

# Ordenação: Shellsort

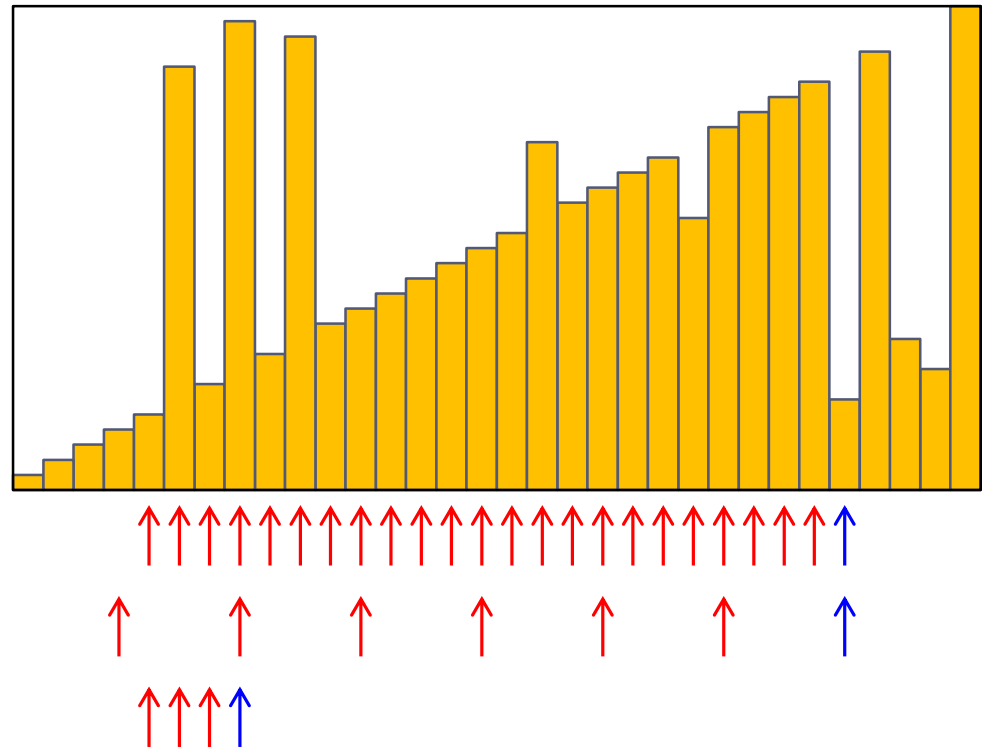
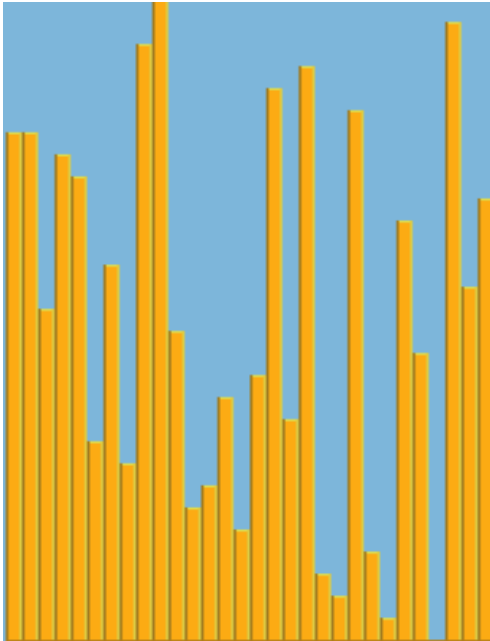
Algoritmos e Estruturas de Dados II

# Introdução

---

- ▶ Proposto por Donald Shell em 1959.
- ▶ Extensão do algoritmo de ordenação inserção.

# Motivação



```
while (( j >= 0 ) && (aux.Chave < v[j].Chave)) {  
    v[j + 1] = v[j];  
    j--;  
}
```

11 comp.  
24 comp.

# Motivação

---

- ▶ Problema do método de inserção
  - ▶ Troca itens adjacentes para determinar o ponto de inserção.
  - ▶ São efetuadas  $n - 1$  comparações e movimentações quando o menor item está na posição mais à direita no vetor.
- ▶ O método de Shell contorna este problema permitindo trocas de registros distantes um do outro.

# Algoritmo

---

- ▶ Os itens separados de  $h$  posições são rearranjados.
- ▶ Todo  $h$ -ésimo item leva a uma sequência ordenada.
- ▶ Tal sequência é dita estar  $h$ -ordenada.
  
- ▶ Quando  $h=1$ , o algoritmo é equivalente ao algoritmo de inserção

# Exemplo

---

- ▶ Entrada: O R D E N A
- ▶  $h = 4, 2, 1$

	1	2	3	4	5	6
Chaves iniciais:	<i>O</i>	<i>R</i>	<i>D</i>	<i>E</i>	<i>N</i>	<i>A</i>
$h = 4$	<i>N</i>	<i>A</i>	<i>D</i>	<i>E</i>	<i>O</i>	<i>R</i>
$h = 2$	<i>D</i>	<i>A</i>	<i>N</i>	<i>E</i>	<i>O</i>	<i>R</i>
$h = 1$	<i>A</i>	<i>D</i>	<i>E</i>	<i>N</i>	<i>O</i>	<i>R</i>

# Escolha do h

---

- ▶ Sequência para h:

$$h(s) = 1, \text{ para } s = 1$$

$$h(s) = 3h(s-1) + 1 \text{ para } s > 1$$

- ▶ A sequência corresponde a 1, 4, 13, 40, 121, 364, 1093, 3280, ...
- ▶ Knuth (1973, p. 95) mostrou experimentalmente que esta sequência é difícil de ser batida por mais de 20% em eficiência.

# ShellSort

---

```
void Shellsort (Item* A, int n) {
    int i, j;
    int h = 1;
    Item aux;

    do /* calcula o h inicial */
        h = h * 3 + 1;
    while (h < n);

    do {
        h = (h-1/3); /* atualiza o valor de h */
        for(i = h; i < n; i++) {
            aux = A[i]; j = i;

            /* efetua comparações entre elementos com distância h */
            while (A[j - h].Chave > aux.Chave) {
                A[j] = A[j - h]; j -= h;
                if (j < h) break;
            }
            A[j] = aux;
        }
    } while (h != 1);
}
```

---





# ShellSort

---

## ■ **Análise**

- ❑ A razão da eficiência do algoritmo ainda não é conhecida.
- ❑ Ninguém ainda foi capaz de analisar o algoritmo.
- ❑ A sua análise contém alguns problemas matemáticos muito difíceis.
- ❑ A começar pela própria sequência de incrementos.
- ❑ O que se sabe é que cada incremento não deve ser múltiplo do anterior.



# ShellSort

---

## ■ **Análise**

- Conjecturas referente ao número de comparações para a seqüência de Knuth:

Conjectura 1 :  $C(n) = O(n^{1,25})$

Conjectura 2 :  $C(n) = O(n (\ln n)^2)$



# ShellSort

---

## ■ Vantagens:

- ❑ Shellsort é uma ótima opção para arquivos de tamanho moderado.
- ❑ Sua implementação é simples e requer uma quantidade de código pequena.

## ■ Desvantagens:

- ❑ O tempo de execução do algoritmo é sensível à ordem inicial do arquivo.
- ❑ **O método não é estável.**

