UniCesumar

Estrutura de Dados II

Prof. Me. Pietro M. de Oliveira

Shellsort

Shellsort

Extensão da ordenação por inserção (otimização)

Criador: Donald Shell (1959)

Instável

GAP: otimização que "aproxima" elementos mais distantes

- Maior proveito do melhor caso do Insertionsort

Desempenho:

- Pior caso - n^2
- Melhor caso - **não alcança**, mas chega **próximo** a $n \log n$

Shellsort

Subdivide o arranjo, e ordena as divisões, sucessivamente

GAP é um índice que controla os "pulos"

- Vetor ordenado de ***k*** em ***k*** elementos (k = GAP)
- Índices são múltiplos do GAP
- O GAP varia:
  - Geralmente acaba com o valor GAP = 1
  - Ex. de sequência de GAPs: [23,13,5,1]
  - Valores de GAP armazenados em vetor

Shellsort

Algoritmo

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]*** e ***i = k+1***
3. Enquanto ***i <*** *comprimento*[***A***] faça
4. ***chave*** ← ***A[i]***
5. ***j*** ← ***i***
6. Enquanto ***j ≥ k*** e ***A[j-k] > chave*** faça
7. ***A[j]*** ← ***A[j-k]***
8. ***j*** ← ***j - k***
9. ***A[j]*** ← ***chave***
10. ***i = i + 1***

Shellsort

O **Shellsort** é uma otimização do **Insertionsort**

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]*** e ***i = k+1***
3. Enquanto ***i <*** *comprimento*[***A***] faça
4. ***chave*** ← ***A[i]***
5. ***j*** ← ***i***
6. Enquanto ***j ≥ k*** e ***A[j-k] > chave*** faça
7. ***A[j]*** ← ***A[j-k]***
8. ***j*** ← ***j - k***
9. ***A[j]*** ← ***chave***
10. ***i = i + 1***

Shellsort - Exemplo

GAPs

1	2	3
3	**2**	**1**

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

1	2	3	4	5	6	7	8
26	**32**	**46**	**19**	**15**	**67**	**81**	**22**

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	**2**	**1**

1	2	3	4	5	6	7	8
26	**32**	**46**	**19**	**15**	**67**	**81**	**22**

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
26	32	46	19	15	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	32	46	26	15	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	**2**	**1**

chave

1	2	3	4	5	6	7	8
19	**32**	**46**	**26**	**15**	**67**	**81**	**22**

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	32	46	26	15	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

Itera-se o índice *i* do Insertionsort (linha 3 do algoritmo)

1	2	3	4	5	6	7	8
19	32	46	26	15	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	**2**	**1**

chave

1	2	3	4	5	6	7	8
19	**32**	**46**	**26**	**15**	**67**	**81**	**22**

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	46	26	32	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
19	15	46	26	32	67	81	22

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

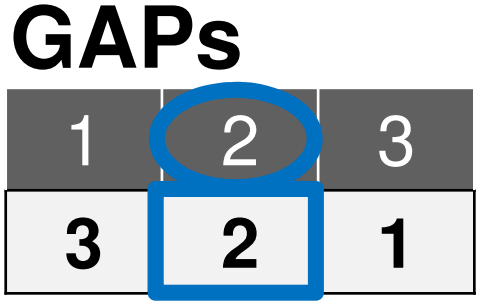
1	2	3
3	2	1

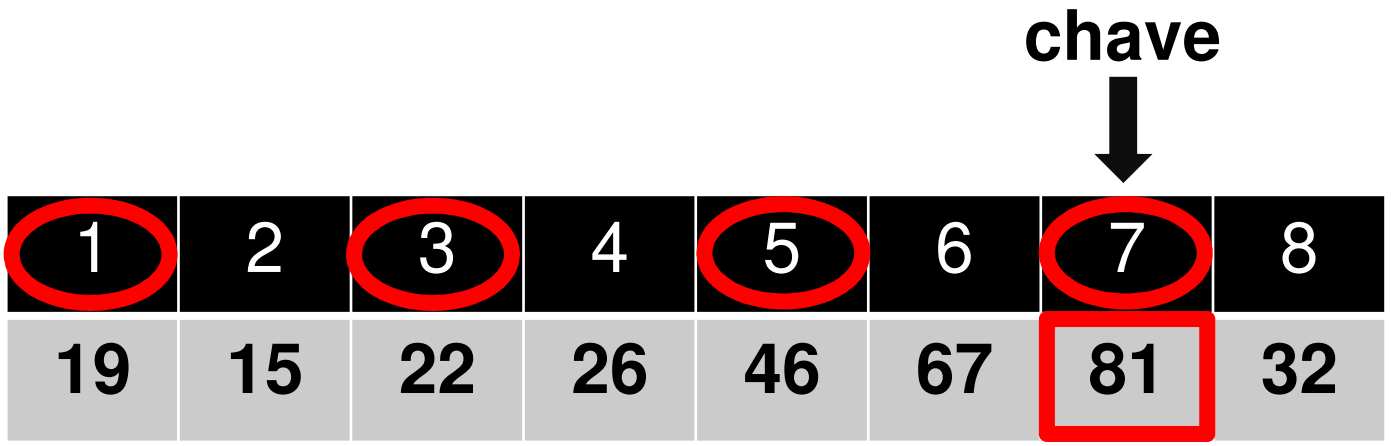
1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***





Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

Itera-se o índice *i* do Insertionsort (linha 3 do algoritmo)

1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

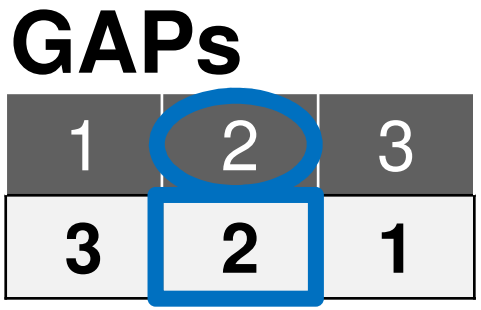
1	2	3
3	2	1

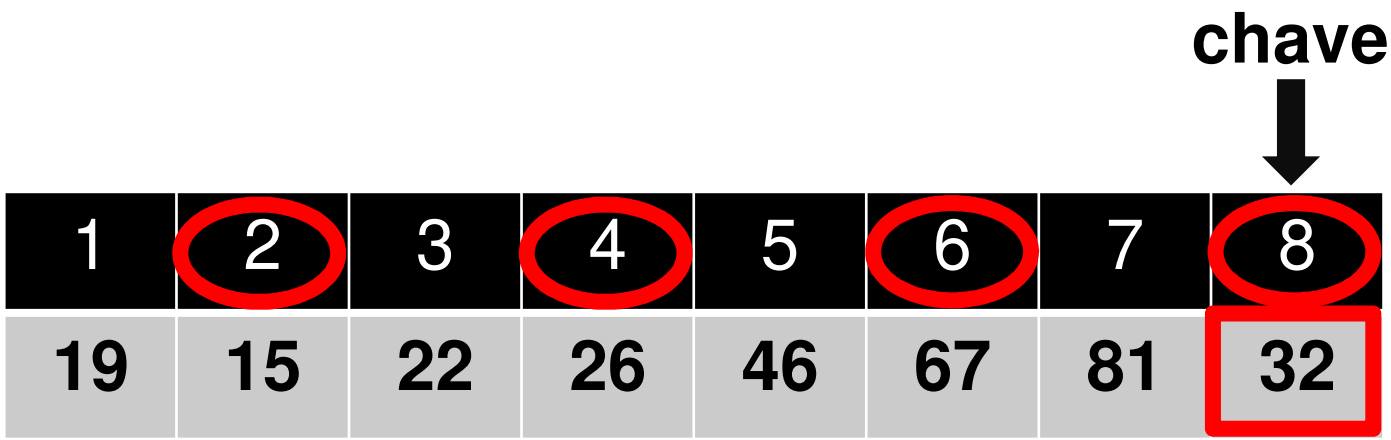
1	2	3	4	5	6	7	8
19	15	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***





Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	22	26	46	32	81	67

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
19	15	46	26	22	67	81	32

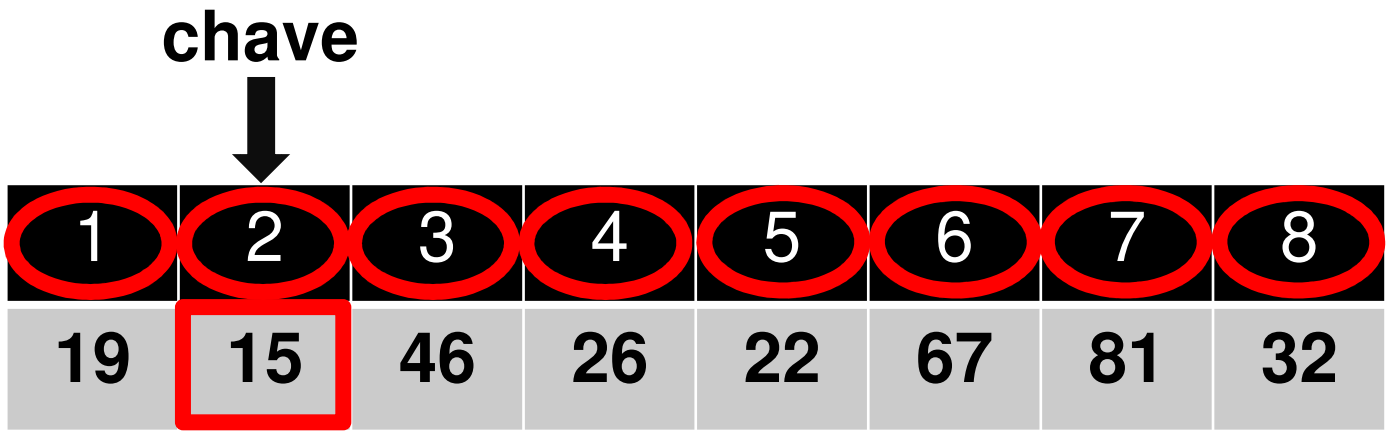
Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1



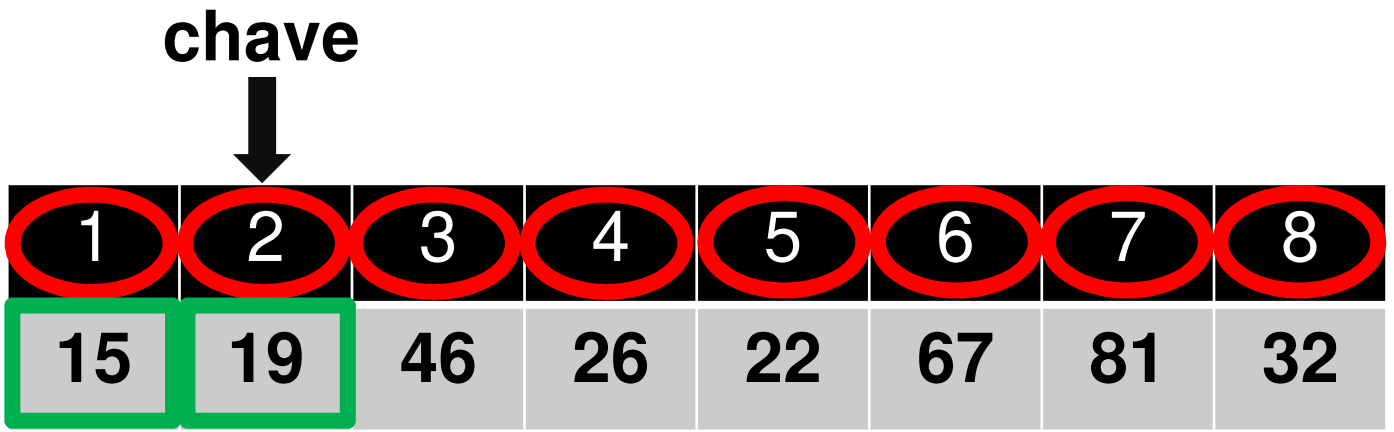
Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1



Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
15	19	46	26	22	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
15	19	46	26	22	67	81	32

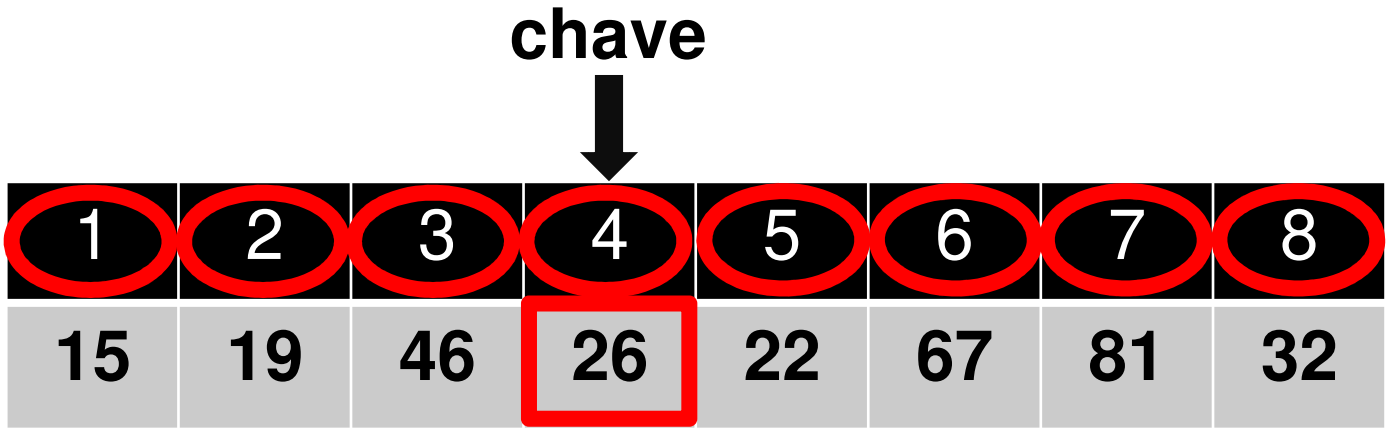
Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1



Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
15	19	26	46	22	67	81	32

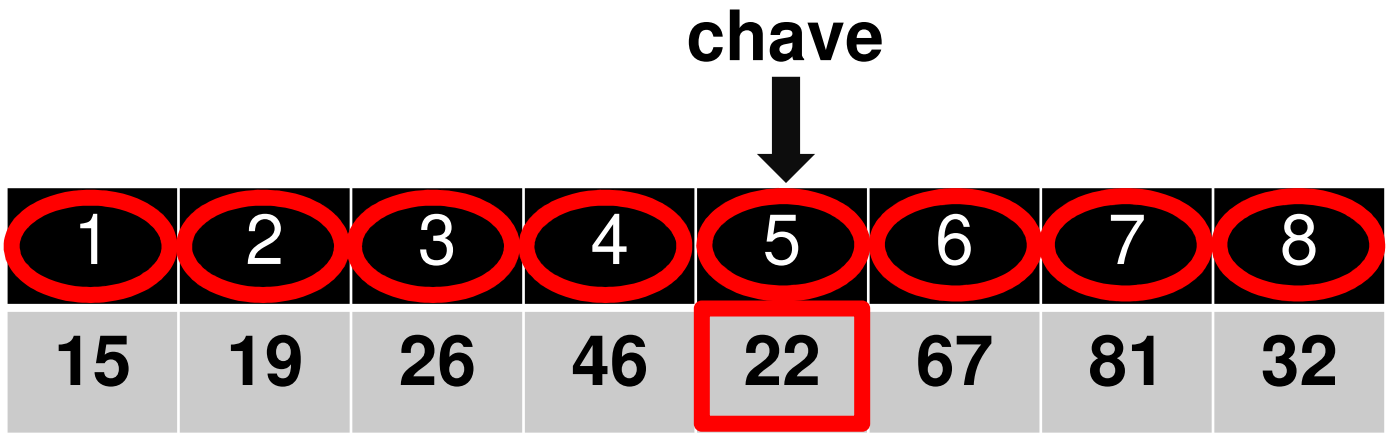
Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1



Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
15	19	22	26	46	67	81	32

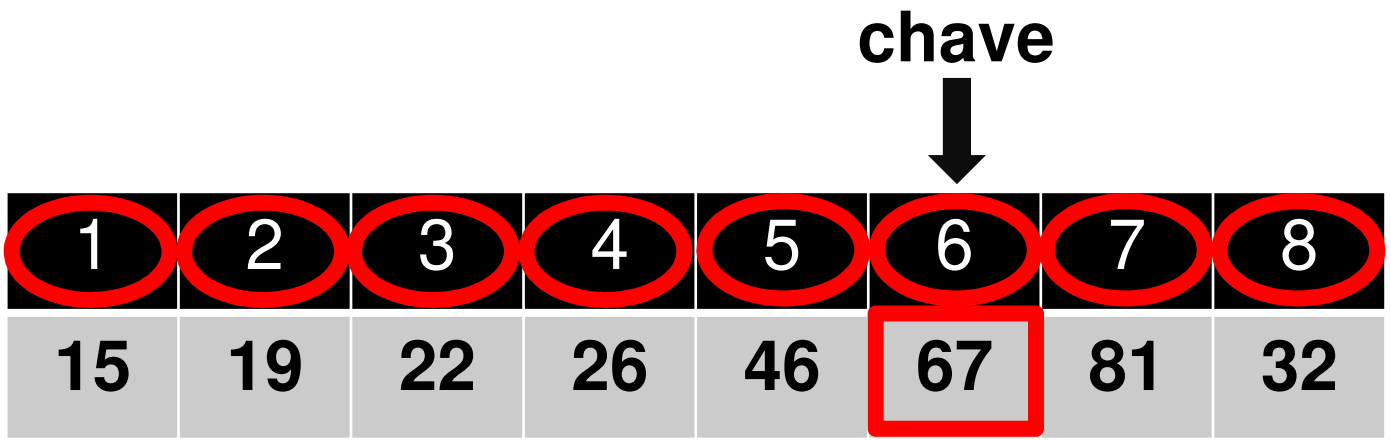
Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1



Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

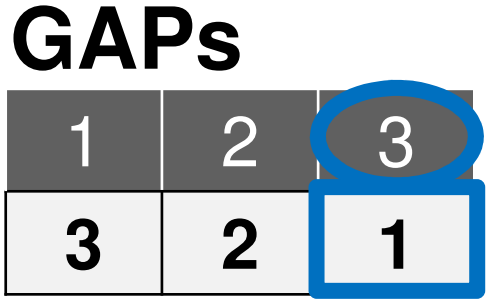
1	2	3
3	2	1

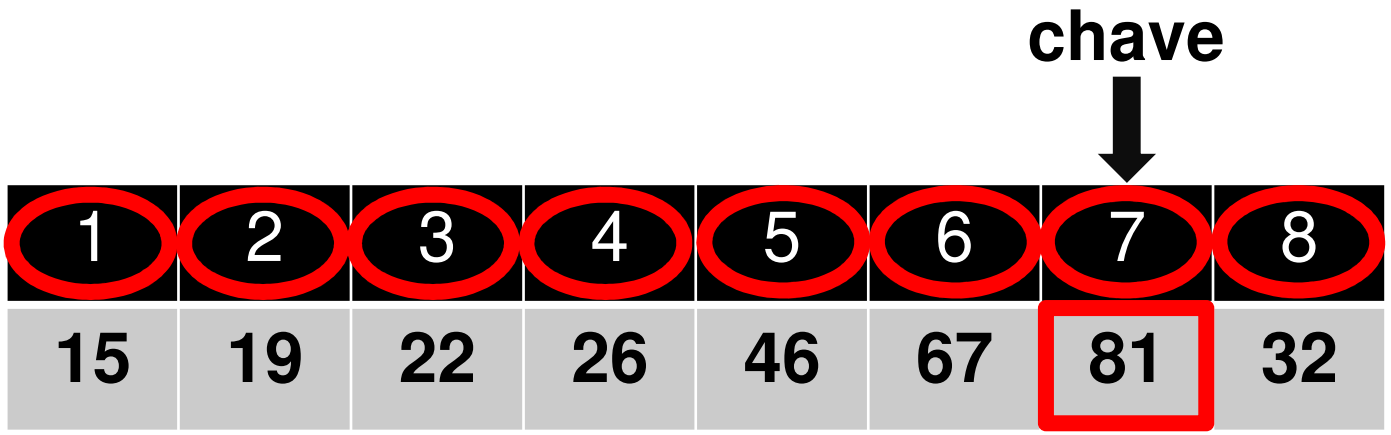
1	2	3	4	5	6	7	8
15	19	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***





Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
15	19	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

chave

1	2	3	4	5	6	7	8
15	19	22	26	46	67	81	32

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	2	1

1	2	3	4	5	6	7	8
15	19	22	26	32	46	67	81

Shellsort

Shellsort(arranjo ***A***)

1. Para ***g*** ← ***1*** até *comprimento*[***GAPs***] faça
2. ***k = GAPs[g]***
3. Insertionsort com o gap ***k***

GAPs

1	2	3
3	**2**	**1**

Arranjo ordenado!

1	2	3	4	5	6	7	8
15	**19**	**22**	**26**	**32**	**46**	**67**	**81**

UniCesumar

Estrutura de Dados II

Prof. Me. Pietro M. de Oliveira

Considerações Finais

Unidade III

Considerações Finais

As próximas comparações levam em consideração arranjos de tamanho ***n***. Ou seja, o vetor a ser ordenado possui ***n*** elementos!

- **Bubblesort** e **Selectionsort** realizam n^2 operações, independentemente da distribuição dos elementos no arranjo.

Considerações Finais

- **Insertionsort** realiza:
  - n^2 operações
    - Arranjo ordenado inversamente (pior caso)
    - Arranjo desordenado (caso médio)
  - n operações
    - Arranjo ordenado (melhor caso)
    - Arranjo quase ordenado (melhor caso)

- **Shellsort**:
  - Otimização do Insertionsort
  - GAP: “aproxima” elementos distantes
  - Se aproveita do melhor caso do Insertionsort

Considerações Finais

- **Bubblesort** × **Selectionsort** × **Insertionsort**
  - **Bubblesort** e **Selectionsort** são "ruins", independente do arranjo: dois laços de repetição aninhados;
  - **Insertionsort** é muito bom quando o arranjo está quase ordenado: laço de repetição interno é interrompido em casos especiais.
- **Insertionsort** × **Shellsort**
  - Shellsort "aproxima" elementos distantes;
  - Quase atinge ***n log n***, no melhor caso;
  - A cada mudança no GAP, o arranjo encontra-se quase ordenado (melhor caso do Insertionsort)

UniCesumar

Resumo

- Ordenação por flutuação (Bubblesort)
- Ordenação por seleção (Selectionsort)
- Ordenação por inserção (Insertionsort)
- Shellsort
- Comparações entre os métodos

UniCesumar

Estrutura de Dados II

Prof. Me. Pietro M. de Oliveira