Classificação e Pesquisa de Dados

Cristiano Santos

cristiano.santos@amf.edu.br

ShellSort

Algoritmo ShellSort

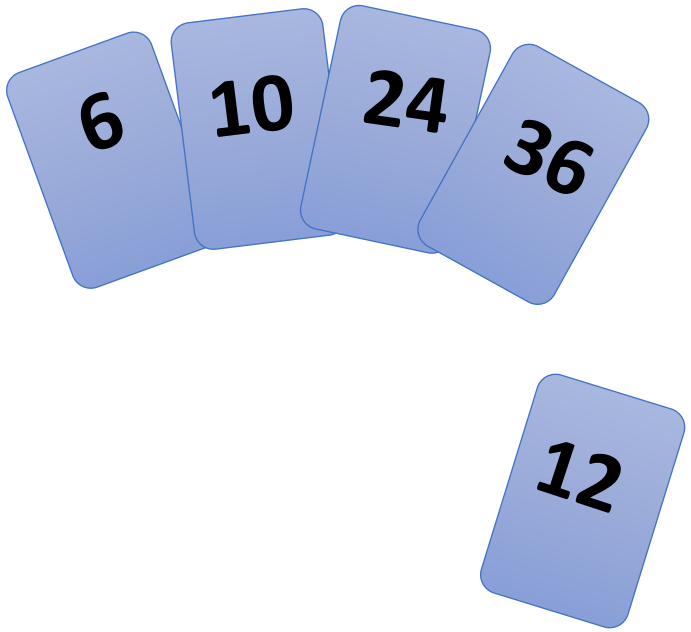
- Proposto por Donald Shell em 1959.
- **É uma extensão do algoritmo de ordenação por inserção**.
- 1º algoritmo a quebrar a barreira do tempo quadrático

Algoritmo ShellSort

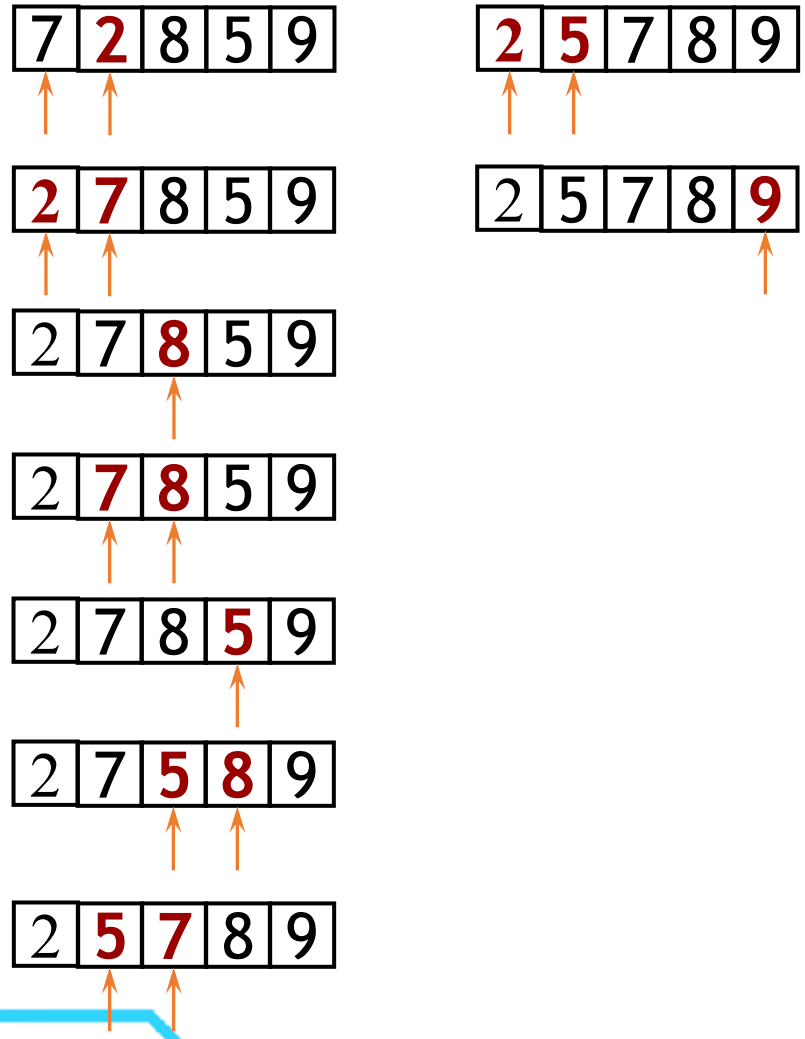
- **Problema com o algoritmo de ordenação por inserção**:
  - **Troca itens adjacentes** para determinar o ponto de inserção.
  - **São efetuadas n - 1 comparações** e movimentações quando o menor item está na **posição mais à direita no vetor.**

Relembrando (Insertion Sort)

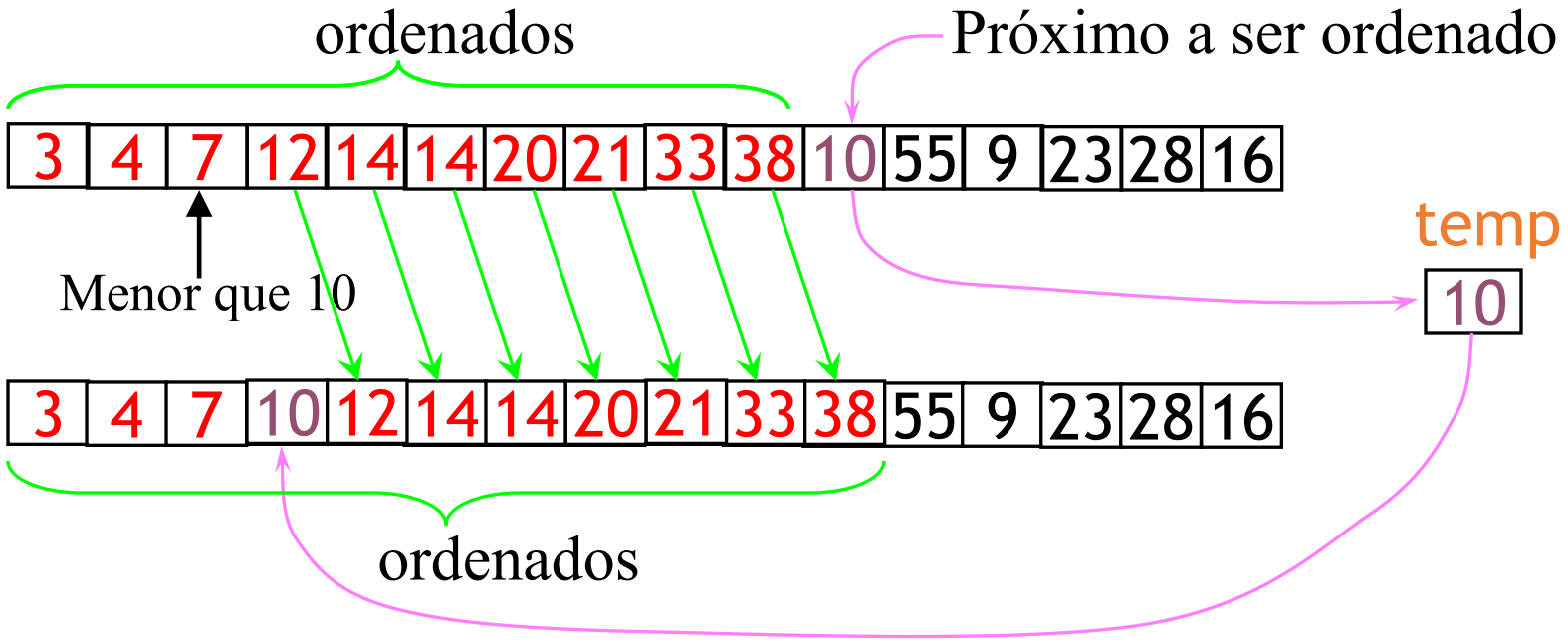
- Método do jogador de cartas



Relembrando (Insertion Sort)



Relembrando (Insertion Sort)



Algoritmo ShellSort

- O método de **Shell contorna este problema** permitindo **trocas de registros distant**es um do outro.

- **Shellsort usa uma sequência** h1, h2, …, h_k chamada **sequência de incremento**.

  - **Qualquer sequência de incremento é boa**, desde que **h1 = 1** e algumas outras escolhas sejam melhores que outras.

Algoritmo ShellSort

- A **distância entre as comparações diminui à medida que o algoritmo de classificação é executado** até a **última fase** na qual os **elementos adjacentes** são comparados

Algoritmo ShellSort

- **Após cada fase e algum incremento h_k**, para cada i, temos **a[i] ≤ a [i + h_k]** todos os elementos **espaçados h_k separados são classificados.**

- **Diz-se que o arquivo é hk – classificado.**

ShellSort: Exemplo

Sort: 18 32 12 5 38 33 16 2

8 números para ser ordenado, O aumento da Shell será o arredondamento(floor) de (n/2)

* floor**(8/2)** ➔ floor**(4) = 4**

incremento 4: **1** **2** **3** **4**

18 **32** **12** **5** **38** **33** **16** **2**

Passo **1**) Observe apenas **18** e **38** e classifique em ordem;
18 e **38** permanecem em sua posição atual porque estão em ordem.

Step **2**) Observe apenas **32** e **33** e classifique em ordem;
32 e **33** permanecem na posição atual porque estão em ordem.

ShellSort: Exemplo

Sort: 18 32 12 5 38 33 16 2

8 Números a serem classificados, o incremento da Shell será floor(n/2)

*** floor(8/2) ➔ floor(4) = 4**

incremento 4: **1** **2** **3** **4**

18 **32** **12** **5** **38** **33** **16** **2**

Passo 3) Observe apenas 12 e 16 e classifique em ordem;
12 e 16 permanecem em sua posição atual porque estão em ordem.

Passo 4) Olhe apenas para 5 e 2 e classifique em ordem;
2 e 5 precisam ser trocados para ficarem em ordem.

ShellSort: Exemplo

Sort: 18 32 12 5 38 33 16 2

Números resultantes após a passagem do incremento 4 :

18 32 12 *2* 38 33 16 *5*

*** floor(4/2) ➔ floor(2) = 2**

incremento 2: **1** **2**

18 32 12 2 38 33 16 5

Passo **1**) Olha **18**, **12**, **38**, **16** e classifica em seus locais apropriados:

12 38 16 2 18 33 38 5

Step **2**) Olha para **32**, **2**, **33**, **5** e classifica em seus locais apropriados :

12 2 16 5 18 32 38 33

ShellSort: Exemplo

Sort: 18 32 12 5 38 33 16 2

*** floor(2/2) ➔ floor(1) = 1**

incremento 1: 1

12	2	16	5	18	32	38	33
2	**5**	**12**	**16**	**18**	**32**	**33**	**38**

O último incremento ou fase do Shellsort é basicamente um algoritmo de classificação por inserção.

ShellSort: Vantagem

- A vantagem do Shellsort é que ele é **eficiente apenas para listas de tamanho médio**.
- **Para listas maiores**, o **algoritmo não é a melhor escolha**. O mais rápido de todos os algoritmos de classificação O(N^2).
- **5 vezes mais rápido que o bubble sort** e um pouco mais de **duas vezes mais rápido que o insertion sort,** seu concorrente mais próximo.

ShellSort - Desvantagem

- Não é **nem de longe tão eficiente quanto o merge, heap e quick sorts.**

- O shellsort ainda **é significativamente mais lento que o merge, heap e quick sorts,** mas seu algoritmo relativamente simples o torna uma boa escolha para **classificar listas de menos de 5000 itens**, a menos que a velocidade seja importante.

- Nota: É uma excelente escolha para classificação repetitiva de listas menores.

ShellSort – MELHOR caso

- O melhor caso na classificação de shell é quando o **array já está classificado na ordem correta.** O número de comparações é menor.

ShellSort – PIOR caso

- O tempo de execução do Shellsort **depende da escolha da sequência de incrementos.**

Resumo

- 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,**

Implementação: Ideia

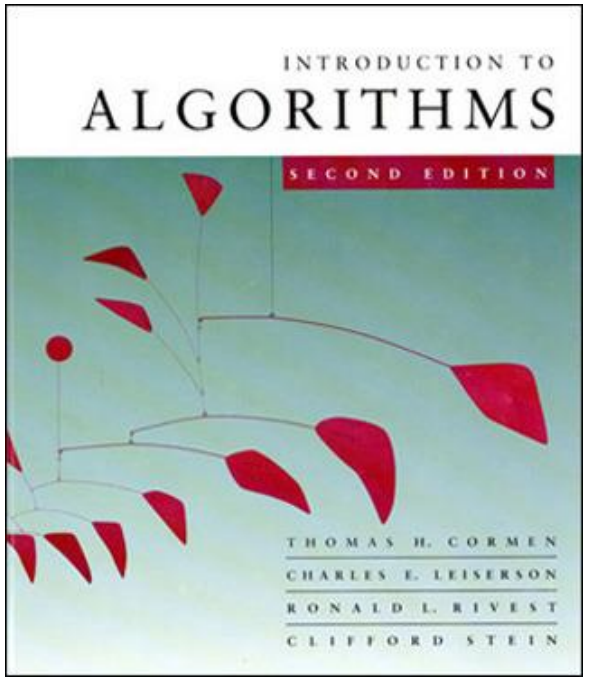
```
def shellSort(alist):
    sublistcount = len(alist)//2
    while sublistcount > 0:

      for startposition in range(sublistcount):
        gapInsertionSort(alist,startposition,sublistcount)

      print("After increments of size",sublistcount,
                                   "The list is",alist)

      sublistcount = sublistcount // 2
```

Implementação: Ideia

```
def gapInsertionSort(alist,start,gap):
    for i in range(start+gap,len(alist),gap):

        currentvalue = alist[i]
        position = i

        while position>=gap and alist[position-gap]>currentvalue:
            alist[position]=alist[position-gap]
            position = position-gap

        alist[position]=currentvalue
```

Leitura importante

livro “Algorithms” de Cormen et al.



Classificação e Pesquisa de Dados

Cristiano Santos

cristiano.santos@amf.edu.br