

# Busca e Ordenação

## Algoritmos de Ordenação Quadráticos

---

Prof. Edson Alves - UnB/FGA

2018

1. Fundamentos de Ordenação
2. *Selection Sort*
3. *Insertion Sort*
4. *Bubble Sort*

# Fundamentos de Ordenação

---

# Ordenação parcial e ordenação total

- Seja  $a_s = \{a_1, a_2, \dots, a_N\}$  uma sequência de  $N$  objetos, denominados elementos
- Seja  $R \in a_s \times a_s$  uma relação
- Dados dois elementos  $a_i, a_j \in a_s$ ,  $a_i$  se relaciona com  $a_j$  se  $(a_i, a_j) \in R$  (isto não implica necessariamente que  $a_j$  se relaciona com  $a_i$ )
- Seja  $S = \{a \in a_s \mid \exists b \in a_s : (a, b) \in R \vee (b, a) \in R\}$
- Dizemos que  $R$  é uma relação de ordem parcial se, para todos  $a, b, c \in S$ , temos que
  1.  $(a, a) \in R$
  2. se  $(a, b) \in R$  e  $(b, a) \in R$  então  $a$  e  $b$  são iguais
  3. se  $(a, b) \in R$  e  $(b, c) \in R$  então  $(a, c) \in R$
- Se para todos  $a, b \in a_s$  vale  $(a, b) \in R$  ou  $(b, a) \in R$ , então  $R$  é uma relação de ordem total

## Definição de ordenação

- Dizemos que uma sequência  $a_s$  está ordenada de acordo com a relação de ordem  $R$  se, para todos  $i = 2, 3, \dots, N$ , temos que  $(a_{i-1}, a_i) \in R$
- Um algoritmo de ordenação  $A(a_s, R)$  recebe, como entrada, uma sequência  $a_s$  e uma relação de ordem  $R$  e, ao final do algoritmo, a sequência  $a_s$  (ou uma nova sequência  $b_s$ ) está ordenada de acordo com a relação  $R$
- Na prática, a relação  $R$  é implementada como uma função binária  $f$  tal que  $f(a, b)$  retorna verdadeiro se  $(a, b) \in R$
- Como a definição de ordenação depende da relação  $R$ , uma mesma sequência pode estar ordenada de acordo com  $R_1$  e não ordenada de acordo com  $R_2$

# Características dos algoritmos de ordenação

- Se a sequência a ser ordenada pode ser armazenada inteiramente em memória, o algoritmo é dito interno; caso contrário, é chamado externo
- Se o algoritmo usa apenas a memória da própria sequência (e talvez uma pequena quantidade adicional para variáveis temporárias), o algoritmo é denominado *in-place*
- Se o algoritmo demanda uma cópia extra da sequência, é chamado *not-in-place* ou *out-of-place*
- Um algoritmo de ordenação é estável se ele preserva a ordem relativa de elementos iguais

# Selection Sort

---

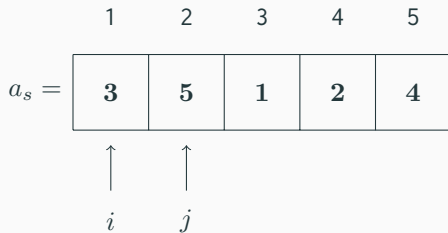
# Selection Sort

- *Selection sort* é um algoritmo de ordenação de simples entendimento e codificação
- Primeiramente ele identifica o menor dentre todos os elementos da sequência e o armazena na primeira posição
- Em seguida, procura o menor elemento dentre os que restaram, e o move para segunda posição
- Em faz o mesmo para a terceira, quarta, até a última posição
- A complexidade assintótica é  $O(N^2)$ , onde  $N$  é o número de elementos da sequência



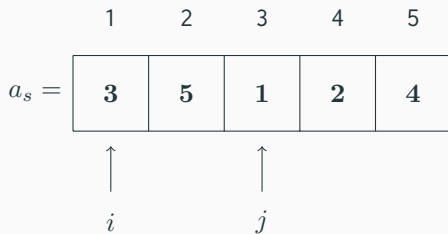
# Visualização do selection sort

$$k = 1$$



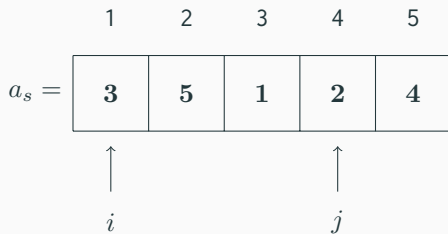
# Visualização do selection sort

$$k = 3$$



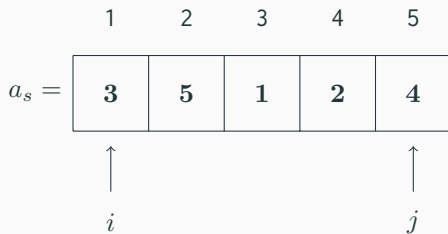
# Visualização do selection sort

$$k = 3$$



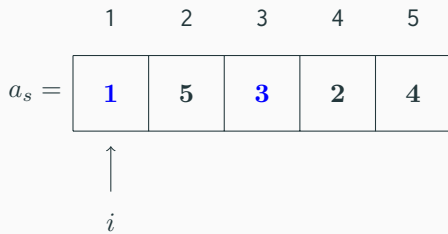
# Visualização do selection sort

$$k = 3$$



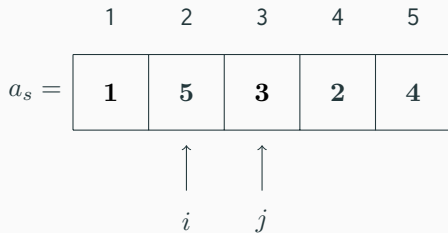
# Visualização do selection sort

$$k = 3$$



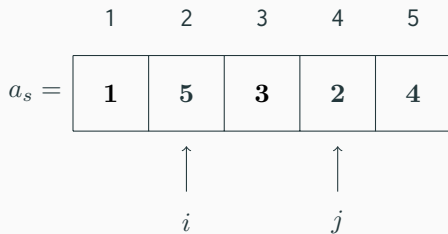
# Visualização do selection sort

$$k = 3$$



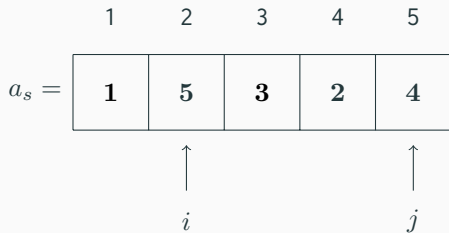
# Visualização do selection sort

$$k = 4$$



# Visualização do selection sort

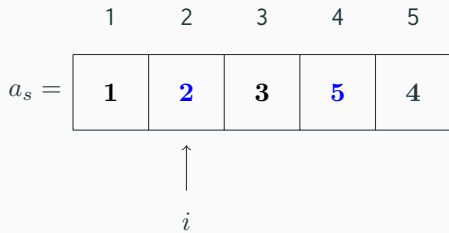
$$k = 4$$





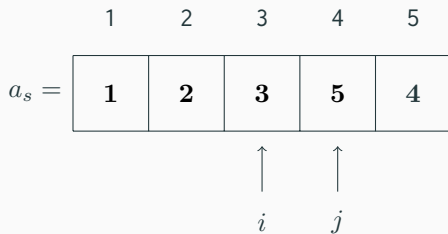
# Visualização do selection sort

$$k = 4$$



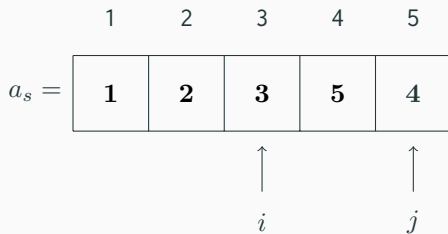
# Visualização do selection sort

$$k = 3$$



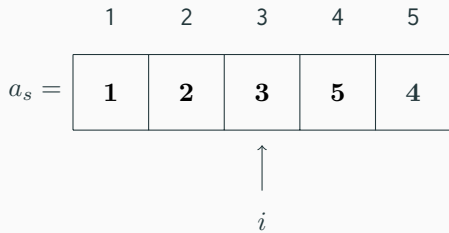
# Visualização do selection sort

$$k = 3$$



# Visualização do selection sort

$$k = 3$$



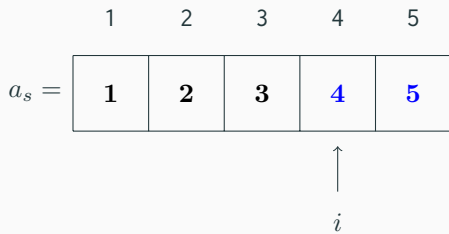
# Visualização do selection sort

$$k = 5$$

	1	2	3	4	5
$a_s =$	1	2	3	5	4
				$\uparrow$	$\uparrow$
				$i$	$j$

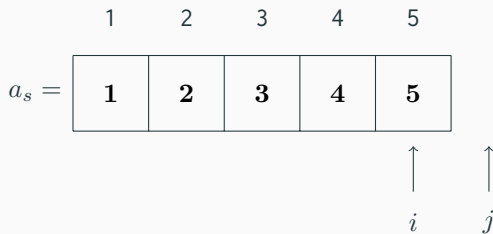
# Visualização do selection sort

$$k = 5$$



# Visualização do selection sort

$$k = 5$$



# Implementação do selection sort em C++

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 template<typename T>
7 void selection_sort(vector<T>& as)
8 {
9     int N = as.size();
10
11     for (int i = 0; i < N; ++i)
12     {
13         int k = i;                                // k = índice do menor elemento
14
15         for (int j = i + 1; j < N; ++j)
16             if (as[j] < as[k])
17                 k = j;
18
19         swap(as[i], as[k]);
20     }
21 }
```



# Implementação do selection sort em C++

```
22
23 int main()
24 {
25     vector<int> as { 3, 5, 1, 2, 4 };
26
27     selection_sort<int>(as);
28
29     for (size_t i = 0; i < as.size(); ++i)
30         cout << as[i] << (i + 1 == as.size() ? '\n' : ' ');
31
32     return 0;
33 }
```

## Observações sobre o selection sort

- O pior caso do algoritmo acontece quando a sequência está ordenado em sentido contrário, isto é,  $\forall i = 2, 3, \dots, N, (a_i, a_{i-1}) \in R$
- No pior caso, é feita 1 atribuição no início,  $6N$  atribuições no laço externo (considerando 3 atribuições por `swap()`) e

$$\sum_{i=0}^{N-1} 2(N - i - 1) = 2 \sum_{k=0}^{N-1} k = N(N - 1)$$

atribuições no laço interno

- Assim,  $f(N) = 1 + 6N + N(N - 1)$  é  $O(N^2)$
- O *selection sort* é um algoritmo estável *in-place*

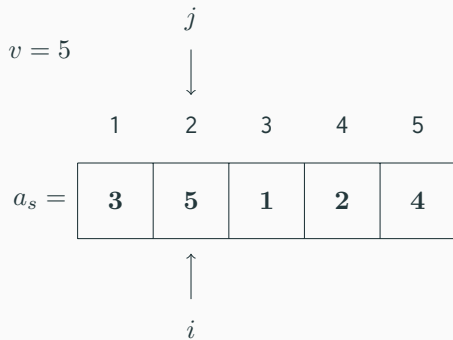
# Insertion Sort

---

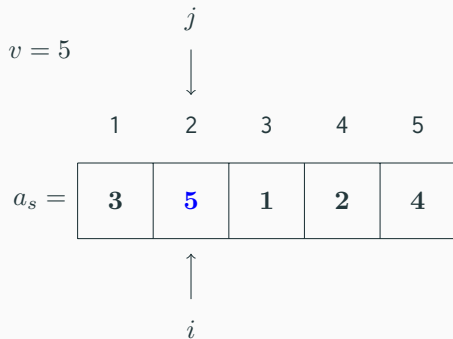
# Insertion Sort

- *Insertion sort* é um algoritmo de ordenação similar ao *selection sort*
- Ele também é estável, *in-place* e tem complexidade  $O(N^2)$
- Ele considera, inicialmente, que uma sequência com um único elemento já está ordenada
- Em seguida, para cada elemento da sequência, ele procura a posição correta no vetor ordenado que está à esquerda do elemento, e o insere nesta posição
- É o tipo de ordenação que os jogadores de cartas costumam usar

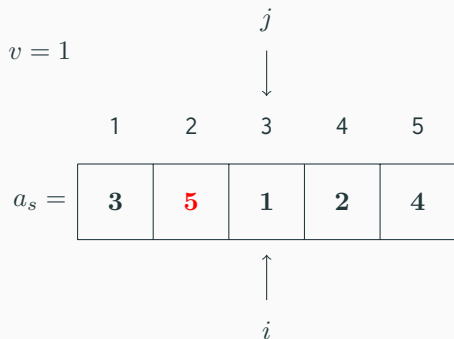
# Visualização do selection sort



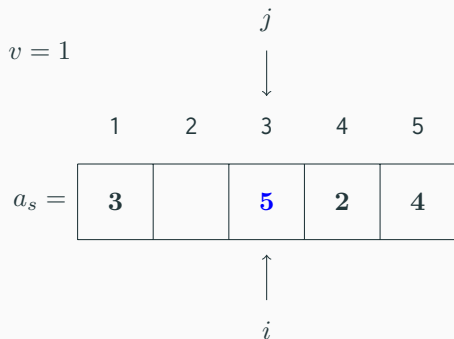
# Visualização do selection sort



# Visualização do selection sort

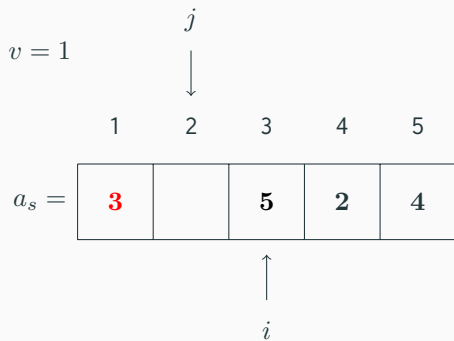


# Visualização do selection sort

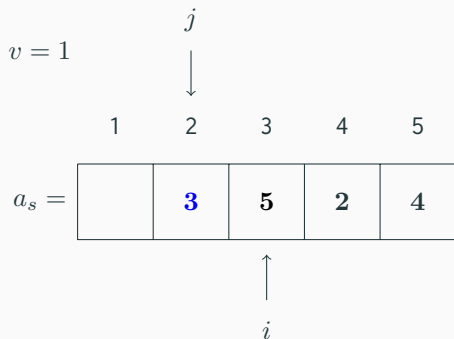




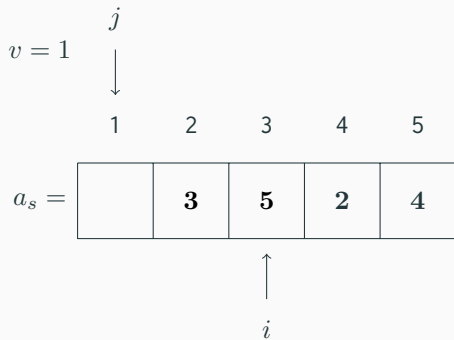
# Visualização do selection sort



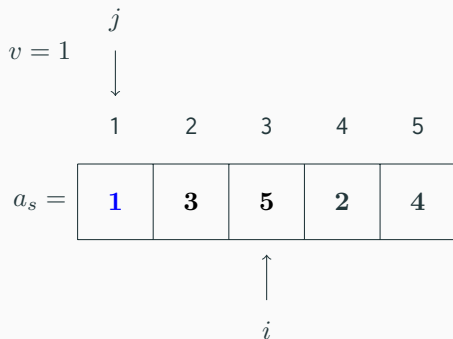
# Visualização do selection sort



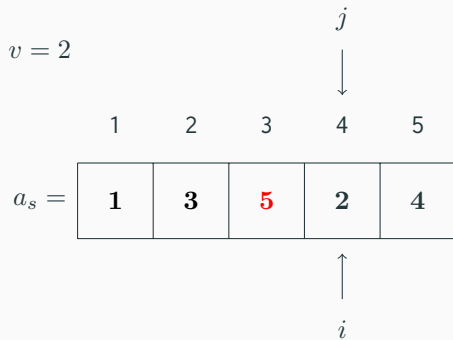
# Visualização do selection sort



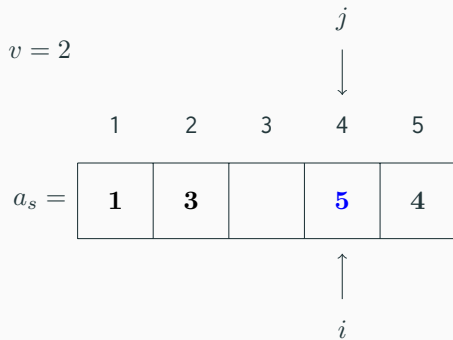
# Visualização do selection sort



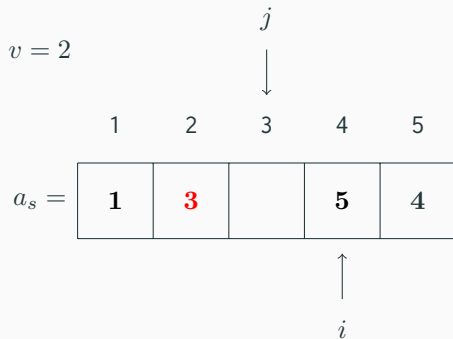
# Visualização do selection sort



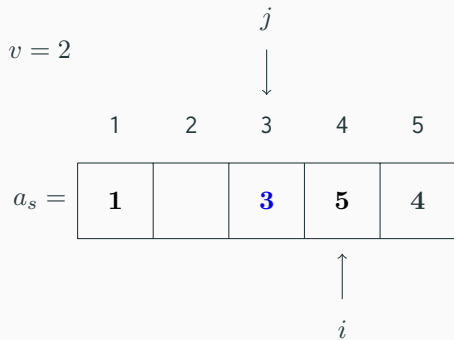
# Visualização do selection sort



# Visualização do selection sort

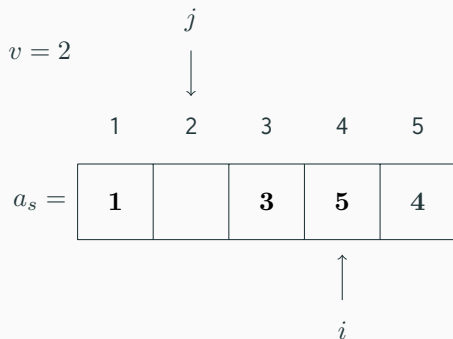


# Visualização do selection sort

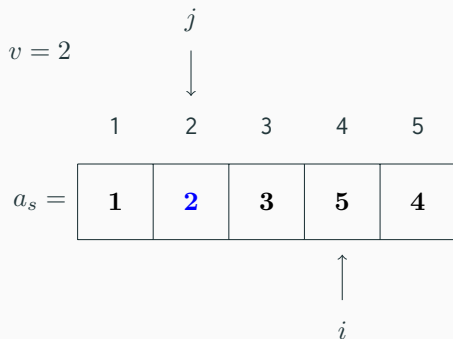




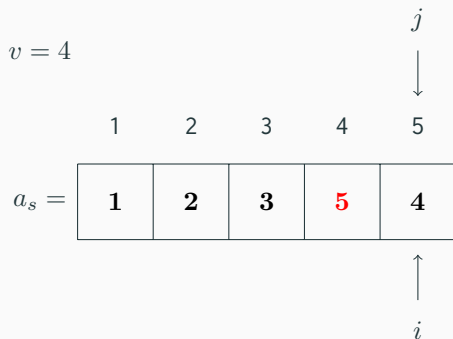
# Visualização do selection sort



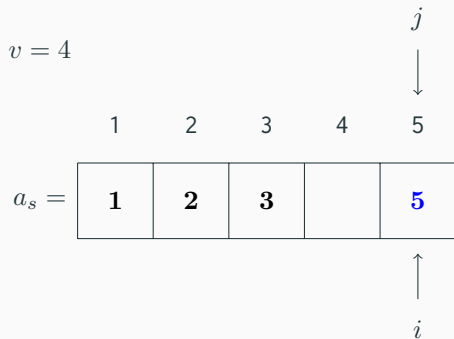
# Visualização do selection sort



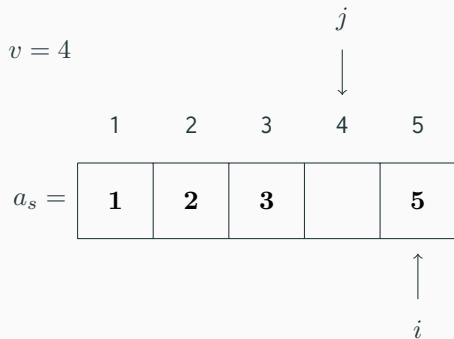
# Visualização do selection sort



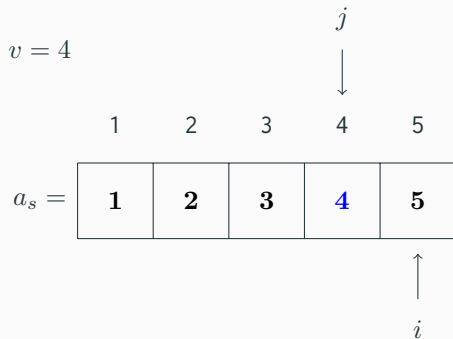
# Visualização do selection sort



# Visualização do selection sort



# Visualização do selection sort



# Implementação do insert sort em C++

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 template<typename T>
7 void insert_sort(vector<T>& as)
8 {
9     int N = as.size();
10
11     for (int i = 1, j; i < N; ++i)
12     {
13         auto v = as[i];
14
15         for (j = i; j and as[j - 1] > v; --j)
16             as[j] = as[j - 1];
17
18         as[j] = v;
19     }
20 }
21
```

# Implementação do insert sort em C++

```
22 int main()
23 {
24     vector<int> as { 3, 5, 1, 2, 4 };
25
26     insert_sort<int>(as);
27
28     for (size_t i = 0; i < as.size(); ++i)
29         cout << as[i] << (i + 1 == as.size() ? '\n' : ' ');
30
31     return 0;
32 }
```



# Bubble Sort

---

1. **DROZDEK**, Adam. *Algoritmos e Estruturas de Dados em C++*, 2002.
2. **KERNIGHAN**, Bryan; **RITCHIE**, Dennis. *The C Programming Language*, 1978.
3. **STROUSTROUP**, Bjarne. *The C++ Programming Language*, 2013.
4. **SWEDIG**, Robert. *Algorithms*, ???.
5. Wikipédia. *In-place algorithm*, acesso em 01/10/2018.<sup>1</sup>
6. Wikipédia. *Partially Ordered Set*, acesso em 01/10/2018.<sup>2</sup>

---

<sup>1</sup>[https://en.wikipedia.org/wiki/In-place\\_algorithm](https://en.wikipedia.org/wiki/In-place_algorithm)

<sup>2</sup>[https://en.wikipedia.org/wiki/Partially\\_ordered\\_set](https://en.wikipedia.org/wiki/Partially_ordered_set)