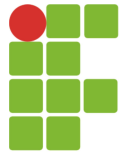
Ordenação – Bubblesort

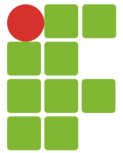
Estrutura de Dados e Algoritmos – Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

IFB – Instituto Federal de Brasília,
Campus Taguatinga



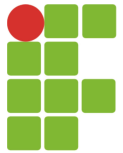
Sumário



Bubblesort

Bubblesort

- O Bubblesort, em cada iteração, lê o vetor da esquerda para a direita e troca os elementos se $v[i] > v[i+1]$.
- Como consequência disso, os maiores elementos são colocados em sua posição devida após cada iteração.
- Observe que são necessárias $n-1$ iterações para o algoritmo ordenar a sequência original, sendo que cada iteração precisa passar por toda a sequência.
- É possível implementar uma otimização que interrompe as varreduras do vetor assim que é detectado que o vetor já está ordenado, mas isso não impacta no pior caso.

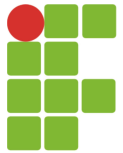


Bubblesort

Exemplo

11	17	23	2	7	29	3	13	5	19
11	17	**2**	**23**	7	29	3	13	5	19
11	17	2	**7**	**23**	29	3	13	5	19
11	17	2	7	23	**3**	**29**	13	5	19
11	17	2	7	23	3	**13**	**29**	5	19
11	17	2	7	23	3	13	**5**	**29**	19
11	17	2	7	23	3	13	5	**19**	**29**

11	**2**	**17**	7	23	3	13	5	19	29
11	2	**7**	**17**	23	3	13	5	19	29
11	2	7	17	**3**	**23**	13	5	19	29
11	2	7	17	3	**13**	**23**	5	19	29
11	2	7	17	3	13	**5**	**23**	19	29
11	2	7	17	3	13	5	**19**	**23**	29



Bubblesort

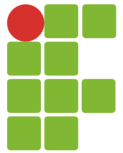
Exemplo

11	2	7	17	3	13	5	19	23	29
2	11	7	17	3	13	5	19	23	29
2	7	11	17	3	13	5	19	23	29
2	7	11	3	17	13	5	19	23	29
2	7	11	3	13	17	5	19	23	29
2	7	11	3	13	5	17	19	23	29

2	7	3	11	13	5	17	19	23	29
2	7	3	11	5	13	17	19	23	29

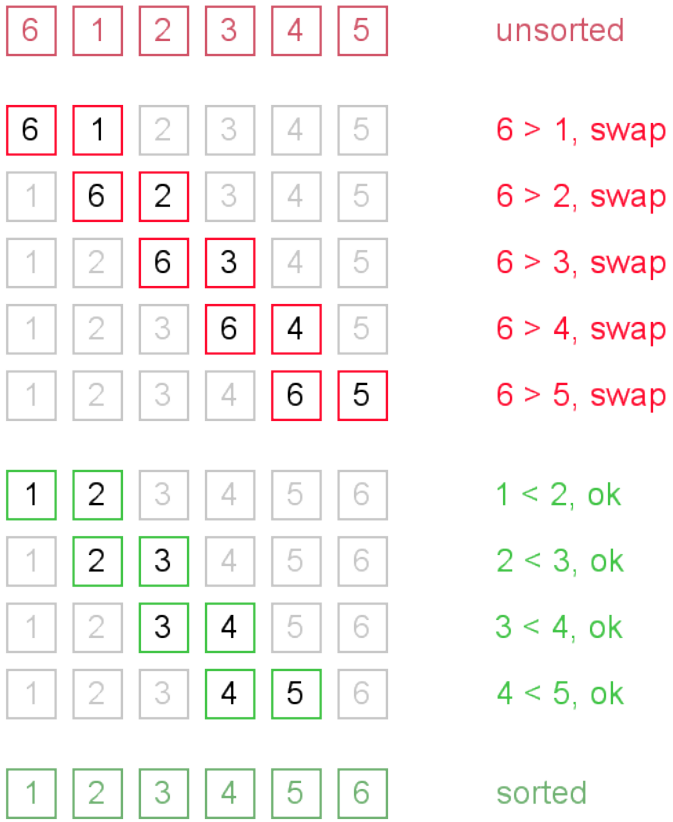
2	3	7	11	5	13	17	19	23	29
2	3	7	5	11	13	17	19	23	29

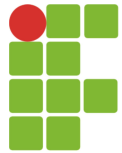
2	3	5	7	11	13	17	19	23	29
2	3	5	7	11	13	17	19	23	29



Bubblesort

Exemplo

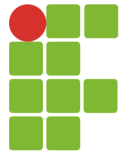




Bubblesort

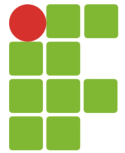
Exemplo

5 1 12 -5 16	unsorted
5 1 12 -5 16	5 > 1, swap
1 5 12 -5 16	5 < 12, ok
1 5 12 -5 16	12 > -5, swap
1 5 -5 12 16	12 < 16, ok
1 5 -5 12 16	1 < 5, ok
1 5 -5 12 16	5 > -5, swap
1 -5 5 12 16	5 < 12, ok
1 -5 5 12 16	1 > -5, swap
-5 1 5 12 16	1 < 5, ok
-5 1 5 12 16	-5 < 1, ok
-5 1 5 12 16	sorted

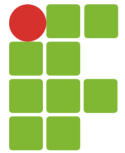


Bubblesort

```
1  void swap(int *v, int i, int j) {
2      int t = v[i];
3      v[i] = v[j];
4      v[j] = t;
5  }
6
7  void bubble_sort(int *v, size_t size) {
8      int swapped = 1;
9      for (int i = 0; i < size - 1 && swapped; i++) {
10         swapped = 0;
11         for (int j = 0; j < size - i - 1; j++) {
12             if (v[j] > v[j + 1]) {
13                 swap(v, j, j + 1);
14                 swapped = 1;
15             }
16         }
17     }
18 }
```



Sumário



Bubblesort

Análise

No pior caso, são necessários $n-1$ iterações sobre a sequência original. Na iteração i são realizadas $n-1-i$ comparações ao todo. Portanto, o custo do algoritmo é dado como:

$$\sum_{i=0}^{n-1} i = 1 + 2 + \ldots + n - 1 \in \Theta(n^2)$$

In-place	Estável
✓	✓