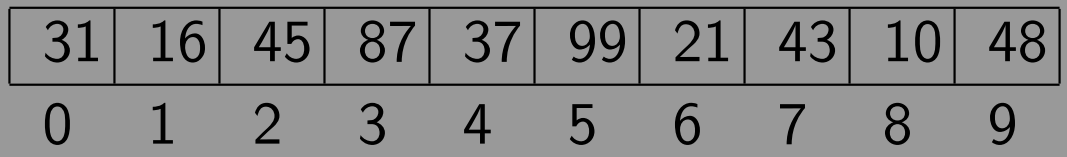
Busca Sequencial, Busca Binária e Ordenação BubbleSort

Prof. Andrey Masiero

4 de setembro de 2017

Agenda

Busca Sequencial



- Qual o algoritmo para encontrar o número 87?

Busca Sequencial

31	16	45	87	37	99	21	43	10	48
0	1	2	3	4	5	6	7	8	9

- Qual o algoritmo para encontrar o número 87?
- Dado que o vetor tem 10 posições.

Busca Sequencial

31	16	45	87	37	99	21	43	10	48
0	1	2	3	4	5	6	7	8	9

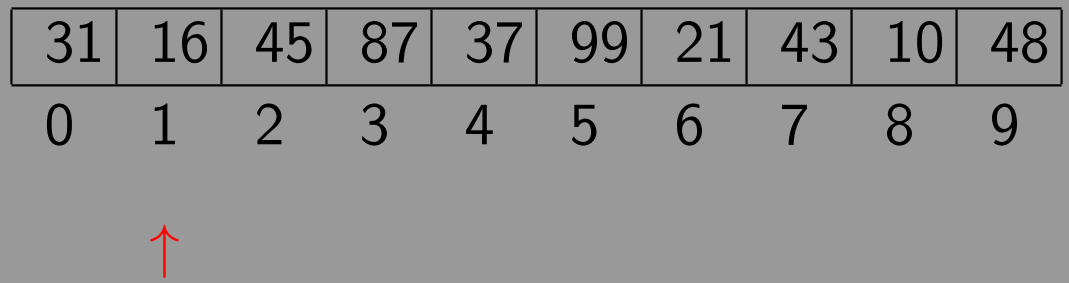
- Qual o algoritmo para encontrar o número 87?
- Dado que o vetor tem 10 posições.
- Pode-se ser feito uma busca sequencial.

Busca Sequencial

31	16	45	87	37	99	21	43	10	48
0	1	2	3	4	5	6	7	8	9

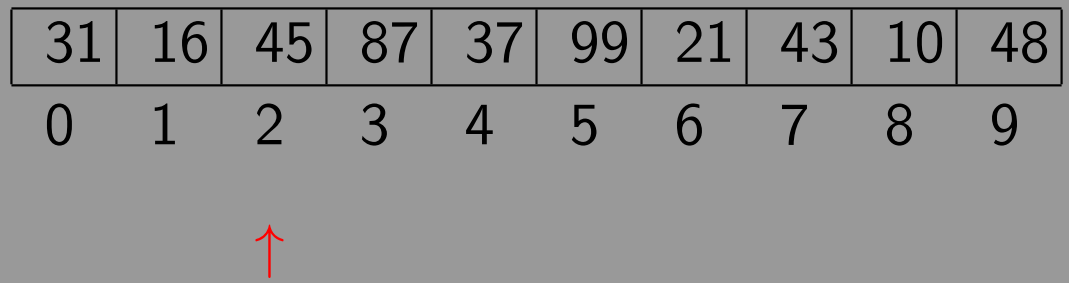
$31 == 87$? Não

Busca Sequencial

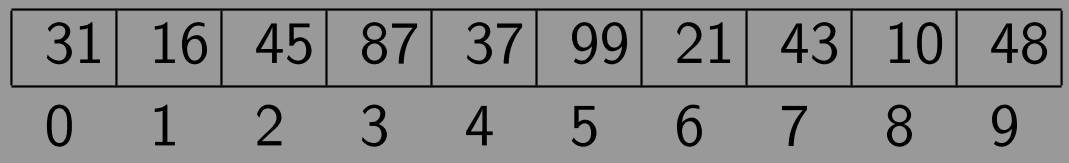


$16 == 87$? Não

Busca Sequencial



Busca Sequencial



$87 == 87$? Sim

Encontrado!

Busca Sequencial

```
vetor[10] = {31, 16, 45, 87, 37, 99, 21, 43, 10, 48}
valorProcurado = 87
indice = -1

for (i = 0; i < vetor.length - 1; i++) {
    if (vetor[i] == valorProcurado) {
        indice = i
    }
}
```

Existe uma maneira mais eficiente de buscar um elemento?

Busca Binária

- O vetor precisa estar ordenado, caso contrário o método não funciona.
- Esse é o algoritmo de Busca Binária

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Agora com o vetor ordenado.

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Agora com o vetor ordenado.
- Qual o algoritmo para encontrar o número 87?

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Agora com o vetor ordenado.
- Qual o algoritmo para encontrar o número 87?
- Dado que o vetor tem 10 posições.

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)
- Encontrar o valor de posição central (centro = (inicio+9)/2)

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)
- Encontrar o valor de posição central (centro = (inicio+9)/2)
- Compare se o valor armazenado com o procurado:

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)
- Encontrar o valor de posição central (centro = (inicio+9)/2)
- Compare se o valor armazenado com o procurado:
  - Caso encontrado, retorne a posicao

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

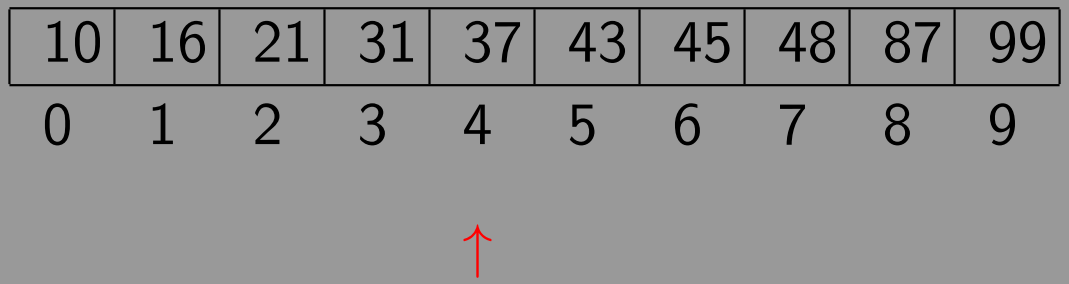
- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)
- Encontrar o valor de posição central (centro = (inicio+9)/2)
- Compare se o valor armazenado com o procurado:
  - Caso encontrado, retorne a posicao
  - Caso valor procurado seja maior inicio = centro + 1

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

- Primeiro armazena o valor da posição inicial (inicio = 0)
- Depois, armazena o valor da posição final (fim = 9)
- Encontrar o valor de posição central (centro = (inicio+9)/2)
- Compare se o valor armazenado com o procurado:
  - Caso encontrado, retorne a posicao
  - Caso valor procurado seja maior inicio = centro + 1
  - Caso valor procurado seja menor fim = centro - 1

Busca Binária



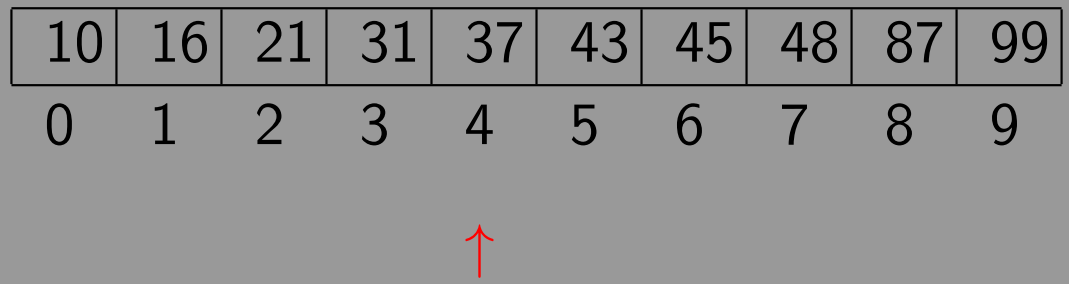
- $centro = (0 + 9)/2 \rightarrow centro = 4.5 \rightarrow centro = 4$

Busca Binária

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

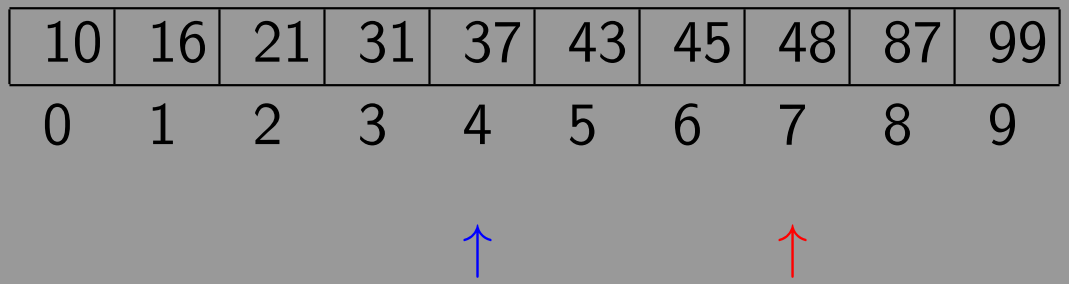
- $centro = (0 + 9)/2 \rightarrow centro = 4.5 \rightarrow centro = 4$
- $37 == 87$? Não

Busca Binária



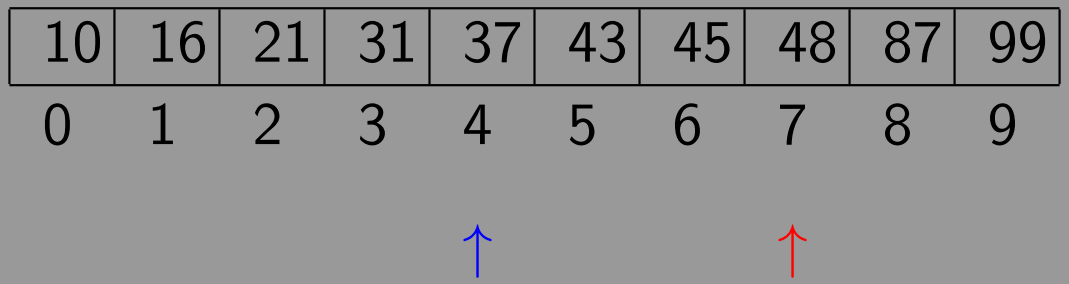
- $centro = (0 + 9)/2 \rightarrow centro = 4.5 \rightarrow centro = 4$
- $37 == 87$? Não
- $37 < 87$? Sim $\rightarrow inicio = centro + 1 \rightarrow inicio = 5$

Busca Binária



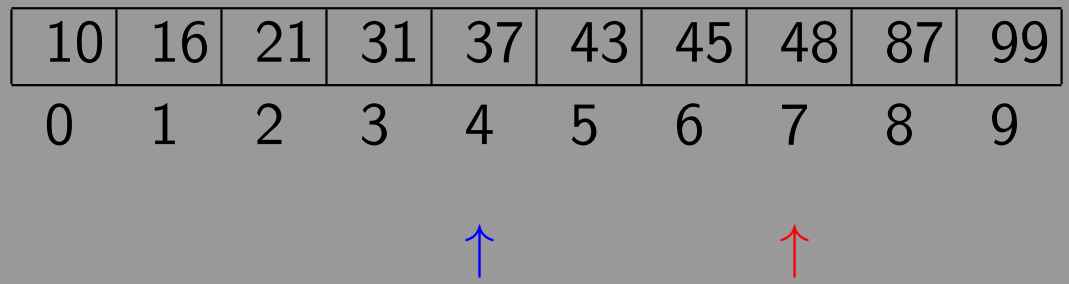
- $centro = (5 + 9)/2 \rightarrow centro = 7$

Busca Binária



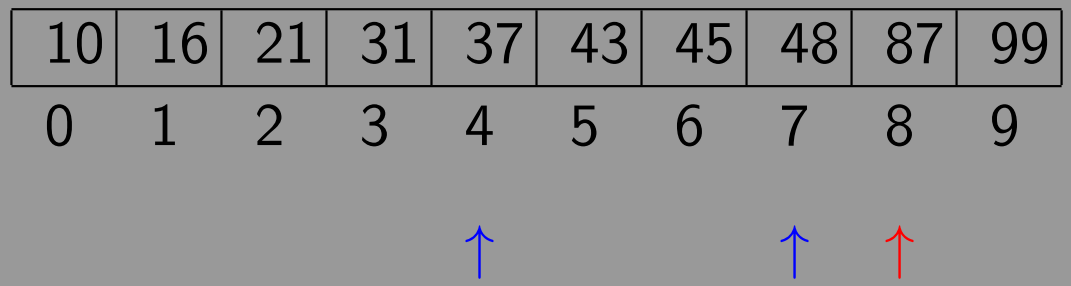
- $centro = (5 + 9)/2 \rightarrow centro = 7$
- $48 == 87$? Não

Busca Binária



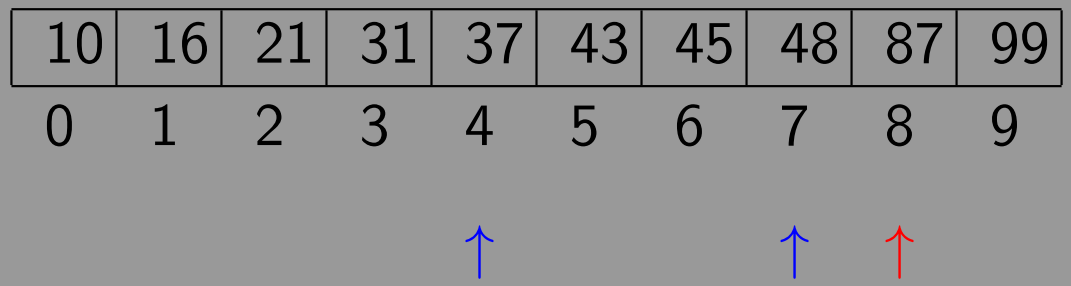
- $centro = (5 + 9)/2 \rightarrow centro = 7$
- $48 == 87$? Não
- $48 < 87$? Sim $\rightarrow inicio = centro + 1 \rightarrow inicio = 8$

Busca Binária



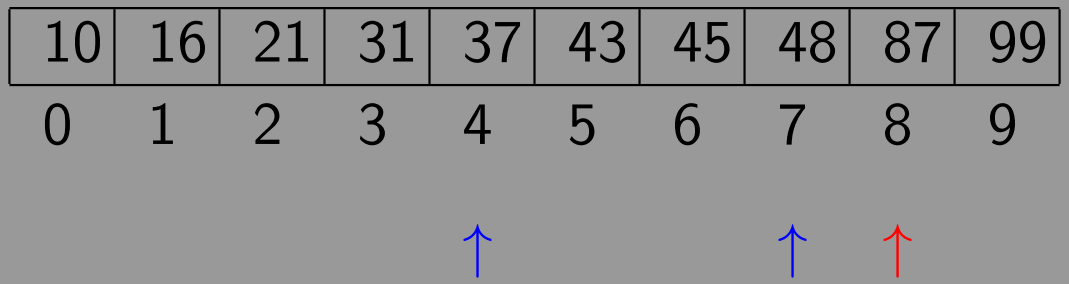
- $centro = (8 + 9)/2 \rightarrow centro = 8.5 \rightarrow centro = 8$

Busca Binária



- $centro = (8 + 9)/2 \rightarrow centro = 8.5 \rightarrow centro = 8$
- $87 == 87$? Sim

Busca Binária



- $centro = (8 + 9)/2 \rightarrow centro = 8.5 \rightarrow centro = 8$
- $87 == 87$? Sim

Encontrado

Busca Binária

```
vetor[10] = {10, 16, 21, 31, 37, 43, 45, 48, 87, 99}
inicio = 0;
fim = vetor.length - 1;
centro;
do{
        centro = inicio + (fim - inicio)/2;
        if(x < vetor[centro])
                fim = centro - 1;
        else if(x > vetor[centro])
                inicio = centro + 1;
        else return centro;
}while(inicio <= fim);
```

Busca Sequencial vs. Busca Binária

- Busca Sequencial é $O(n)$
- Busca Binária é $O(\log_2 n)$
- Busca Binária é mais rápida na Sequencial, porém precisa dos dados ordenados
- Para ordernar os dados existem vários métodos

BubbleSort

- Existem diversos tipos de métodos para ordenar os dados

BubbleSort

- Existem diversos tipos de métodos para ordenar os dados
- A complexidade deles tem a variação de $O(n \log_2 n)$ até $O(n^2)$

BubbleSort

- Existem diversos tipos de métodos para ordenar os dados
- A complexidade deles tem a variação de $O(n \log_2 n)$ até $O(n^2)$
- O mais fácil de implementar é o BubbleSort

BubbleSort

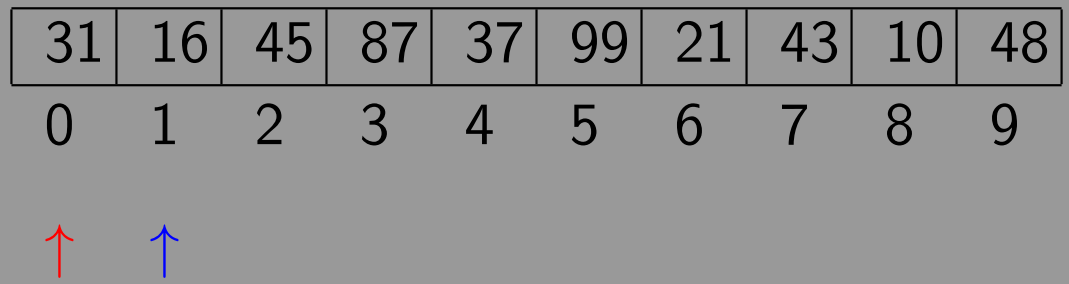
- Existem diversos tipos de métodos para ordenar os dados
- A complexidade deles tem a variação de $O(n \log_2 n)$ até $O(n^2)$
- O mais fácil de implementar é o BubbleSort
- Porém a complexidade dele é $O(n^2)$

BubbleSort

Vamos ordernar este vetor!

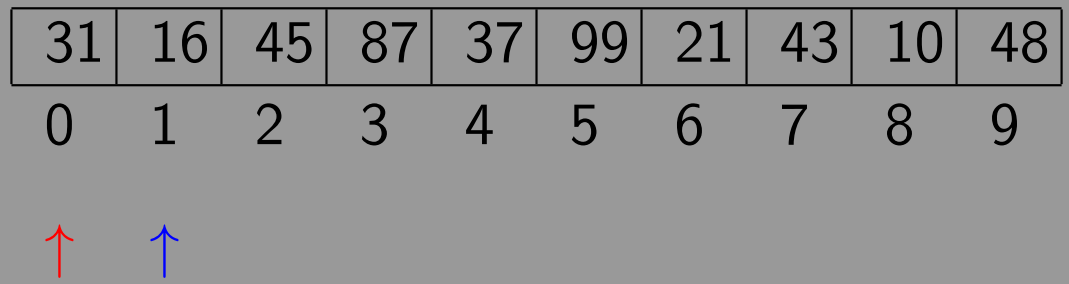
31	16	45	87	37	99	21	43	10	48
0	1	2	3	4	5	6	7	8	9

BubbleSort



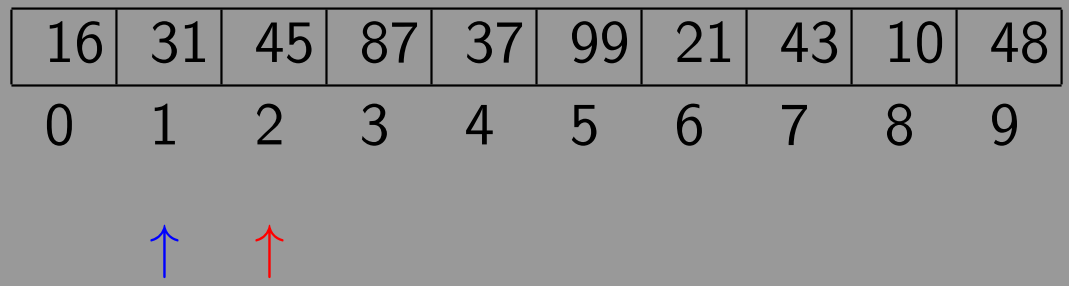
- $31 > 16$? Sim

BubbleSort



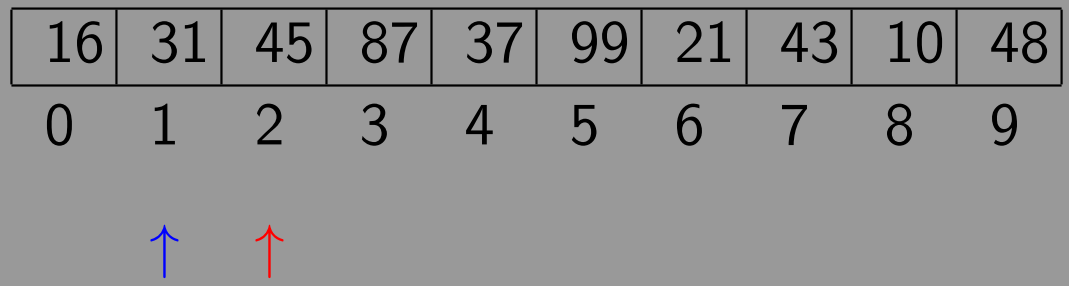
- $31 > 16$? Sim
- Então troca

BubbleSort



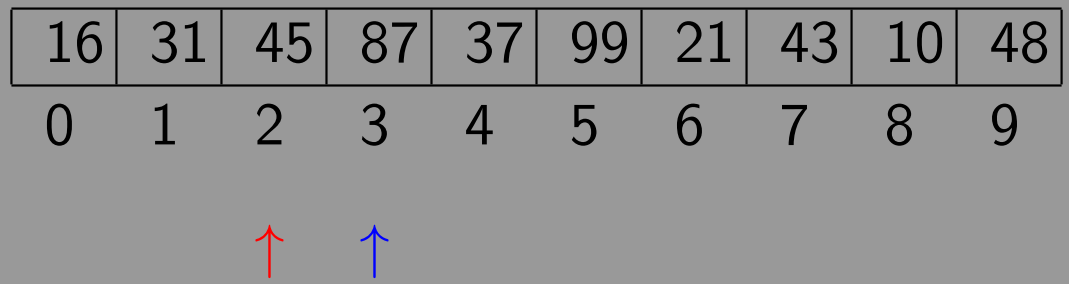
- $31 > 45$? Não

BubbleSort



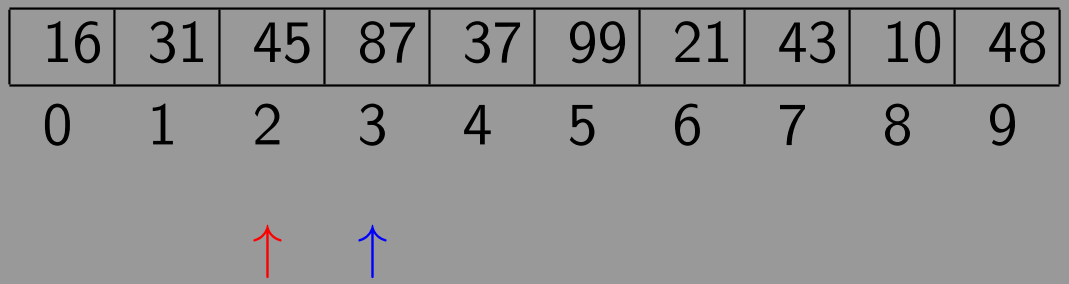
- $31 > 45$? Não
- Então não troca

BubbleSort



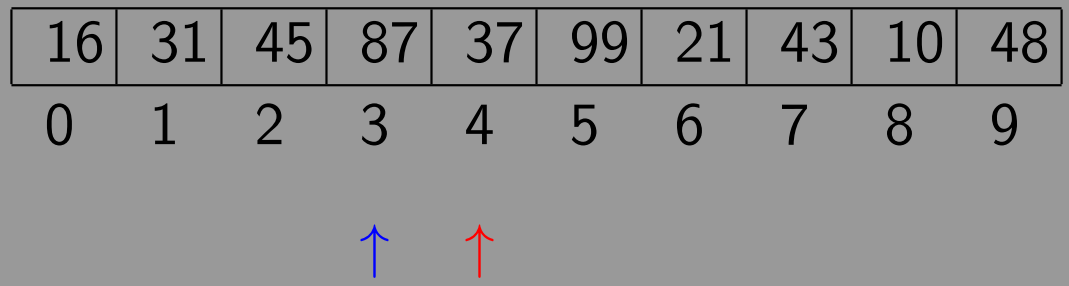
- $45 > 87$? Não

BubbleSort



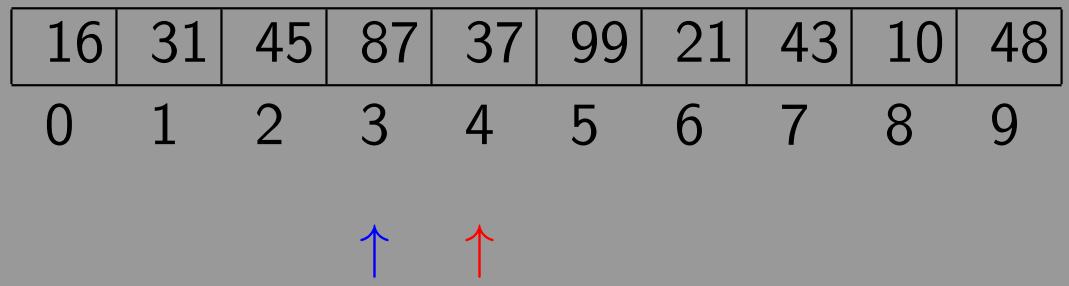
- $45 > 87$? Não
- Então não troca

BubbleSort



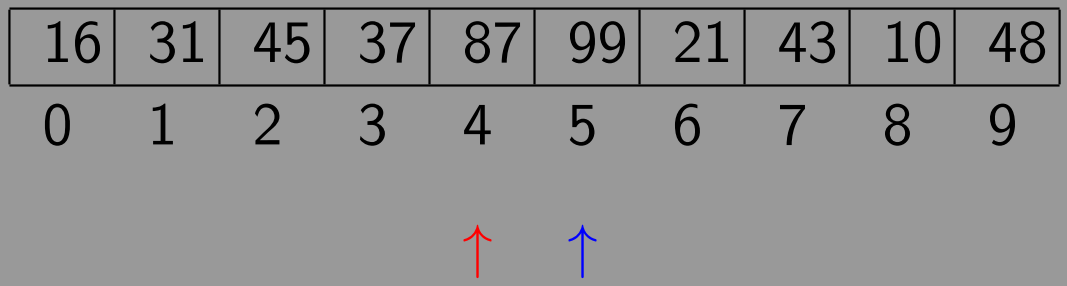
- $87 > 37$? Sim

BubbleSort



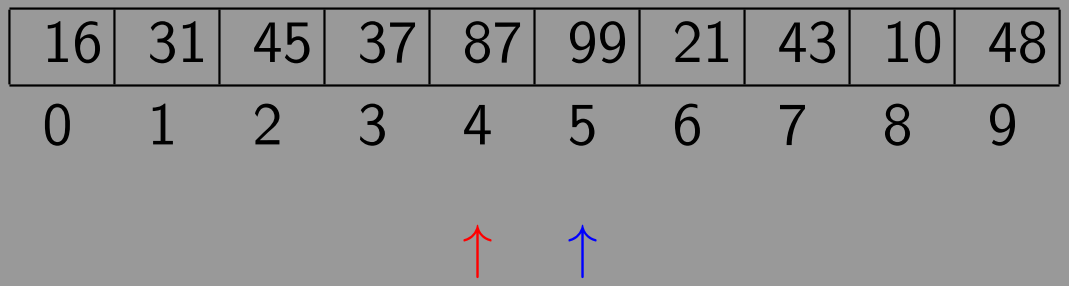
- $87 > 37$? Sim
- Então troca

BubbleSort



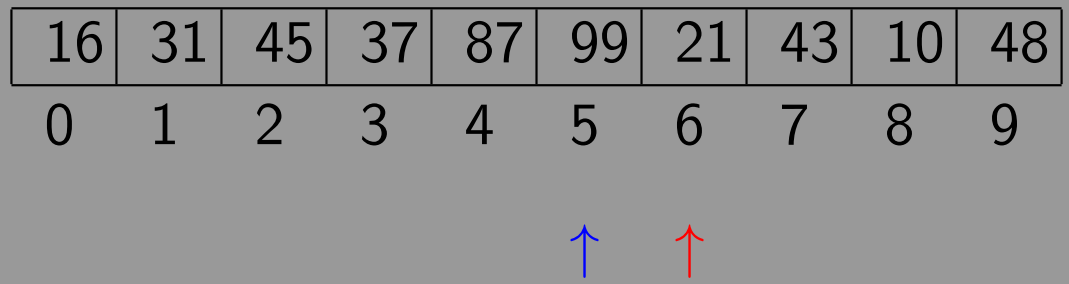
- $87 > 99$? Não

BubbleSort



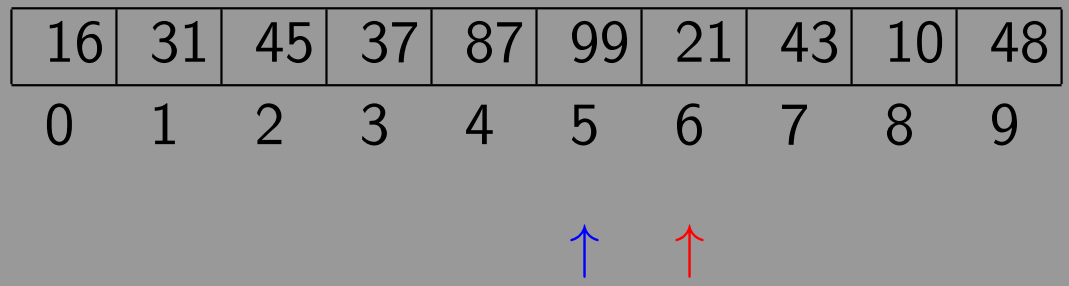
- $87 > 99$? Não
- Então não troca

BubbleSort



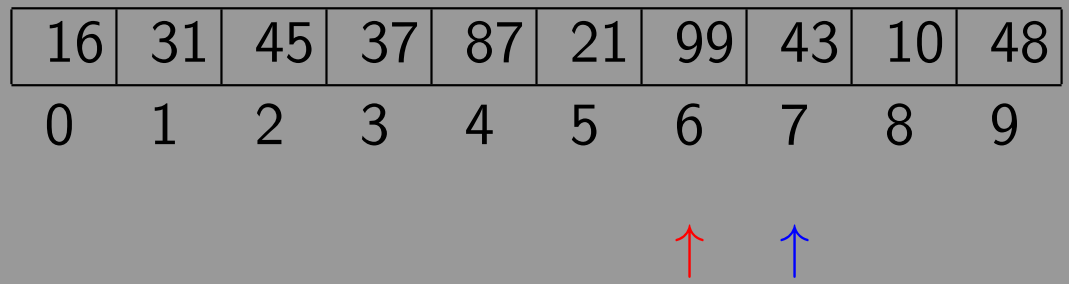
- $99 > 21$? Sim

BubbleSort



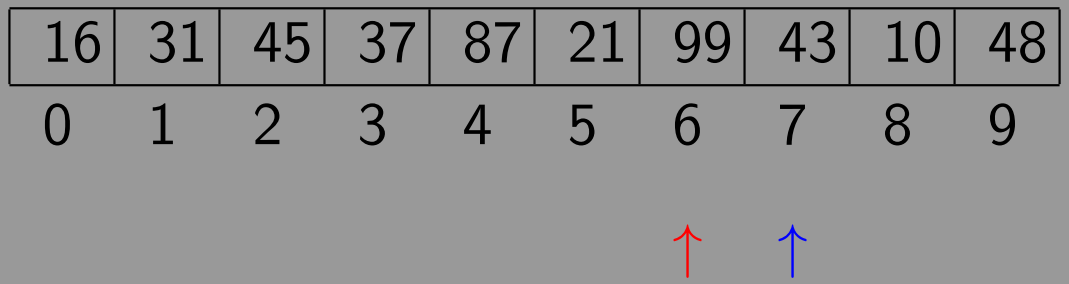
- $99 > 21$? Sim
- Então troca

BubbleSort



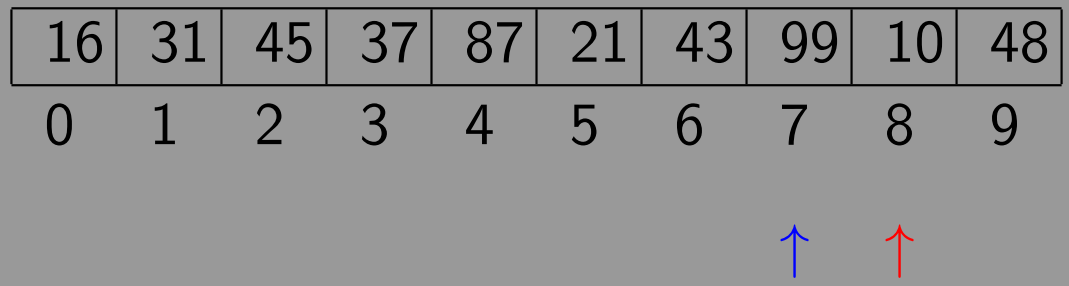
- $99 > 43$? Sim

BubbleSort



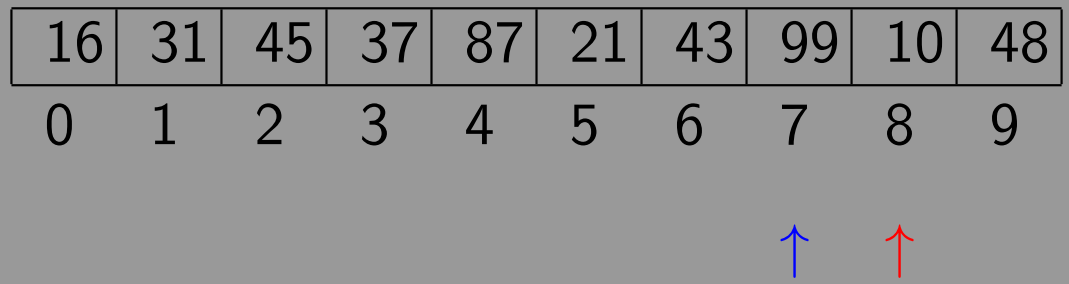
- $99 > 43$? Sim
- Então troca

BubbleSort



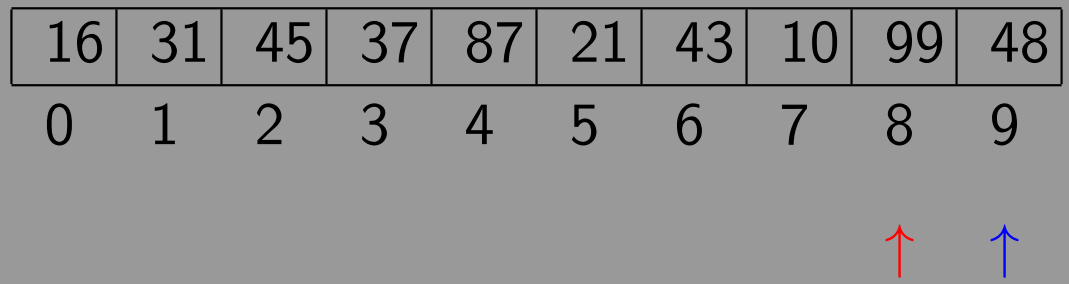
- $99 > 10$? Sim

BubbleSort



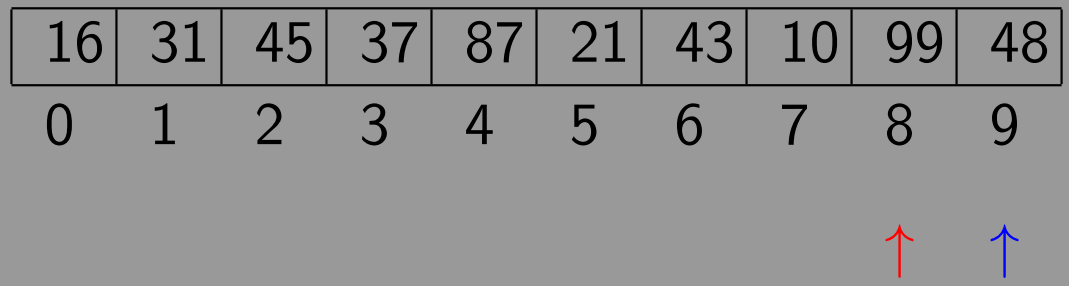
- $99 > 10$? Sim
- Então troca

BubbleSort



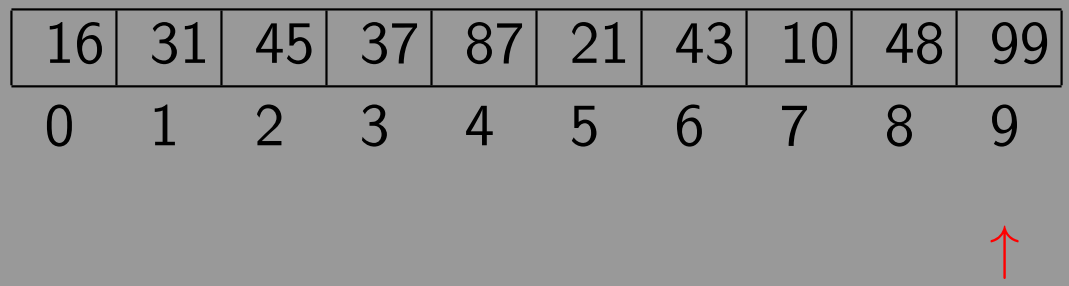
- $99 > 48$? Sim

BubbleSort



- $99 > 48$? Sim
- Então troca

BubbleSort



- 99 se mantém na última posição do vetor, pois está ordenado

BubbleSort

- O processo deve ser repetido até a ordenação completa do vetor

10	16	21	31	37	43	45	48	87	99
0	1	2	3	4	5	6	7	8	9

BubbleSort

```
vetor[10] = {31, 16, 45, 87, 37, 99, 21, 43, 10, 48}

for (i = 0; i < vetor.length - 1; i++) {
    for (j = 0; j < vetor.length - 1; j++) {
        if (vetor[j] > vetor[j + 1]) {
            swap(vetor[j], vetor[j + 1]);
        }
    }
}
```

Exercícios

1. Implemente o método de Busca Sequencial.
2. Implemente o método de Busca Binária.
3. Implemente o método de ordenação BubbleSort.
4. Teste todos os algoritmos em um algoritmo principal

Referências Bibliográficas

1. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to algorithms second edition." (2001).
2. Tamassia, Roberto, and Michael T. Goodrich. "Estrutura de Dados e Algoritmos em Java." Porto Alegre, Ed. Bookman 4 (2007).
3. Ascencio, Ana Fernanda Gomes, and Graziela Santos de Araújo. "Estruturas de Dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++." São Paulo: Perarson Prentice Halt 3 (2010).