

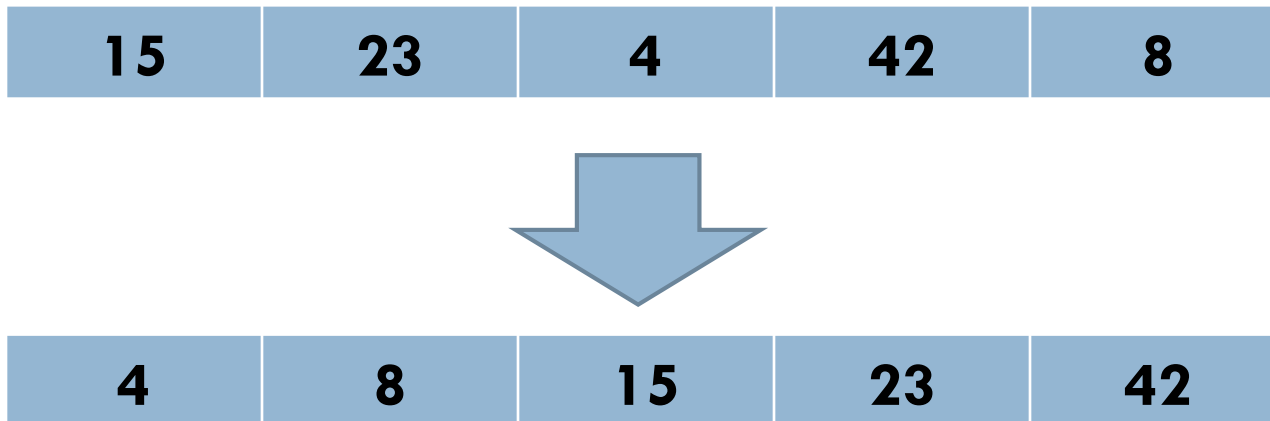
# ORDENAÇÃO

Prof. Muriel Mazzetto  
Estrutura de Dados

# Ordenação

2

- Organizar os elementos em uma ordem específica:
  - Em um conjunto de  $n$  chaves  $\{k_1, k_2, \dots, k_n\}$ , organizá-lo de forma que  $k_1 < k_2 < \dots < k_n$ .
  - Exemplo:



# Ordenação

3

- Organizar os elementos em uma ordem específica:
  - Em um determinado formato organizá-lo
  - Exemplos

Ordem alfabética ou  
ordem numérica.

Ex: ordenação de arquivos  
por tipo, nome, alteração...

# Ordenação

4

- Quando ordenar?
  - ▣ A complexidade da ordenação não deve exceder a complexidade da computação sem ordenação.
    - Se são executadas poucas buscas na estrutura.
    - Se existem poucos elementos na estrutura.

# Ordenação

5

- Métodos de ordenação mais conhecidos:
  - ▣ BubbleSort (por trocas);
  - ▣ InsertionSort (por inserção direta);
  - ▣ SelectionSort (seleção direta);
  - ▣ HeapSort (seleção em árvore);
  - ▣ MergeSort (por intercalação);
  - ▣ QuickSort (por trocas).

# Ordenação

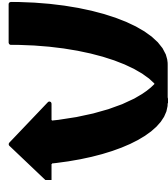
6

## □ Métodos de ordenação mais conhecidos:

- |                                        |   |                    |
|----------------------------------------|---|--------------------|
| □ BubbleSort (por trocas);             | } | Simple             |
| □ InsertionSort (por inserção direta); |   | Muitas comparações |
| □ SelectionSort (seleção direta);      |   | Poucas buscas      |
| □ HeapSort (seleção em árvore);        | } | Poucos elementos   |
| □ MergeSort (por intercalação);        |   | Complexos          |
| □ QuickSort (por trocas).              |   | Poucas comparações |
|                                        |   | Muitas buscas      |
|                                        |   | Vários elementos   |

# Ordenação

7

- **Estabilidade:** quando o método de ordenação mantém a ordem original entre elementos de mesma chave.
  - Ex:
    - 3[a], 2[b], 2[c], 1[d]
    - 1[d], 2[b], 2[c], 3[a]
- 
- APÓS ORDENAR

# Ordenação

8

- **Adaptabilidade:** métodos que têm o tempo de execução reduzido quando a entrada já está ordenada.
  - ▣ São métodos que se adaptam de acordo com a entrada para não realizar todas as operações.



# BubbleSort

9

- ❑ O algoritmo de ordenação mais simples existente.
- ❑ Usa comparações consecutivas, trocando elementos do vetor para ordená-lo.
- ❑ Percorre o vetor diversas vezes, fazendo os elementos “flutuarem” para sua posição.

# BubbleSort

10

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

# BubbleSort

11

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

4	5	2	1	3
---	---	---	---	---

# BubbleSort

12

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

4	5	2	1	3
---	---	---	---	---

4	2	5	1	3
---	---	---	---	---

# BubbleSort

13

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

4	5	2	1	3
---	---	---	---	---

4	2	5	1	3
---	---	---	---	---

4	2	1	5	3
---	---	---	---	---

# BubbleSort

14

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

4	5	2	1	3
---	---	---	---	---

4	2	5	1	3
---	---	---	---	---

4	2	1	5	3
---	---	---	---	---

4	2	1	3	5
---	---	---	---	---

# BubbleSort

15

□ Exemplo:

5	4	2	1	3
---	---	---	---	---

4	5	2	1	3
---	---	---	---	---

4	2	5	1	3
---	---	---	---	---

4	2	1	5	3
---	---	---	---	---

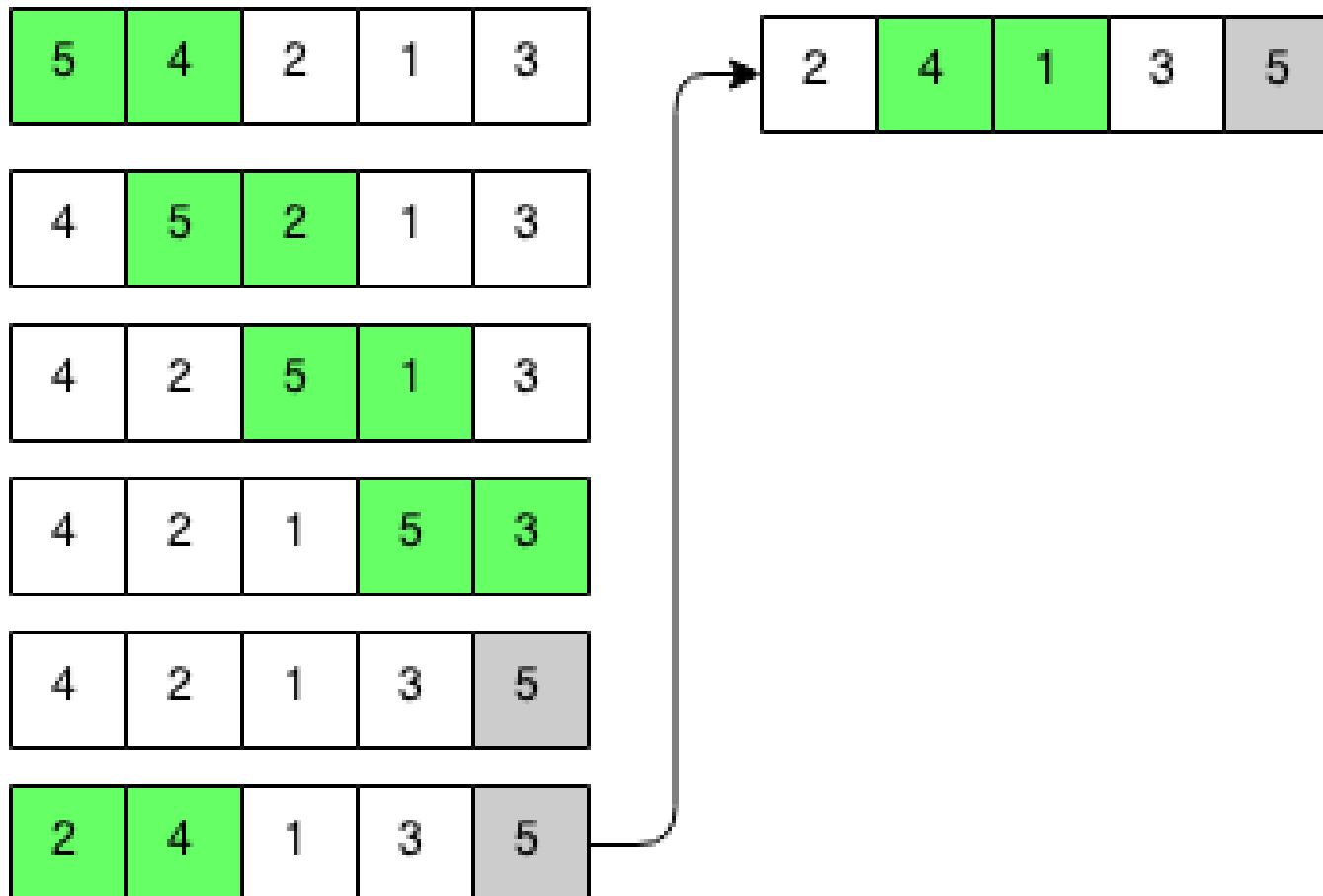
4	2	1	3	5
---	---	---	---	---

2	4	1	3	5
---	---	---	---	---

# BubbleSort

16

□ Exemplo:

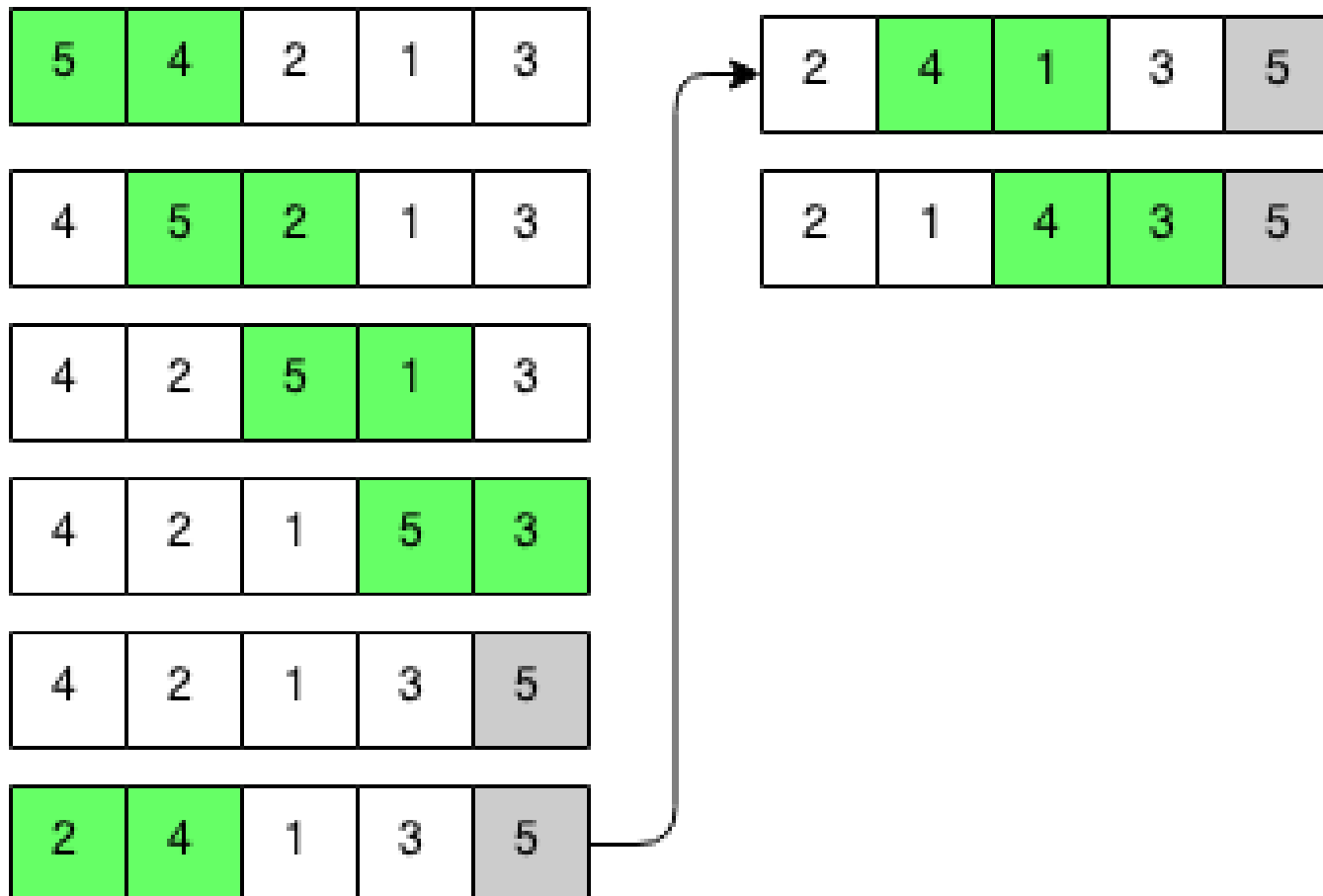




# BubbleSort

17

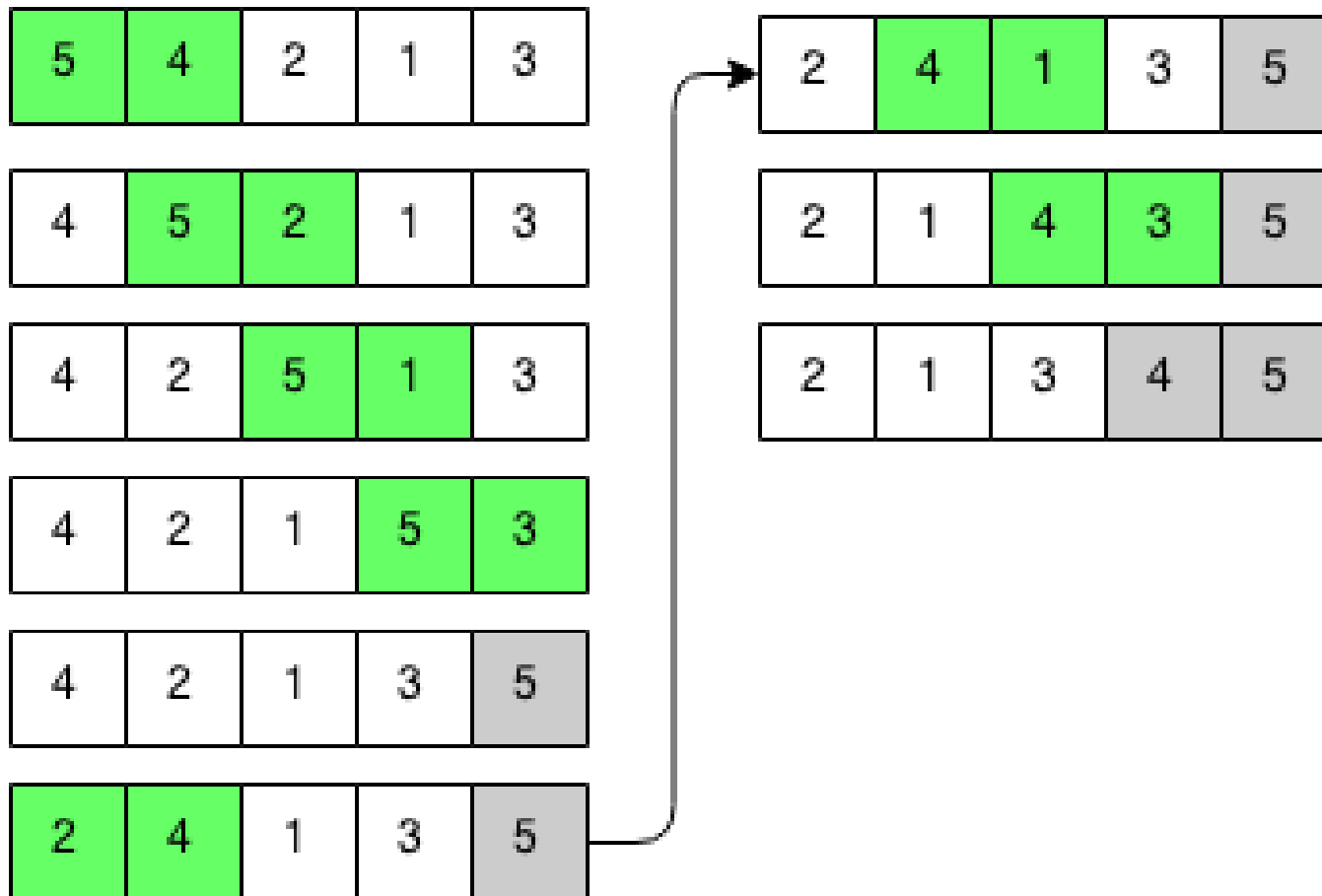
## □ Exemplo:



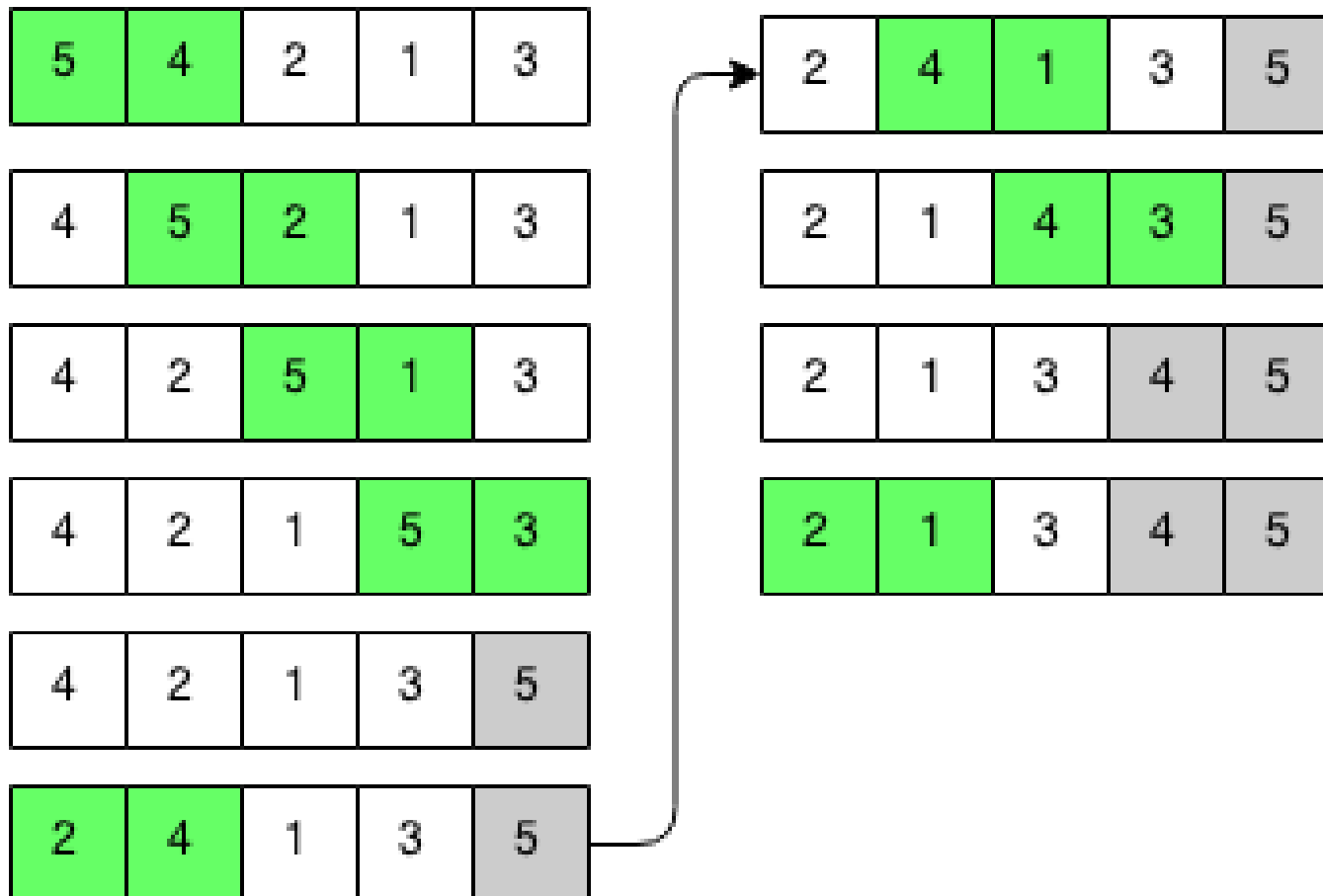
# BubbleSort

18

## □ Exemplo:



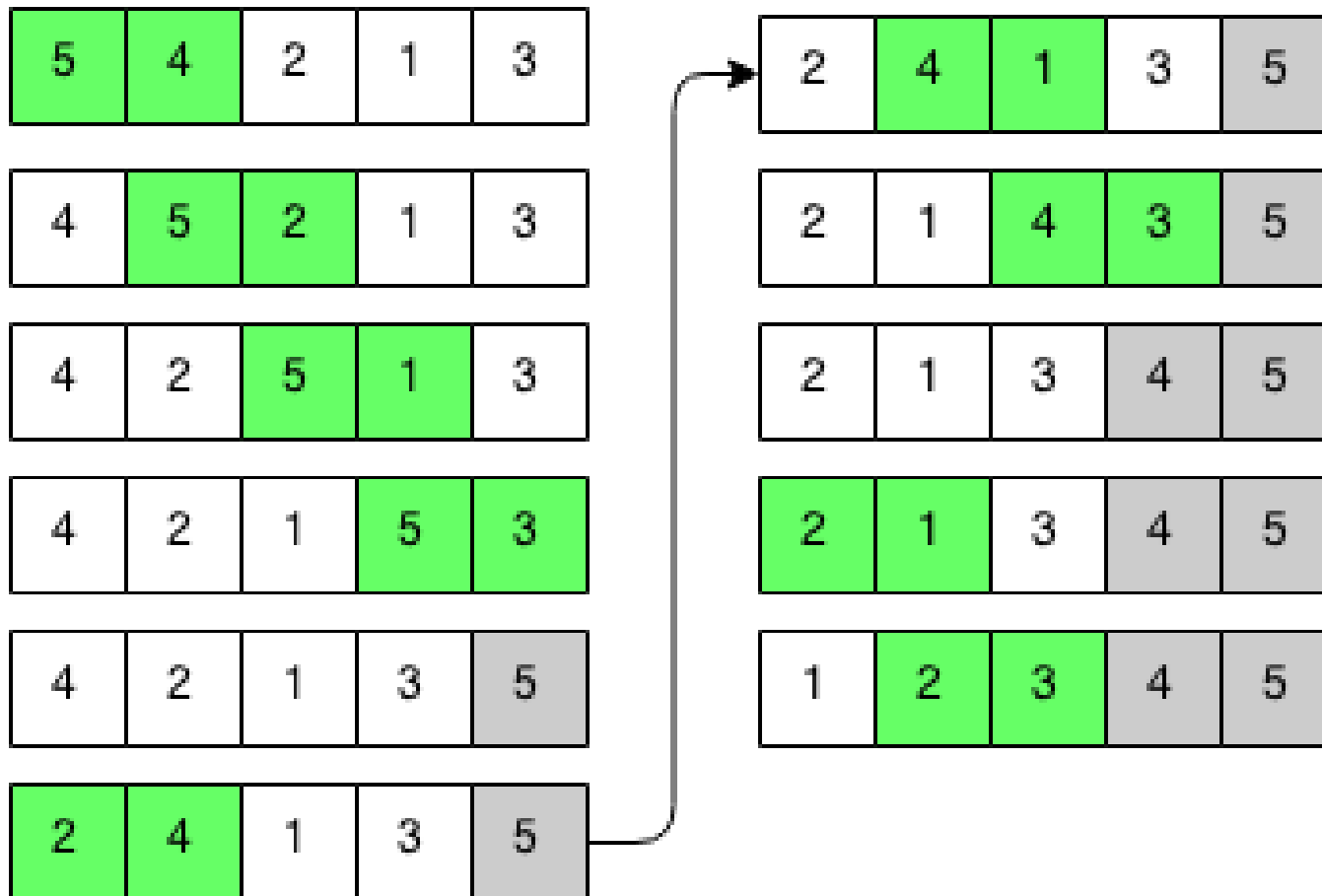
## 19



# BubbleSort

20

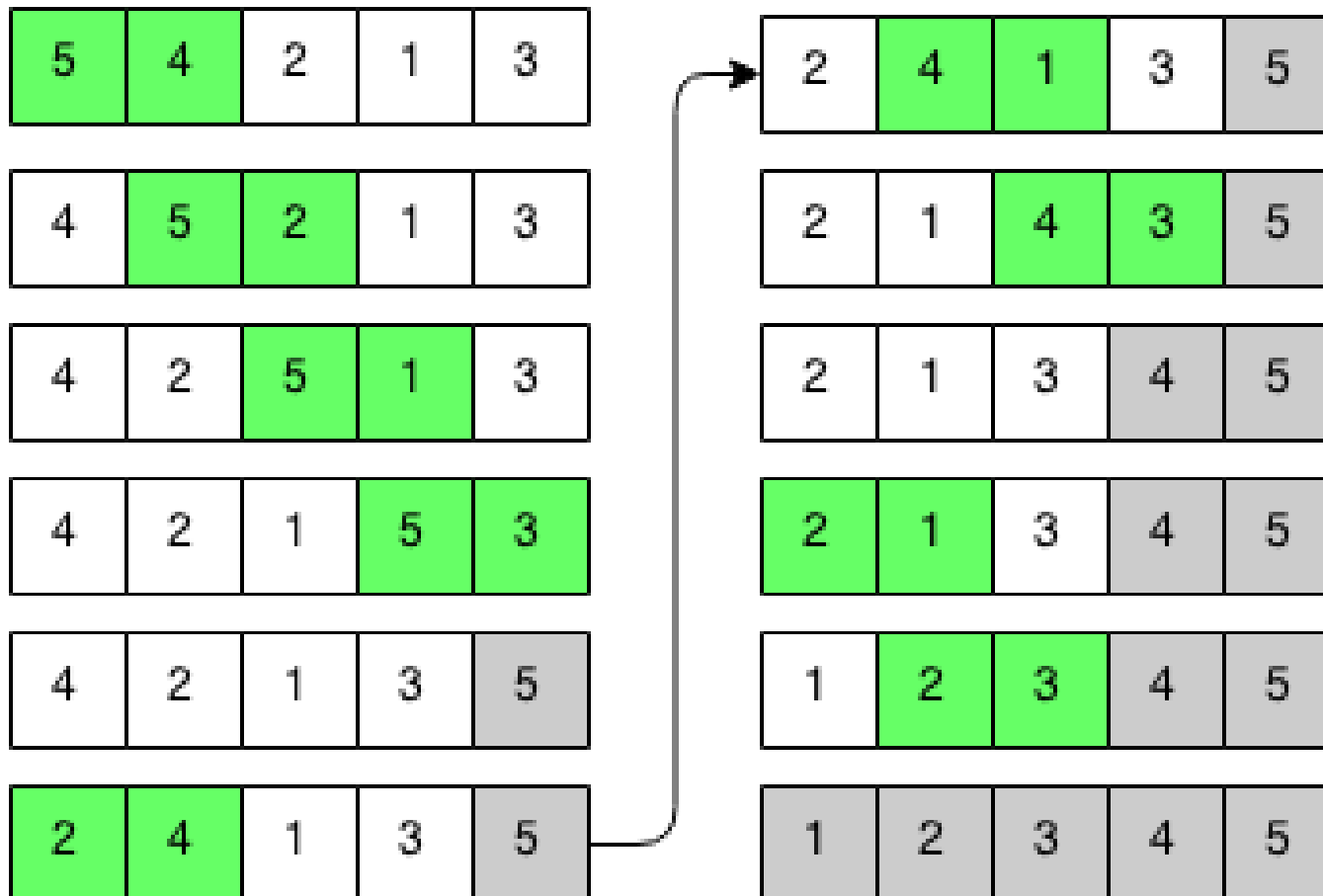
## □ Exemplo:



# BubbleSort

21

## □ Exemplo:



# BubbleSort

22

□ Exemplo:

6 5 3 1 8 7 2 4

# BubbleSort

23

- ❑ O algoritmo de ordenação mais simples existente.
- ❑ Usa comparações consecutivas, trocando elementos do vetor para ordená-lo.
- ❑ Percorre o vetor diversas vezes, fazendo os elementos “flutuarem” para sua posição.
- ❑ Complexidade:  $O(n^2)$ .
- ❑ Estável.
- ❑ Não adaptável. Se a entrada já estiver ordenada, executa a mesma quantidade de vezes.
- ❑ [Vídeo no Youtube](#)

# InsertionSort

24

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.



# InsertionSort

25

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18

# InsertionSort

26

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18

# InsertionSort

27

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18

INSERIR O PRÓXIMO  
ELEMENTO DO VETOR

# InsertionSort

28

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18

BUSCAR POSIÇÃO DENTRO  
DOS **ELEMENTOS JÁ**  
**ORDENADOS**

# InsertionSort

29

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18

# InsertionSort

30

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18

INSERIR O PRÓXIMO  
ELEMENTO DO VETOR

# InsertionSort

31

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	<u>42</u>	94	18

BUSCAR POSIÇÃO DENTRO  
DOS **ELEMENTOS JÁ**  
**ORDENADOS**

# InsertionSort

32

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18



# InsertionSort

33

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais						18
$i = 2$						18
$i = 3$						18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18

# InsertionSort

34

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais						18
$i = 2$						18
$i = 3$						18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18

**BUSCAR POSIÇÃO DENTRO DOS ELEMENTOS JÁ ORDENADOS**

# InsertionSort

35

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18
$i = 5$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18

# InsertionSort

36

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais						18
$i = 2$						18
$i = 3$						18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18
$i = 5$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18

# InsertionSort

37

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais						18
$i = 2$						18
$i = 3$						18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18
$i = 5$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	<b>18</b>

# InsertionSort

38

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18
$i = 5$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18
$i = 6$	<u>12</u>	<u>18</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>

# InsertionSort

39

- Inicia com os dois primeiros elementos do vetor, ordenando-os por comparação e troca.
- Iterativamente, pega o próximo elemento e busca o local ideal para inseri-lo.
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 2$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 3$	<u>12</u>	<u>44</u>	<u>55</u>	42	94	18
$i = 4$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	94	18
$i = 5$	<u>12</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>	18
$i = 6$	<u>12</u>	<u>18</u>	<u>42</u>	<u>44</u>	<u>55</u>	<u>94</u>

# InsertionSort

40

□ Exemplo:

6 5 3 1 8 7 2 4



# InsertionSort

41

- ❑ “Preenche” o vetor adicionando cada elemento na posição adequada.
- ❑ Complexidade:  $O(n^2)$ .
- ❑ Pior caso: entrada em ordem inversa.
- ❑ Melhor caso: entrada ordenada.
- ❑ Estável.
- ❑ Adaptável.
- ❑ [Vídeo no Youtube](#)

# SelectionSort

42

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .

# SelectionSort

43

- Seleciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18

# SelectionSort

44

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .

- Exemplo:

SELECIONA O MENOR  
ELEMENTO ENTRE OS NÃO  
ORDENADOS.

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18

# SelectionSort

45

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .

- Exemplo:

**TROCA** O VALOR COM O  
PRIMEIRO ÍNDICE NÃO  
ORDENADO

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<u>12</u>	<u>44</u>	55	42	94	18

# SelectionSort

46

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .

- Exemplo:

	SELECIONA O MENOR ELEMENTO ENTRE OS NÃO ORDENADOS.					
	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18

# SelectionSort

47

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .

- Exemplo:

**TROCA** O VALOR COM O  
PRIMEIRO ÍNDICE NÃO  
ORDENADO

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<u>18</u>	55	42	94	<u>44</u>

# SelectionSort

48

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

SELECIONA O MENOR  
ELEMENTO ENTRE OS NÃO  
ORDENADOS.

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	<u>42</u>	94	<u>18</u>



# SelectionSort

49

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

**TROCA** O VALOR COM O  
PRIMEIRO ÍNDICE NÃO  
ORDENADO

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>18</u>
$i = 3$	<del>44</del>	<del>12</del>	<u>42</u>	<u>55</u>	94	44

# SelectionSort

50

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

SELECIONA O MENOR  
ELEMENTO ENTRE OS NÃO  
ORDENADOS.

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>18</u>
$i = 3$	<del>44</del>	<del>12</del>	<del>55</del>	<u>42</u>	94	<u>18</u>

# SelectionSort

51

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

**TROCA** O VALOR COM O  
PRIMEIRO ÍNDICE NÃO  
ORDENADO

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>44</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>44</u>
$i = 3$	<del>44</del>	<del>12</del>	<del>55</del>	<u>55</u>	94	44
$i = 4$	<del>44</del>	<del>12</del>	<del>55</del>	<u>44</u>	94	<u>55</u>

# SelectionSort

52

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

SELECIONA O MENOR  
ELEMENTO ENTRE OS NÃO  
ORDENADOS.

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>18</u>
$i = 3$	<del>44</del>	<del>12</del>	<del>55</del>	<u>42</u>	94	18
$i = 4$	<del>44</del>	<del>12</del>	<del>55</del>	<del>42</del>	94	<u>55</u>

# SelectionSort

53

- Selecione o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposicione no índice  $i$ .

- Exemplo:

**TROCA O VALOR COM O  
PRIMEIRO ÍNDICE NÃO  
ORDENADO**

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>18</u>
$i = 3$	<del>44</del>	<del>12</del>	<del>55</del>	<u>42</u>	94	18
$i = 4$	<del>44</del>	<del>12</del>	<del>55</del>	<del>42</del>	94	<u>18</u>
$i = 5$	<del>44</del>	<del>12</del>	<del>55</del>	<del>42</del>	<u>55</u>	18

# SelectionSort

54

- Selecciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .
- Exemplo:

RESTA APENAS O ÚLTIMO  
NA POSIÇÃO CORRETA

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<del>44</del>	<u>12</u>	55	42	94	18
$i = 2$	<del>44</del>	<del>12</del>	55	42	94	<u>44</u>
$i = 3$	<del>44</del>	<del>12</del>	<del>55</del>	<u>42</u>	94	44
$i = 4$	<del>44</del>	<del>12</del>	<del>55</del>	<del>42</del>	94	<u>55</u>
$i = 5$	<del>44</del>	<del>12</del>	<del>55</del>	<del>42</del>	<del>94</del>	<u>94</u>

# SelectionSort

55

- Seleciona o elemento com menor chave, dentro dos  $n-i$  elementos não ordenados, e reposiciona no índice  $i$ .
- Exemplo:

	1	2	3	4	5	6
Chaves iniciais	44	12	55	42	94	18
$i = 1$	<u>12</u>	<u>44</u>	55	42	94	18
$i = 2$	12	<u>18</u>	55	42	94	<u>44</u>
$i = 3$	12	18	<u>42</u>	<u>55</u>	94	44
$i = 4$	12	18	42	<u>44</u>	94	<u>55</u>
$i = 5$	12	18	42	44	<u>55</u>	<u>94</u>

# SelectionSort

56

□ Exemplo:

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7



# SelectionSort

57

- A cada iteração procura a chave de menor valor ainda não ordenada, e insere na posição correta.
- Complexidade:  $O(n^2)$ .
- Estável.
- Não é adaptável. Realiza todas as verificações se já estiver ordenado.
- [Vídeo no Youtube](#)