

”Para quem só sabe usar martelo, todo problema é um prego”
(Abraham Maslow).

Selection Sort

Paulo Ricardo Lisboa de Almeida

Selection Sort

Procure pelo menor elemento no vetor

i	1	2	3	4	5	6
v[i]	42	15	23	8	4	16

Selection Sort

Procure pelo menor elemento no vetor

Troque o elemento que está na primeira posição com o menor elemento

i	1	2	3	4	5	6
v[i]	4	15	23	8	42	16

Selection Sort

Procure pelo menor elemento no vetor

Troque o elemento que está na primeira posição com o menor elemento

A primeira posição do vetor está ordenada

i	1	2	3	4	5	6
v[i]	4	15	23	8	42	16

Selection Sort

Procure pelo menor elemento no vetor no vetor $[a+1..n]$



The diagram shows a table representing an array with indices i and values $v[i]$. The indices are 1 through 6, and the values are 4, 15, 23, 8, 42, and 16. A double-headed arrow is positioned above the table, spanning from index 1 to index 6, indicating the current range of the array being searched for the minimum element. The value 8 at index 4 is highlighted in blue, representing the current minimum element found in the range.

i	1	2	3	4	5	6
$v[i]$	4	15	23	8	42	16

Selection Sort

Procure pelo menor elemento no vetor no vetor $[a+1..n]$

Considerando o vetor $[a+1..n]$, troque o elemento que está na primeira posição com o menor elemento



A diagram illustrating a step in the Selection Sort algorithm. It shows a table with two rows: the first row represents indices i from 1 to 6, and the second row represents the values $v[i]$ at those indices. The values are 4, 8, 23, 15, 42, and 16. A double-headed arrow above the table spans from index 2 to index 6, indicating the current search range for the minimum element. The values 8, 15, and 16, along with their corresponding indices 2, 4, and 6, are highlighted in blue.

i	1	2	3	4	5	6
$v[i]$	4	8	23	15	42	16

Selection Sort

Procure pelo menor elemento no vetor no vetor $[a+1..n]$

Considerando o vetor $[a+1..n]$, troque o elemento que está na primeira posição com o menor elemento
o vetor $[a..2]$ está ordenado

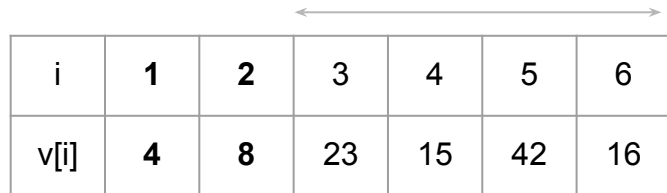
i	1	2	3	4	5	6
v[i]	4	8	23	15	42	16

Selection Sort

Procure pelo menor elemento no vetor no vetor $[a+1..n]$

Considerando o vetor $[a+1..n]$, troque o elemento que está na primeira posição com o menor elemento
o vetor $[a..2]$ está ordenado

Repita o processo enquanto o tamanho do vetor a ser ordenado for maior ou igual a 1



i	1	2	3	4	5	6
v[i]	4	8	23	15	42	16

Faça você mesmo

Tente criar uma versão do algoritmo você mesmo

Em português ou C - iterativo ou recursivo (a escolha é sua)

40 minutos

Selection Sort

função selectionSort (v,a,b)

entrada: vetor v, indexado por [a..b]

saída: o vetor v modificado de forma que $v[i] \leq v[j]$, $\forall i, j \in [a..b]$, e $i < j$.

se $a \geq b$

 retorne v

trocar(v, a, Mínimo(v, a, b))

retorne selectionSort(v,a+1,b)

Análise

Vamos realizar a análise considerando-se o número de comparações entre elementos do vetor

```
função selectionSort (v,a,b)  
se  $a \geq b$   
    retorne v  
trocar(v, a, Mínimo(v, a, b))  
retorne selectionSort(v,a+1,b)
```

Análise

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

```
função selectionSort (v,a,b)
se a ≥ b
    retorne v
trocar(v, a, Mínimo(v, a, b))
retorne selectionSort(v,a+1,b)
```

Análise

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$\begin{aligned} C(n) &= C(n-1) + C_m(n) = C(n-2) + C_m(n-1) + C_m(n) \\ &= C(n-3) + C_m(n-2) + C_m(n-1) + C_m(n) = \dots \end{aligned}$$

```
função selectionSort (v,a,b)  
se a ≥ b  
    retorne v  
trocar(v, a, Mínimo(v, a, b))  
retorne selectionSort(v,a+1,b)
```

Análise

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$\begin{aligned} C(n) &= C(n-1) + C_m(n) = C(n-2) + C_m(n-1) + C_m(n) \\ &= C(n-3) + C_m(n-2) + C_m(n-1) + C_m(n) = \dots \\ &= C(n-\mu) + \sum_{i=n-\mu+1}^n C_m(i) \end{aligned}$$

```
função selectionSort (v,a,b)  
se a ≥ b  
    retorne v  
trocar(v, a, Mínimo(v, a, b))  
retorne selectionSort(v,a+1,b)
```

Análise

Fazendo $n - \mu = 1$

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i)$$

Análise

C_m é o custo da busca pelo mínimo em um vetor de tamanho n .

Das aulas passadas, qual é esse custo?

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i)$$

Análise

C_m é o custo da busca pelo mínimo em um vetor de tamanho n .

Das aulas passadas, qual é esse custo?

$$C_m(n) = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i)$$

Análise

Temos melhor e pior caso?

$$C_m(n) = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i)$$

Análise

Temos melhor e pior caso?

Não. $C_m(n) = n-1$ sempre. E $C(n)$ também varia apenas de acordo com o tamanho do vetor.

$$C_m(n) = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i)$$

Análise

$$C_m(n) = n - 1$$

$$C(n) = \begin{cases} 0, & \text{se } n \leq 1, \\ C(n-1) + C_m(n), & \text{se } n > 1 \end{cases}$$

$$C(n) = \sum_{i=2}^n C_m(i) = \sum_{i=2}^n (n-1)$$

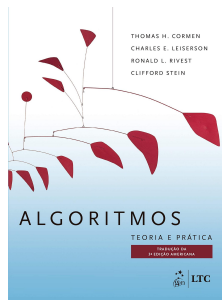
$$C(n) = \frac{(n-1)(1+n-1)}{2} = \frac{n^2 - n}{2} \approx \frac{n^2}{2}$$

Exercícios

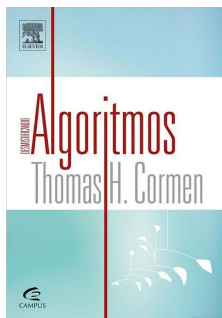
1. Implemente uma versão iterativa do algoritmo dado em aula
2. Qual o número de trocas entre elementos do vetor para o algoritmo selection sort?
 - a. Existe um melhor e um pior caso, ou o número de trocas depende exclusivamente do tamanho do vetor?
3. Compare as análises do *selection sort* com as do *insertion sort*. Existe um algoritmo que parece ser melhor? Ele é melhor em qualquer situação? Quando usar qual algoritmo?
4. Modifique o algoritmo dado em aula para que o número de trocas seja zero no melhor caso. A forma que você implementou impacta no número de comparações? Vale a pena realizar essa mudança? Quando?

Referências

T. Cormen, C. Leiserson,
R. Rivest, C. Stein.
Algoritmos: Teoria e
Prática. 3a ed. 2012



T. Cormen.
Desmistificando
algoritmos. 2017.

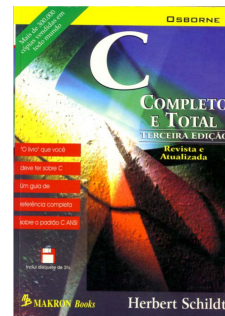


Renato Carmo. Algoritmos e
Estruturas de Dados.
www.inf.ufpr.br/renato

R. Sedgwick, K. Wayne.
Algorithms Part I. 4a ed.
2014



H. Schildt. C completo e
total. 1996



Licença

Este obra está licenciado com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).

