1

```

A saída do programa deve mostrar o estado da sequência após cada iteração, exatamente como mostrado acima, imprimindo o conteúdo do vetor e o incremento usado. Os resultados devem ser colocados em um arquivo chamado **saida1.txt**.

## 2 Testes de escala

Cronometre o tempo de execução em vetores contendo números aleatórios de tamanho 100, 1000, 10000, 100000 e 1000000 para todas as três sequências da questão anterior. Estes vetores serão informados no arquivo **entrada2.txt** seguindo o mesmo formato da questão anterior.

Execute o seu código nesse algoritmo e gere a seguinte saída:

```

SHELL,100,0.011000,3.4 GHz Quad-Core Intel Core i7
KNUTH,100,0.008000,3.4 GHz Quad-Core Intel Core i7
CIURA,100,0.009000,3.4 GHz Quad-Core Intel Core i7
SHELL,1000,0.200000,3.4 GHz Quad-Core Intel Core i7
KNUTH,1000,0.130000,3.4 GHz Quad-Core Intel Core i7
CIURA,1000,0.135000,3.4 GHz Quad-Core Intel Core i7
SHELL,10000,4.148000,3.4 GHz Quad-Core Intel Core i7
KNUTH,10000,2.025000,3.4 GHz Quad-Core Intel Core i7
CIURA,10000,2.032000,3.4 GHz Quad-Core Intel Core i7
SHELL,100000,93.591000,3.4 GHz Quad-Core Intel Core i7
KNUTH,100000,23.972000,3.4 GHz Quad-Core Intel Core i7
CIURA,100000,22.418000,3.4 GHz Quad-Core Intel Core i7
SHELL,1000000,2433.681000,3.4 GHz Quad-Core Intel Core i7
KNUTH,1000000,354.408000,3.4 GHz Quad-Core Intel Core i7
CIURA,1000000,279.913000,3.4 GHz Quad-Core Intel Core i7

```

Em cada linha coloca-se o nome da sequência usada, seguido do tamanho de vetor de entrada e tempo em **milisegundos** para ordenar o vetor usando o algoritmo de ShellSort e a sequência respectiva, e ao final uma descrição do processador usado para rodar os testes. Os resultados devem ser colocados em um arquivo chamado **saida2.txt**.

## 3 Entrega

A solução deve ser enviada pelo Moodle dentro de um arquivo .zip, contendo os seguintes arquivos:

- **integrantes.txt**: coloque o nome dos integrantes do grupo (até 2 pessoas) , com um nome por linha
- **saida1.txt**: resultado do exercício 1, no exato formato listado acima
- **saida2.txt**: resultado do exercício 2, no exato formato listado acima
- código fonte correspondente a solução

## 4 Desafio Bônus - DNA Sorting

Resolva o problema UVA 612 - DNA Sorting

Este é um problema de maratona de programação. O problema será considerado completo se for aceito no site da <https://uva.onlinejudge.org> e o código for entregue com o resto dos exercícios. O entregue correto valerá um adicional de 25% pontos.

## Referências

- [1] Ciura, Marcin (2001). "Best Increments for the Average Case of Shellsort". In Freiwalds, Rusins. Proceedings of the 13th International Symposium on Fundamentals of Computation Theory. London: Springer-Verlag. pp. 106–117. ISBN 978-3-540-42487-1.