Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Algoritmos para Ordenação

Monael Pinheiro Ribeiro, D.Sc.

Algoritmos Estudados

- Bubble Sort
  - Consumo de Tempo no Pior Caso: $O(n^2)$
  - Consumo de Tempo no Melhor Caso: $O(n^2)$
- Selection Sort
  - Consumo de Tempo no Pior Caso: $O(n^2)$
  - Consumo de Tempo no Melhor Caso: $O(n^2)$
- Insertion Sort
  - Consumo de Tempo no Pior Caso: $O(n^2)$
  - Consumo de Tempo no Melhor Caso: $O(n)$
- Merge Sort
  - Consumo de Tempo no Pior Caso: $O(n \log_2 n)$
  - Consumo de Tempo no Melhor Caso: $O(n \log_2 n)$

Algoritmos Estudados

- Além disso, verificou-se que o lower bound do problema da ordenação é:

$$\Omega(n \log_2 n)$$

Particionamento

- Problema do Particionamento
  - Dado um vetor **v** de **n** posições e um índice **p** qualquer.
  - Desenvolva um procedimento que garanta que todos os elementos com índice menores que **p** são menores ou iguais a **v**[**p**] e todos os elementos com índice maiores que **p** são maiores que v[**p**]

$$\mathbf{v}[0, \ldots, \mathbf{p}-1] \leq \mathbf{v}[\mathbf{p}] < \mathbf{v}[\mathbf{p}+1, \ldots, \mathbf{n}-1]$$

Particionamento

- Problema do Particionamento
  - Exemplo:

Entrada: V = [5,7,56,97,2,3,67,68,32,98,89,34]
p = 11

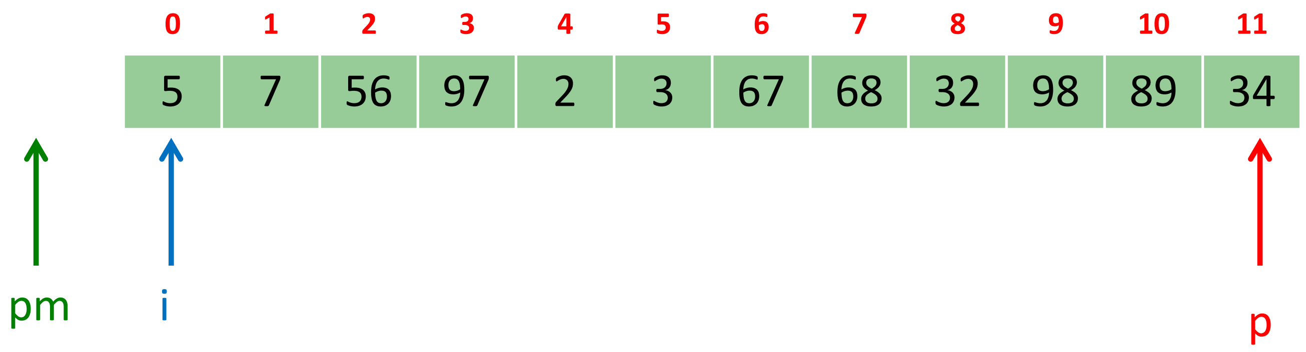
Portanto, V[p] = 34

Saída: V = [5,7,2,3,32,**34**,67,68,56,98,89,97]

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

Particionamento



i=0

p=ultimo elemento do vetor

pm = -1

enquanto i < n-1 : Verdade

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento

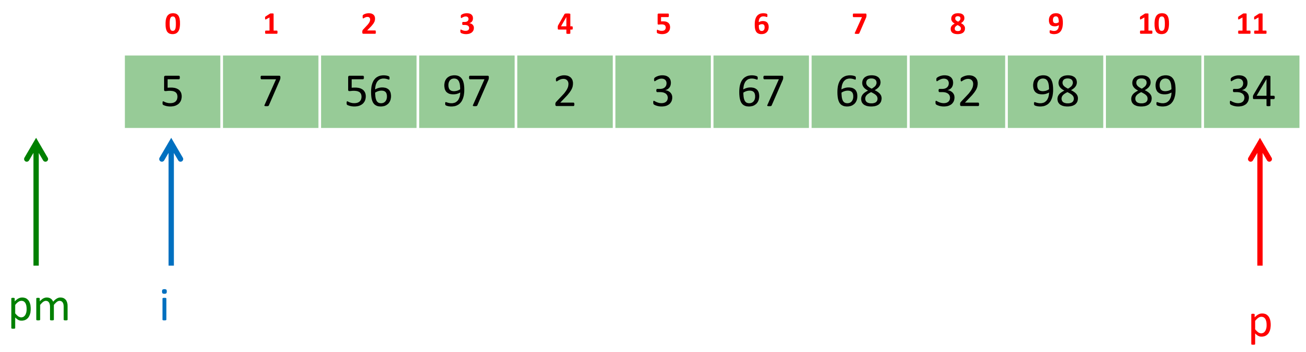
0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento

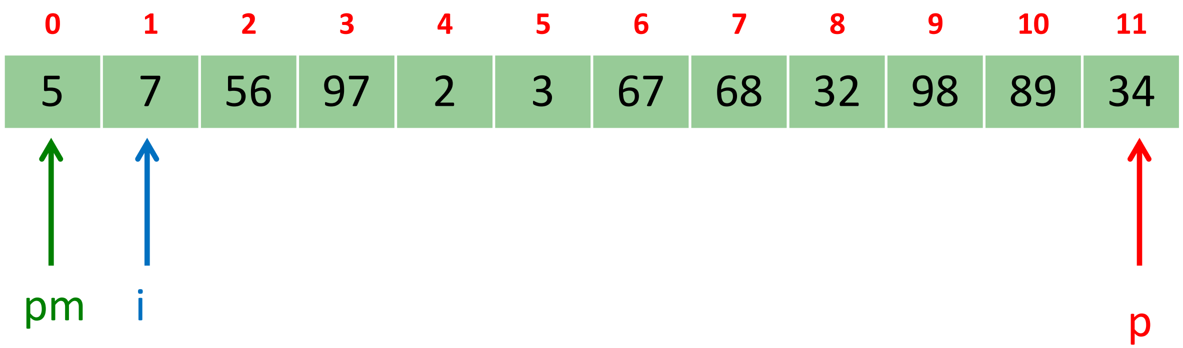
0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim
Então: pm = pm + 1 e troca v[i] com v[pm]
i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

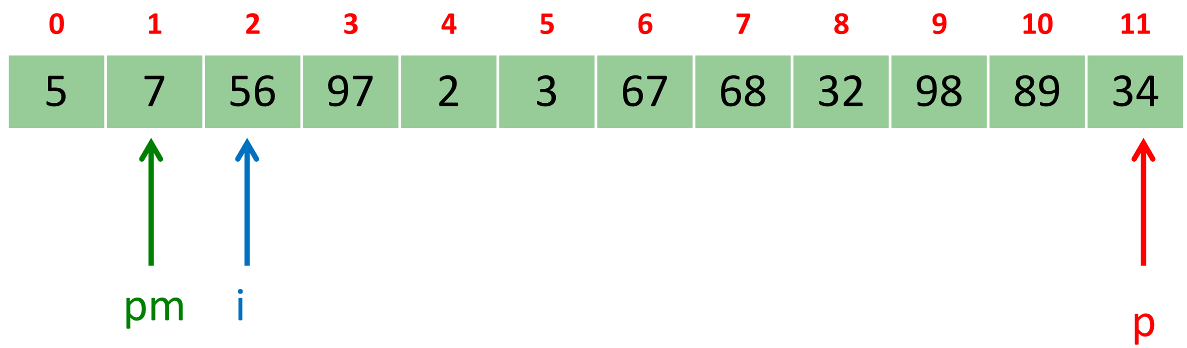
Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34
pm	i										p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim
Então: pm = pm + 1 e troca v[i] com v[pm]
i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim
Então: pm = pm + 1 e troca v[i] com v[pm]
i = i + 1

Particionamento

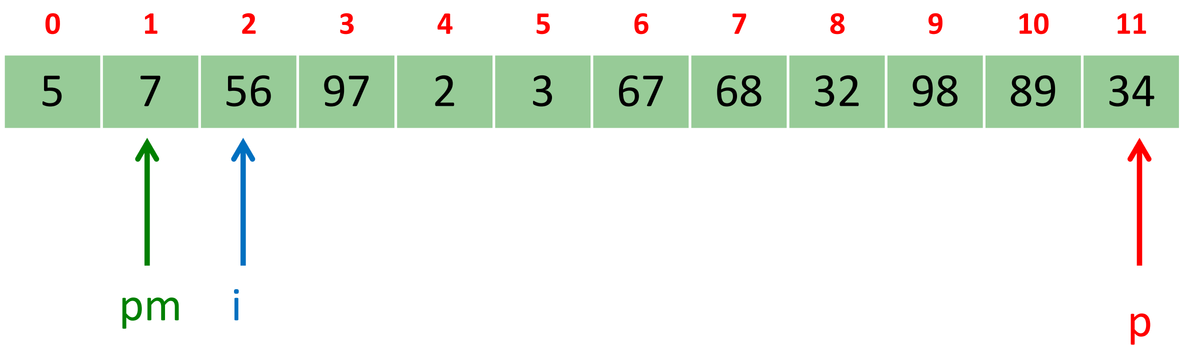
0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

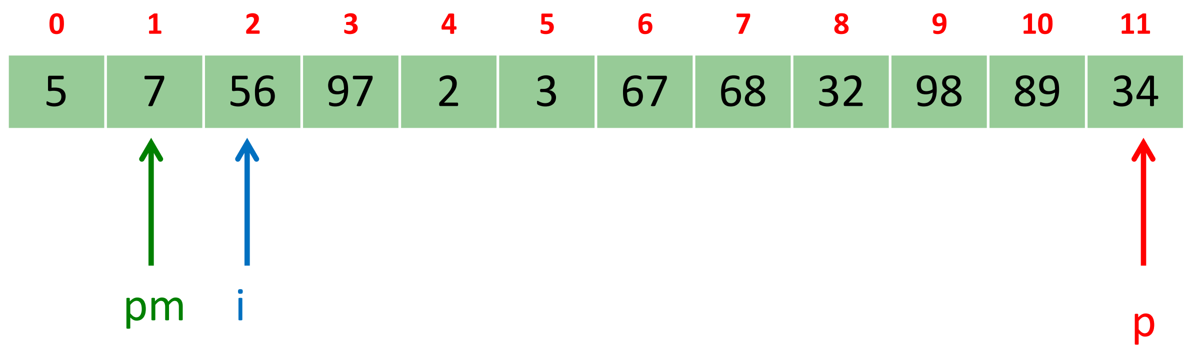
Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

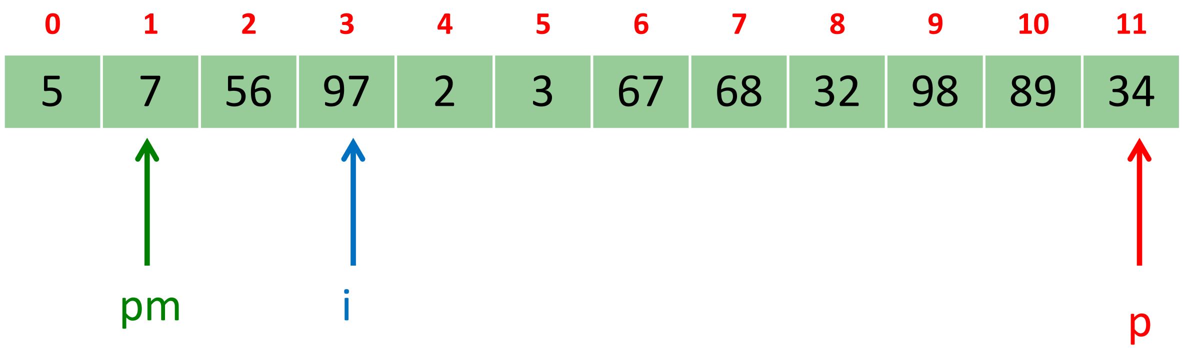
0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

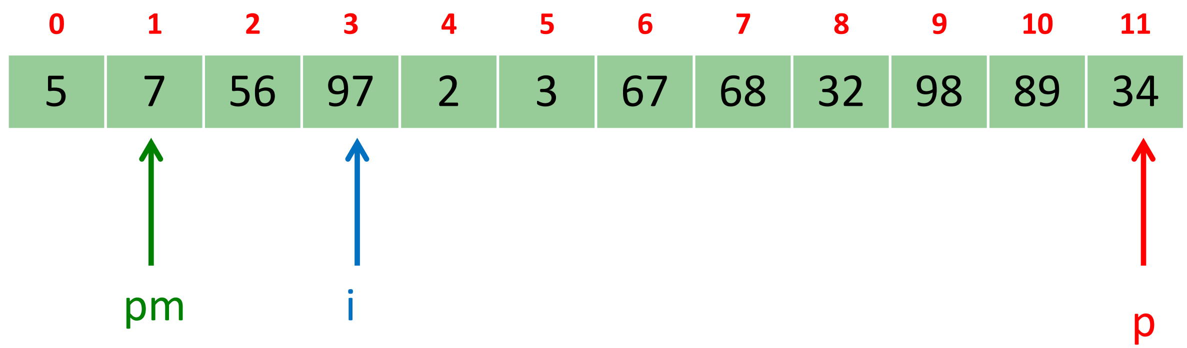
Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ?

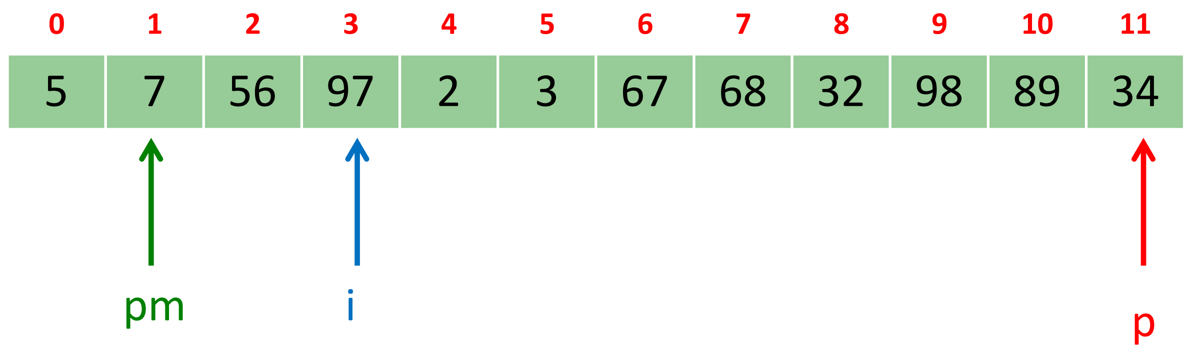
Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

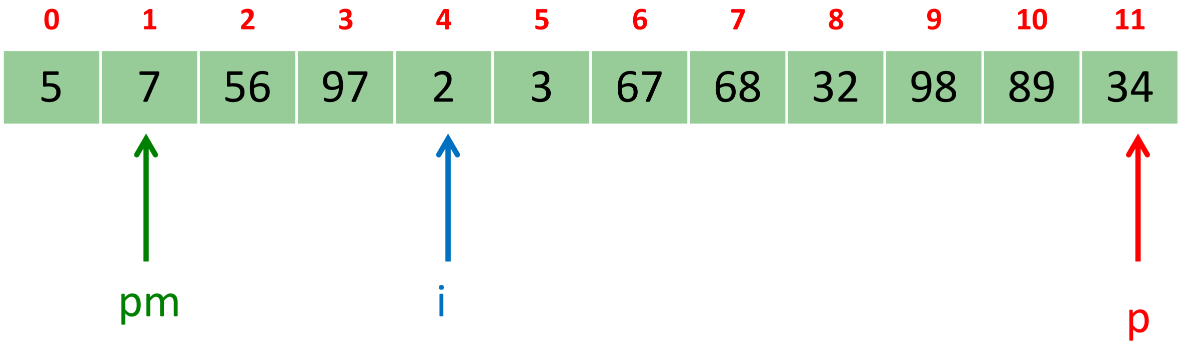
0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34
	pm			i							p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Então: i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	56	97	2	3	67	68	32	98	89	34

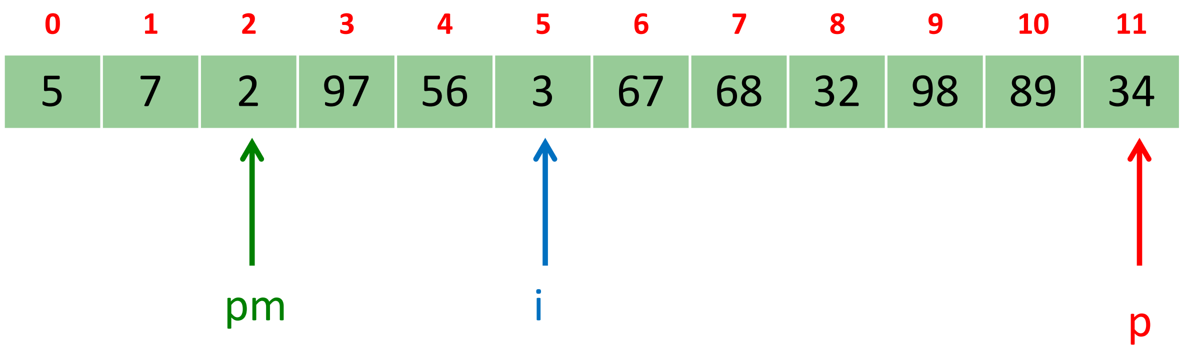
enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento



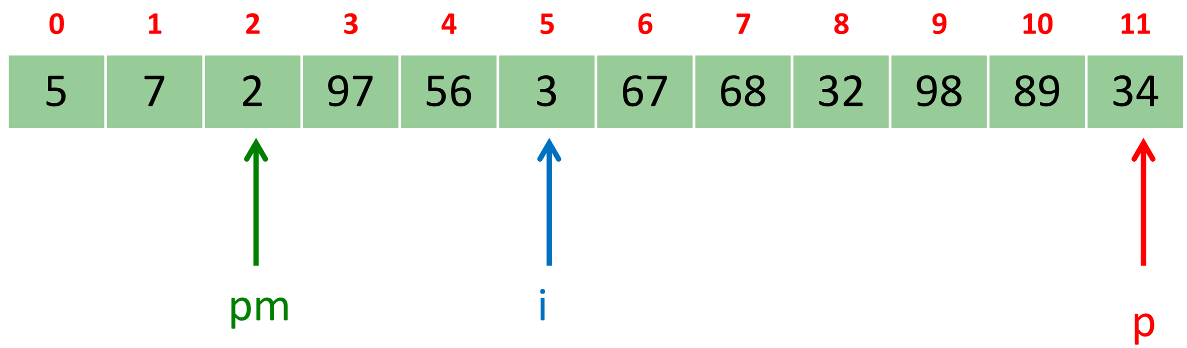
enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ?

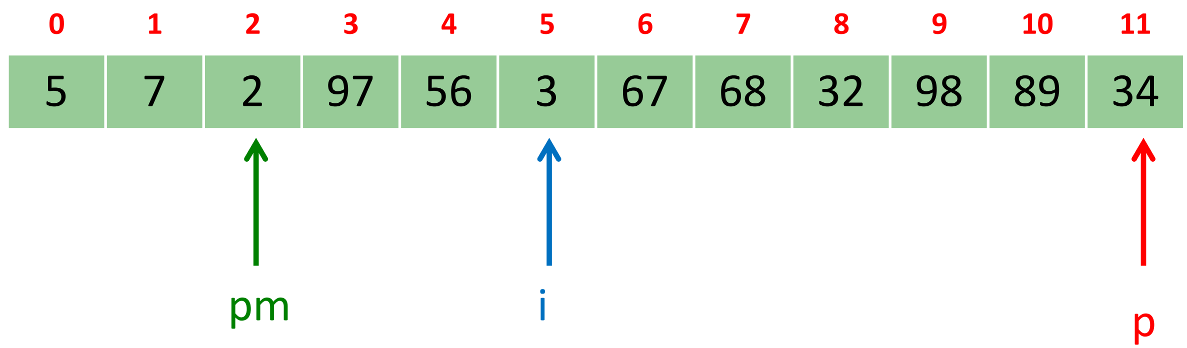
Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	97	56	3	67	68	32	98	89	34
		pm			i						p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34
			pm			i					p

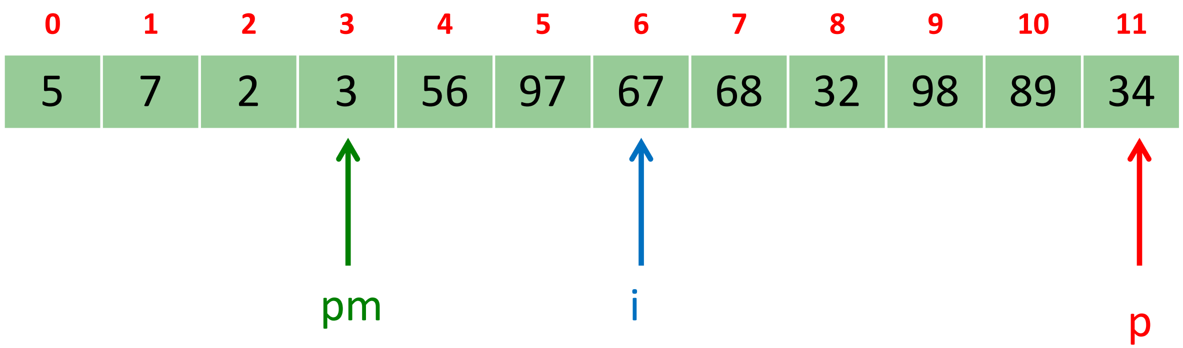
enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Particionamento

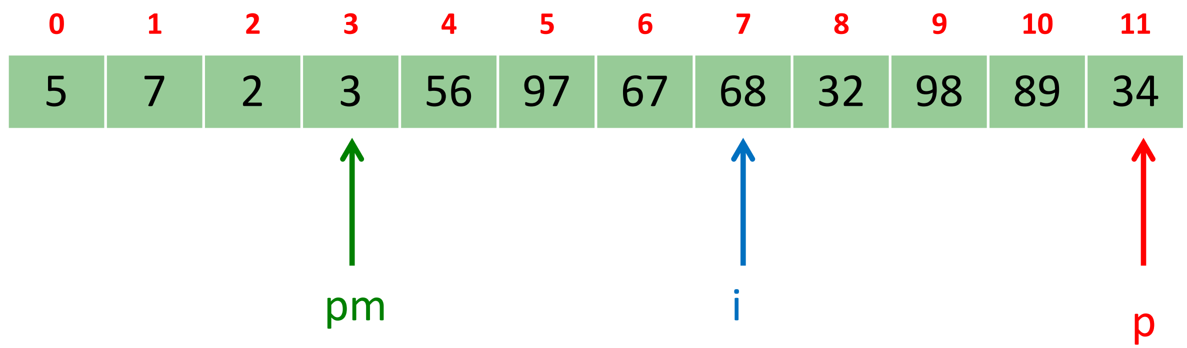
0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34
			pm				i				p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

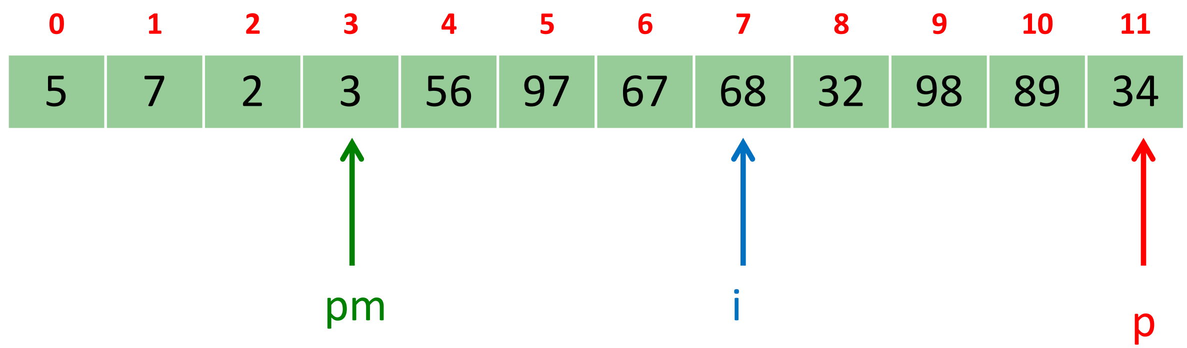
Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34
			pm				i				p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34

pm

i

p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

Então: i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34
			pm					i			p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

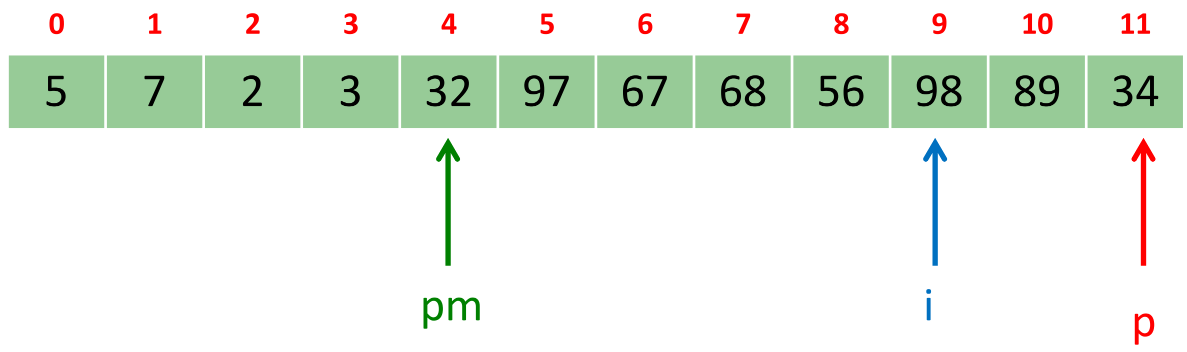
Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	56	97	67	68	32	98	89	34

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim
Então: pm = pm + 1 e troca v[i] com v[pm]
i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Sim

Então: pm = pm + 1 e troca v[i] com v[pm]

i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	32	97	67	68	56	98	89	34
				pm					i		p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento

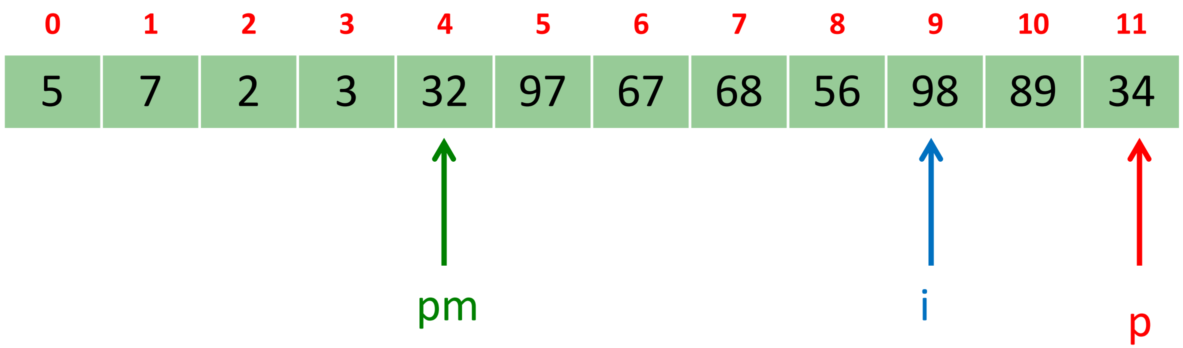
0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	32	97	67	68	56	98	89	34

pm i p

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

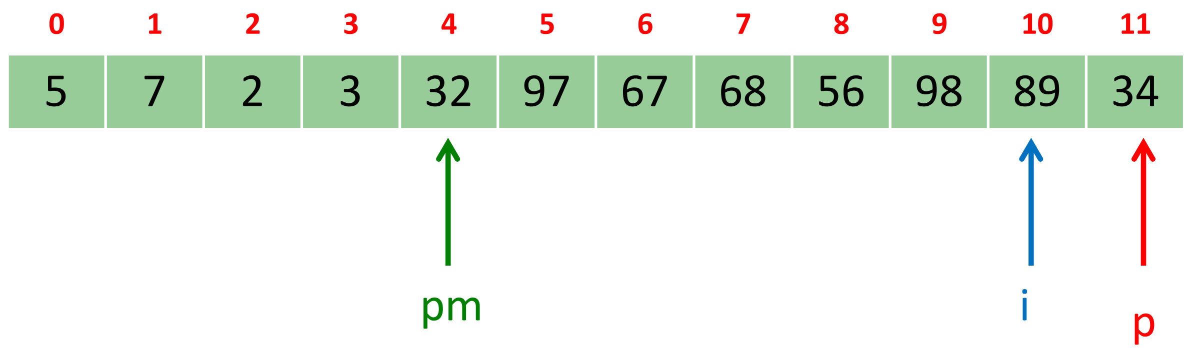
Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	32	97	67	68	56	98	89	34

pm

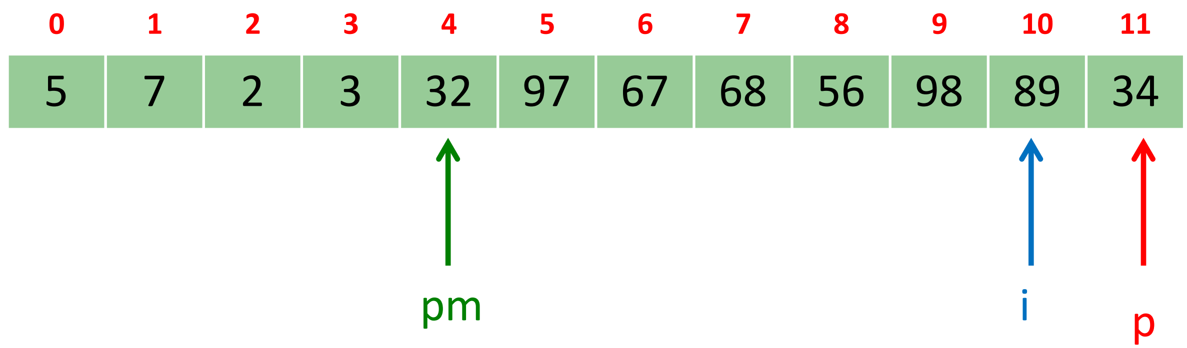
i

p

enquanto i < n-1 : Verdade

v[i] <= v[p] ?

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não

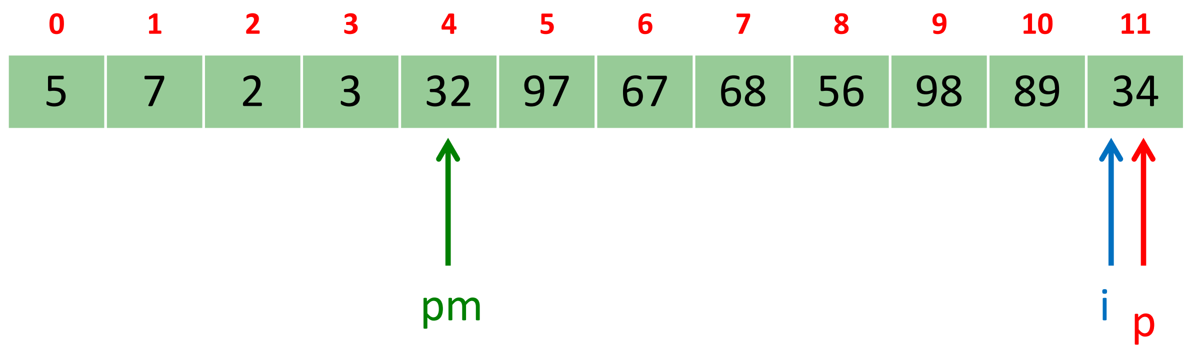
Particionamento

0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	32	97	67	68	56	98	89	34

enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento



enquanto i < n-1 : Verdade

v[i] <= v[p] ? Não
Então: i = i + 1

Particionamento

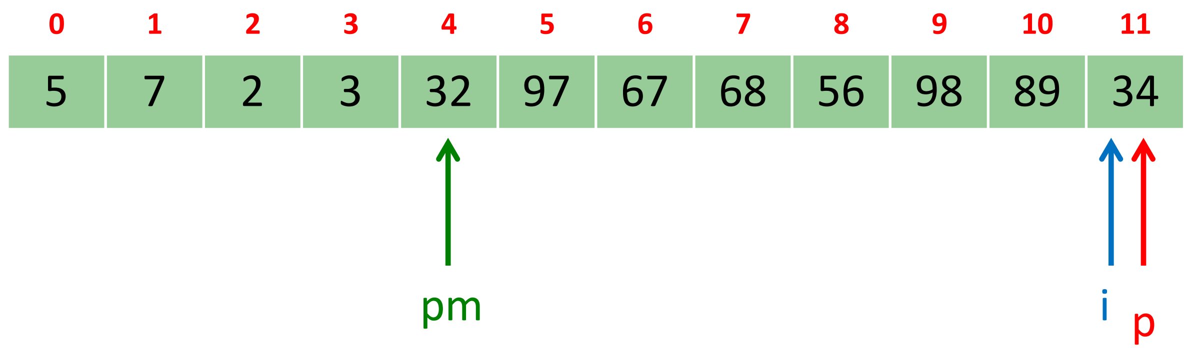
0	1	2	3	4	5	6	7	8	9	10	11
5	7	2	3	32	97	67	68	56	98	89	34

pm

i p

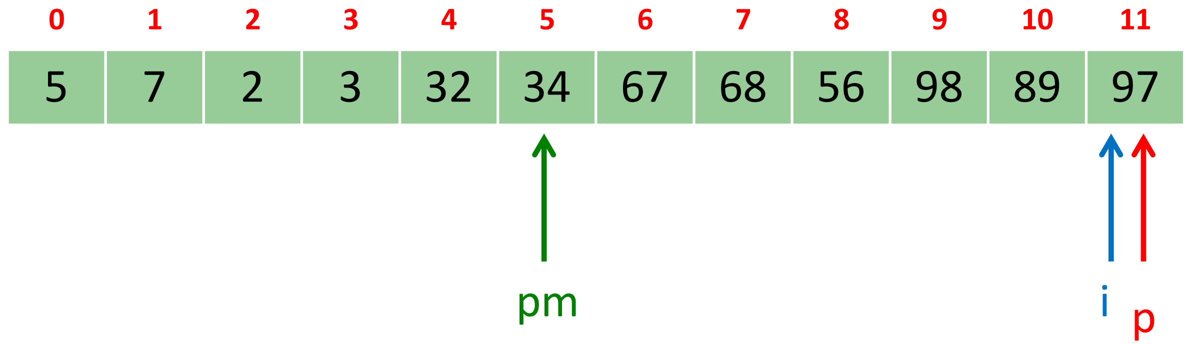
enquanto i < n-1 : Falso

Particionamento



enquanto i < n-1 : Falso
pm = pm + 1
troca v[p] com v[pm]
retorne pm

Particionamento



enquanto i < n-1 : Falso
pm = pm + 1
troca v[p] com v[pm]
retorne pm

Particionamento

```
int particiona(int *v, int n)  {
    int pm=-1, i, aux;
    for(i=0; i<n-1; i++)
    {
        if(v[i]<=v[n-1])
        {
            pm++;
            aux = v[pm];
            v[pm] = v[i];
            v[i] = aux;
        }
    }
    aux = v[pm+1];
    v[pm+1] = v[n-1];
    v[n-1] = aux;
    return pm+1;
}
```

Consumo de Tempo?

Particionamento

```c
int particiona(int *v, int n)  {
    int pm=-1, i, aux;
    for(i=0; i<n-1; i++)
    {
        if(v[i]<=v[n-1])
        {
            pm++;
            aux = v[pm];
            v[pm] = v[i];
            v[i] = aux;
        }
    }
    aux = v[pm+1];
    v[pm+1] = v[n-1];
    v[n-1] = aux;
    return pm+1;
}
```

Consumo de Tempo?

O(n)

Particionamento

- Dado o vetor v antes e depois do particionamento, o que pode ser observado?

Entrada: **v** = [05,07,56,97,02,03,67,68,32,98,89,34]

Saída: **v** = [05,07,02,03,32,**34**,67,68,56,98,89,97]

Particionamento

- Dado o vetor v antes e depois do particionamento, o que pode ser observado?

Entrada: **v** = [05,07,56,97,02,03,67,68,32,98,89,34]

Saída: **v** = [05,07,02,03,32,**34**,67,68,56,98,89,97]

Ordenado: **v** = [02,03,05,07,32,**34**,56,67,68,89,97,98]

Particionamento

- Dado o vetor v antes e depois do particionamento, o que pode ser observado?

Entrada: **v** = [05,07,56,97,02,03,67,68,32,98,89,34]
Saída: **v** = [05,07,02,03,32,**34**,67,68,56,98,89,97]
Ordenado: **v** = [02,03,05,07,32,**34**,56,67,68,89,97,98]

- É garantido que o elemento **p** ficou na sua posição correta do vetor ordenado.

Particionamento

- Dado o vetor v antes e depois do particionamento, o que pode ser observado?

Entrada: **v** = [05,07,56,97,02,03,67,68,32,98,89,34]
Saída: **v** = [05,07,02,03,32,**34**,67,68,56,98,89,97]
Ordenado: **v** = [02,03,05,07,32,**34**,56,67,68,89,97,98]

- É garantido que o elemento **p** ficou na sua posição correta do vetor ordenado.
- O que acontece se aplicar o procedimento recursivamente nas porções separadas por **p** ?

Particionamento

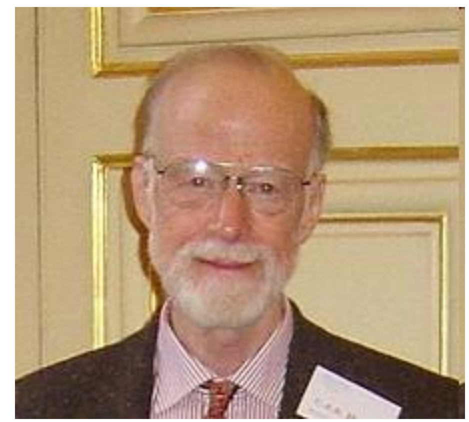
- Dado o vetor v antes e depois do particionamento, o que pode ser observado?

Entrada: **v** = [05,07,56,97,02,03,67,68,32,98,89,34]
Saída: **v** = [05,07,02,03,32,**34**,67,68,56,98,89,97]
Ordenado: **v** = [02,03,05,07,32,**34**,56,67,68,89,97,98]

- É garantido que o elemento **p** ficou na sua posição correta do vetor ordenado.
- O que acontece se aplicar o procedimento recursivamente nas porções separadas por **p** ?
- Esse é o **Quick Sort** !

Quick Sort

- Proposto em 1962 por Charles Antony Richard Hoare no Computer Journal, 5, pp.10-15, 1962
  – É considerado o método de ordenação mais eficiente até os dias atuais;
  – Emprega a Divisão e Conquista;
  – O método consiste em:
    - Eleger um pivô
    - Garantir que todos os elementos a direita do pivô são menores ou iguais que ele e a esquerda são maiores
    - Repetir recursivamente na metade direita e na metade esquerda do arranjo (usando como referencia o pivô).



Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

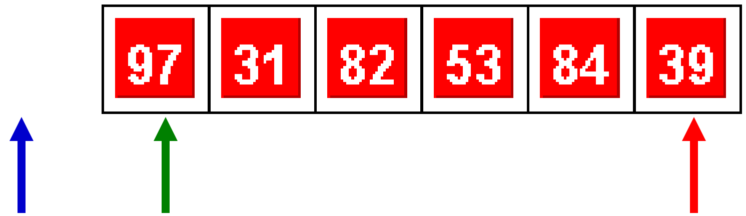
97	31	82	53	84	39

Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

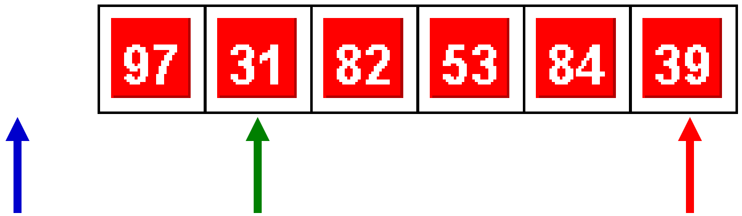


Quick Sort

Idéia

Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

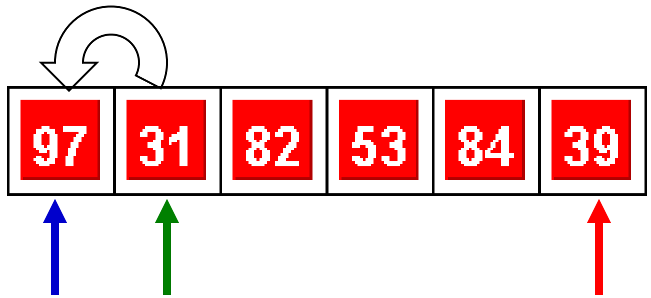


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia

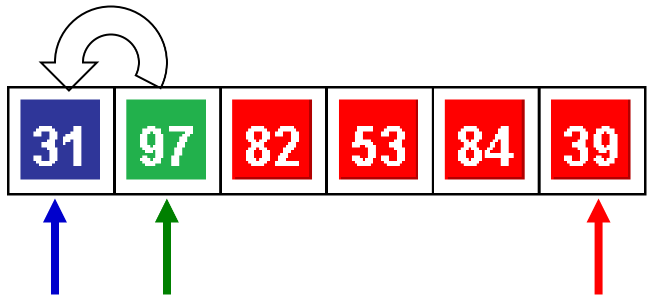


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

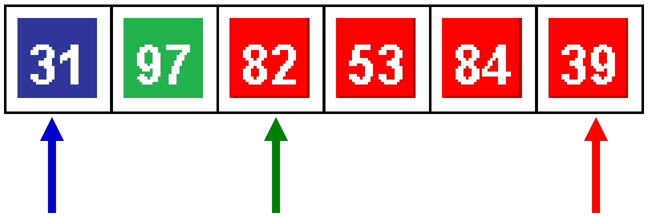
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca

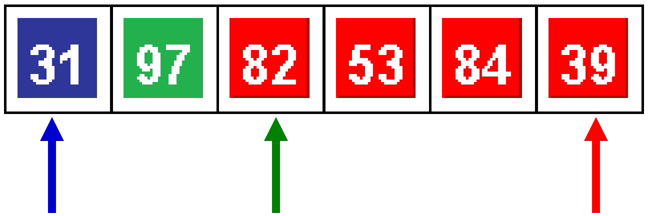


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

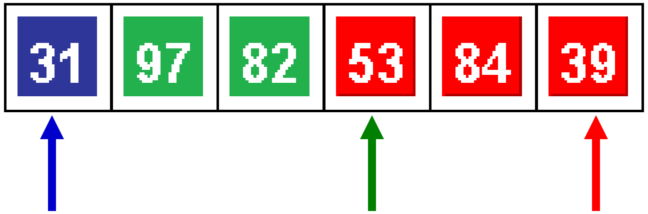


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

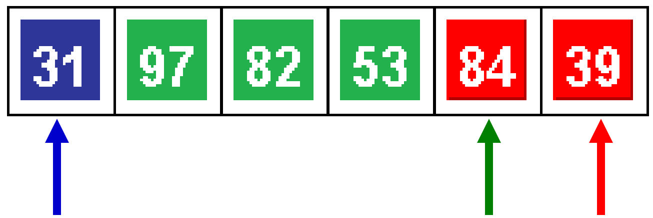


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

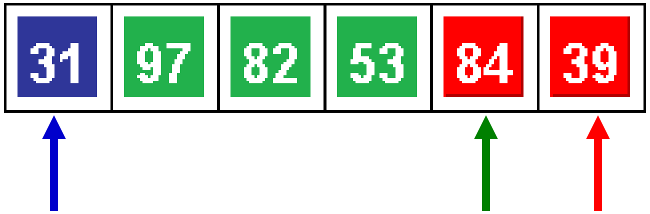


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

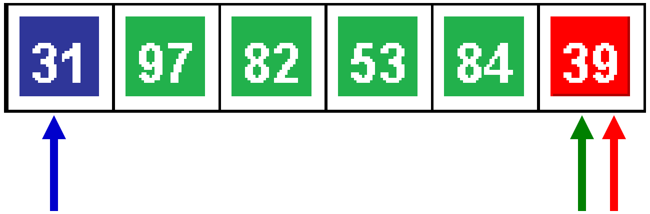


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

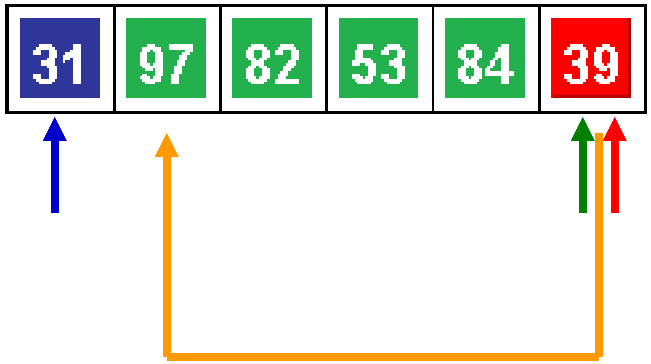


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

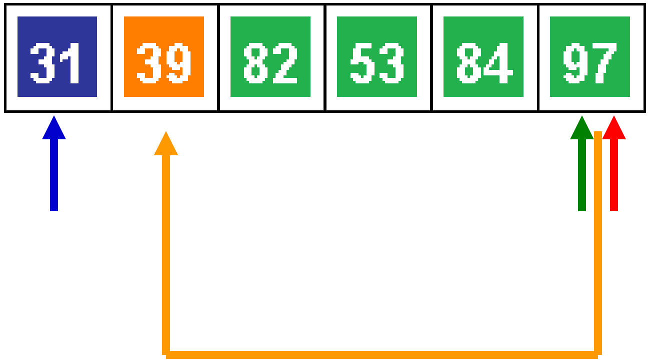
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca



Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

Quick Sort

Idéia

31	39	82	53	84	97

Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

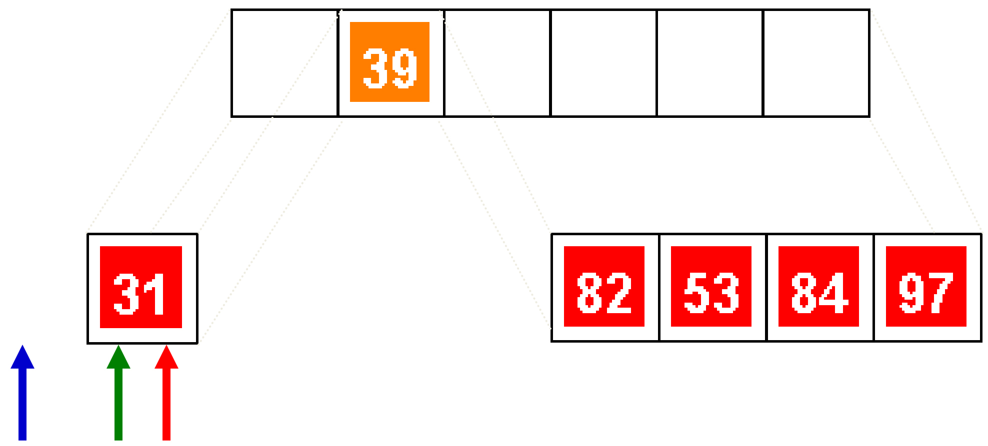
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca

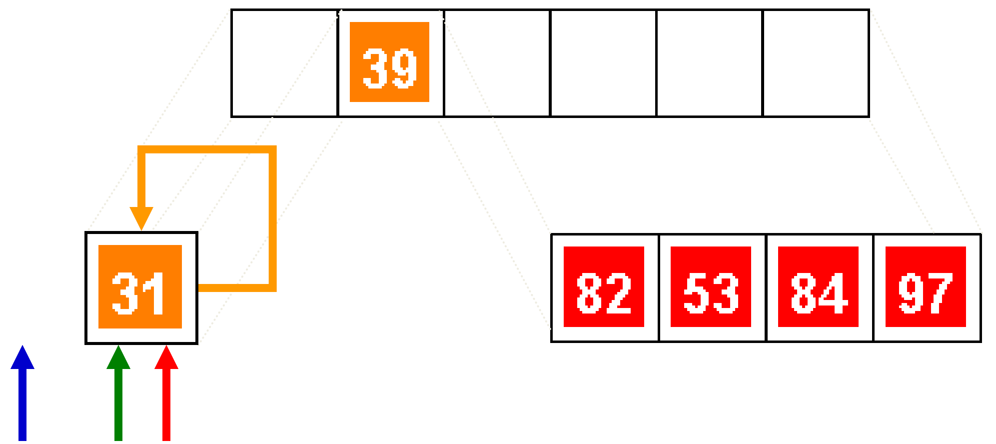


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia

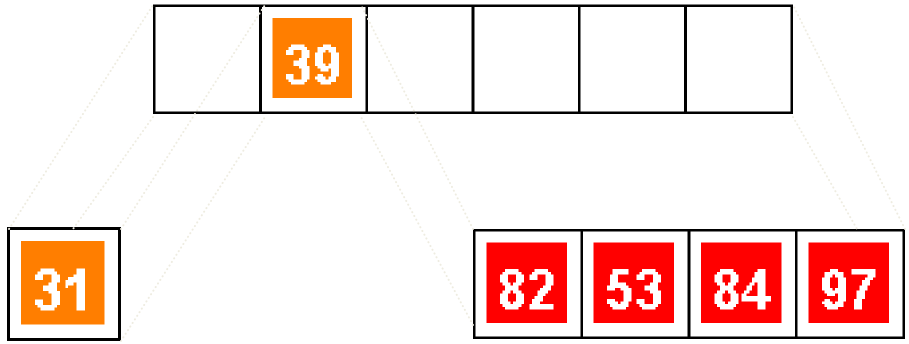


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

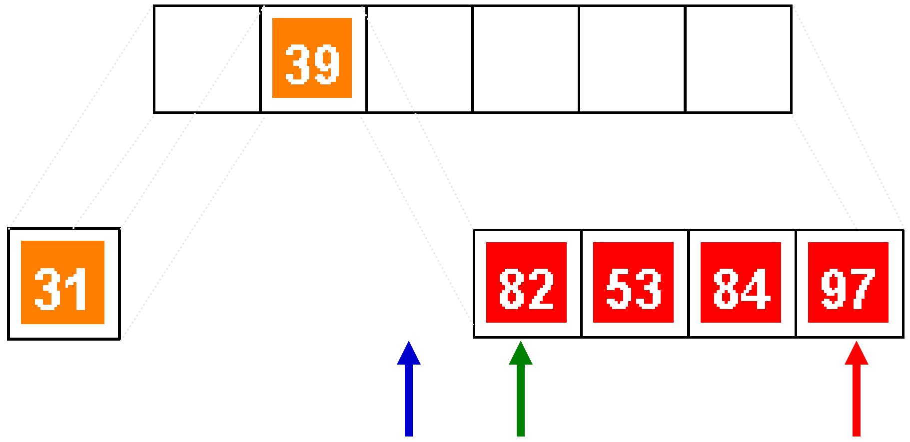
Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Quick Sort

Idéia

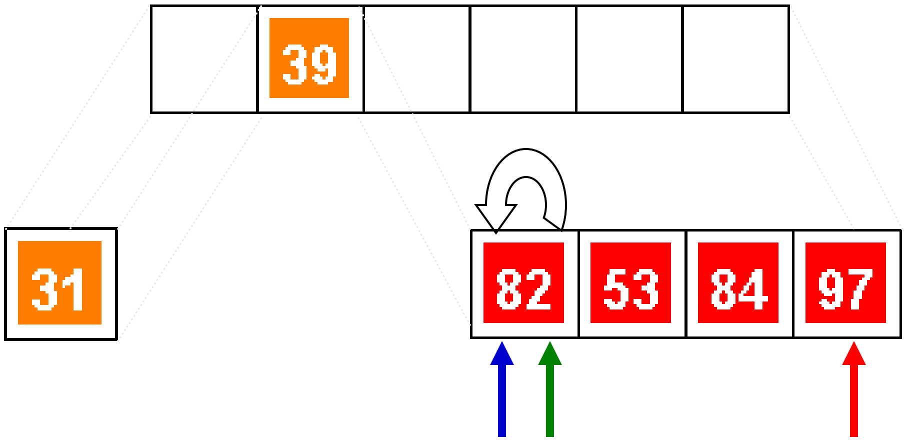


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

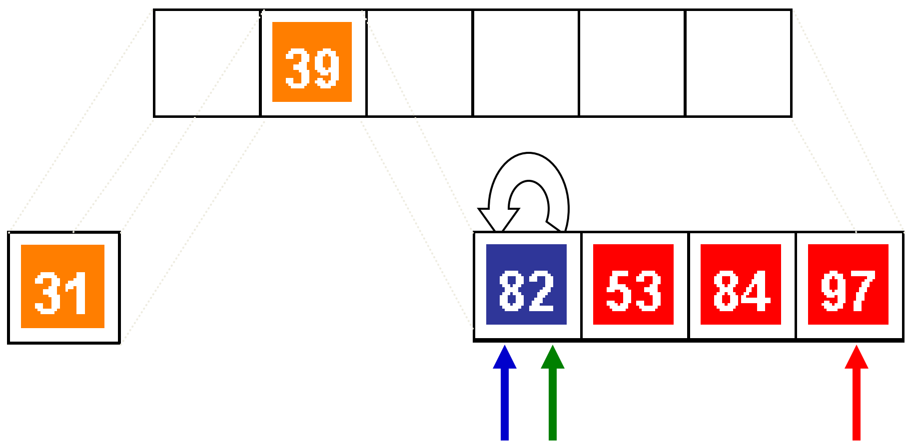


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

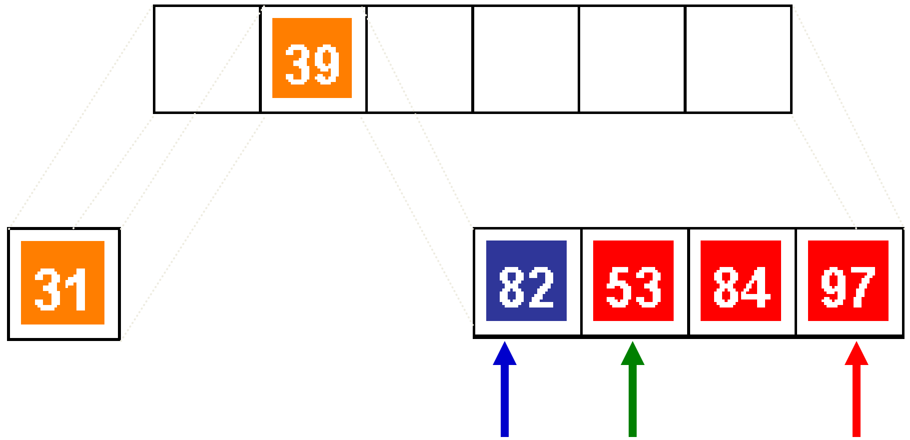


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

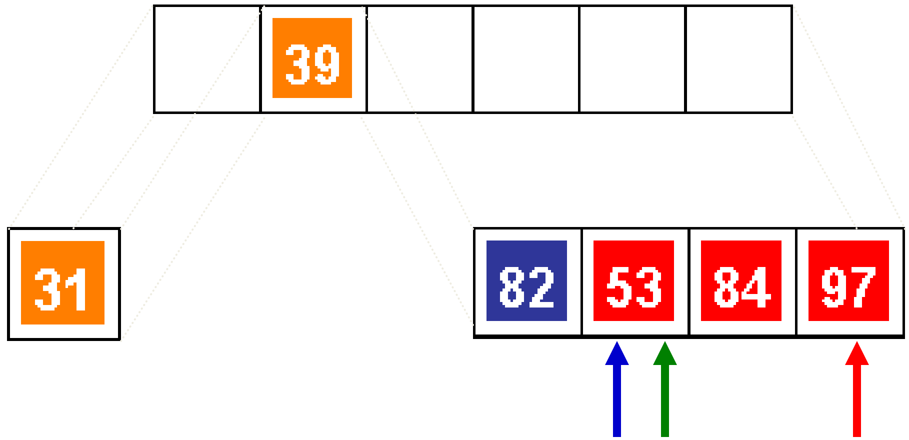


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

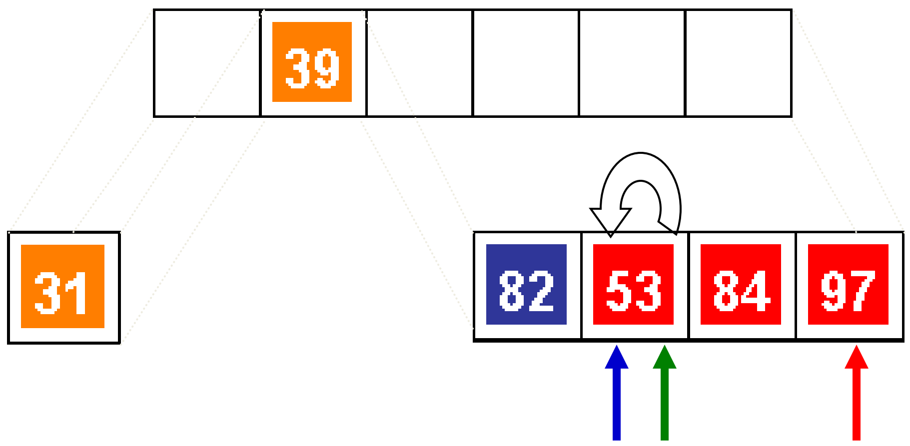


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

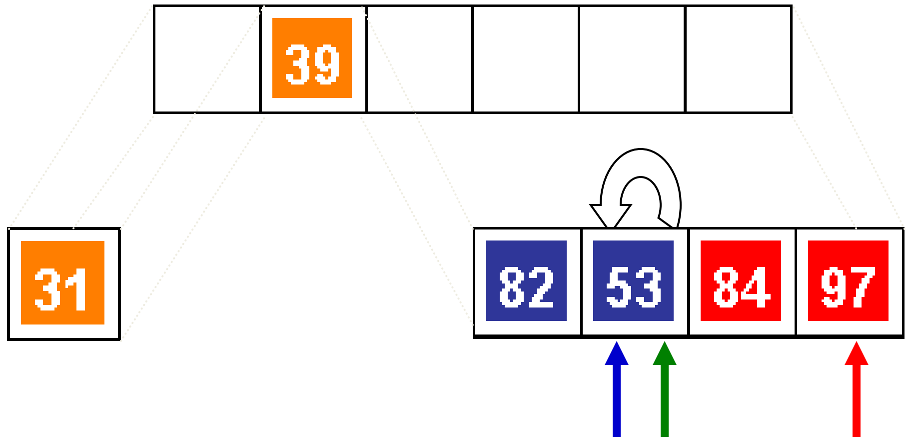
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca

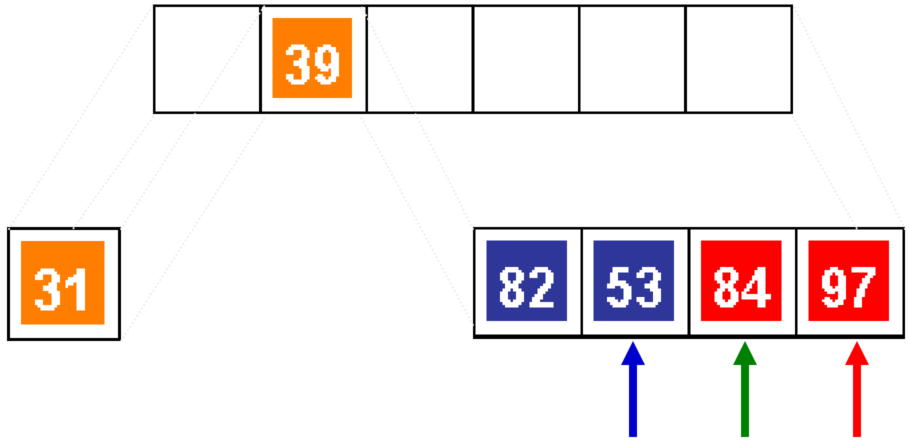


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

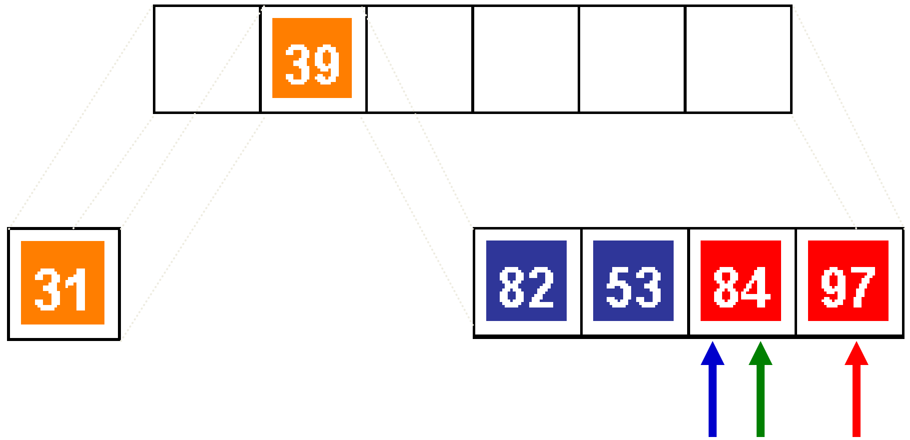
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca

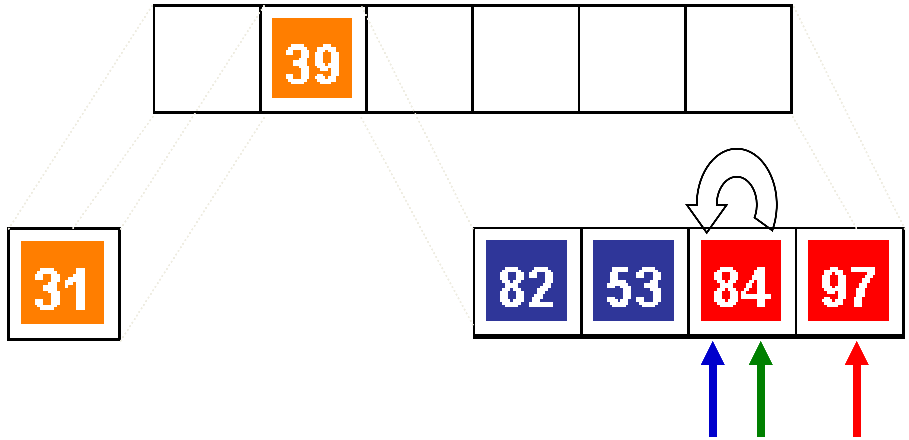


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

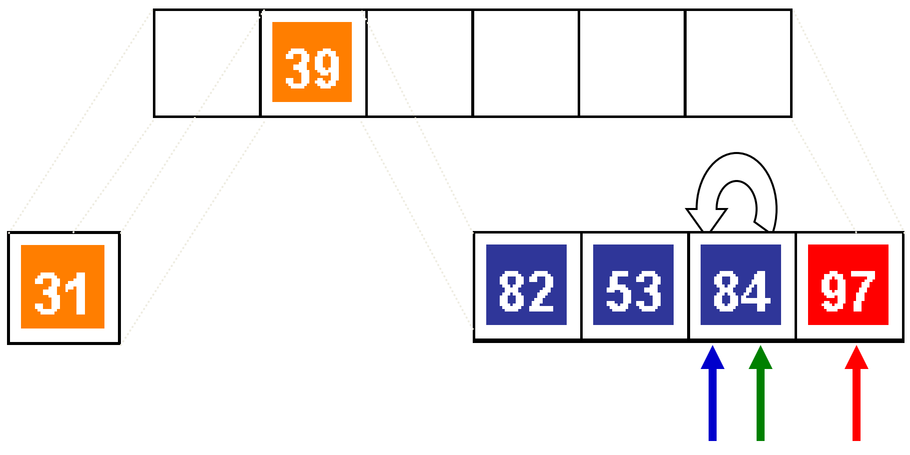


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

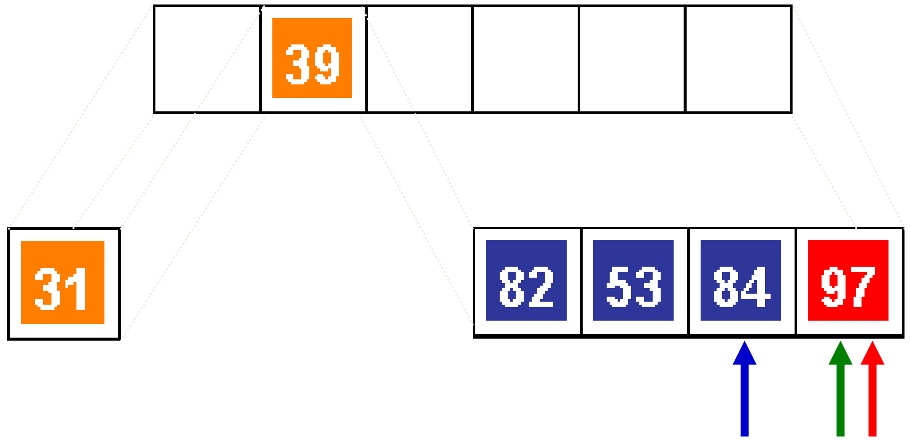


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

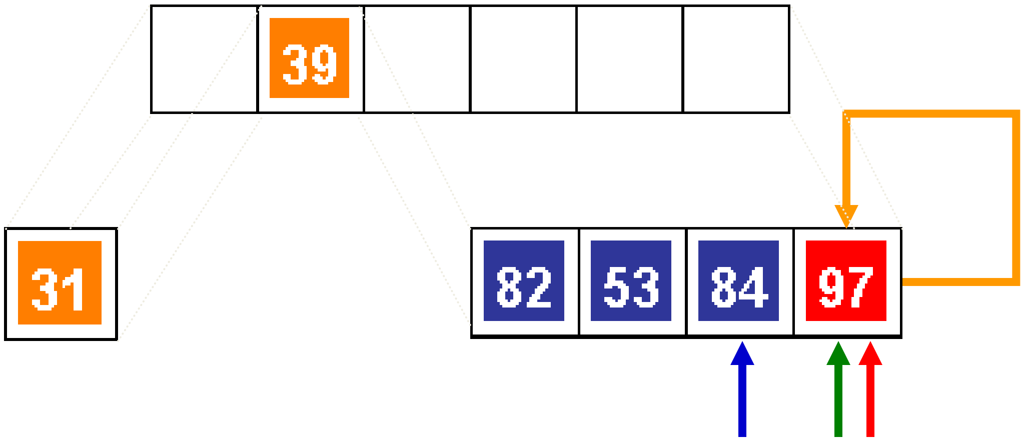
Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Quick Sort

Idéia



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

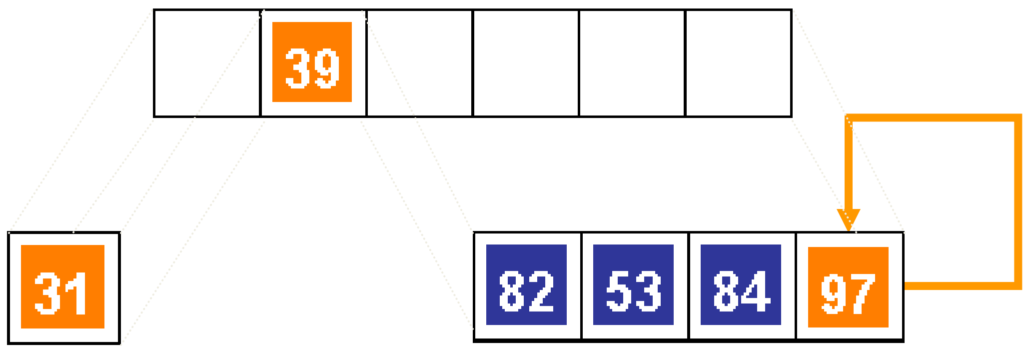
Elemento pivô

Último menor elementos

Elemento atual

Troca

Idéia

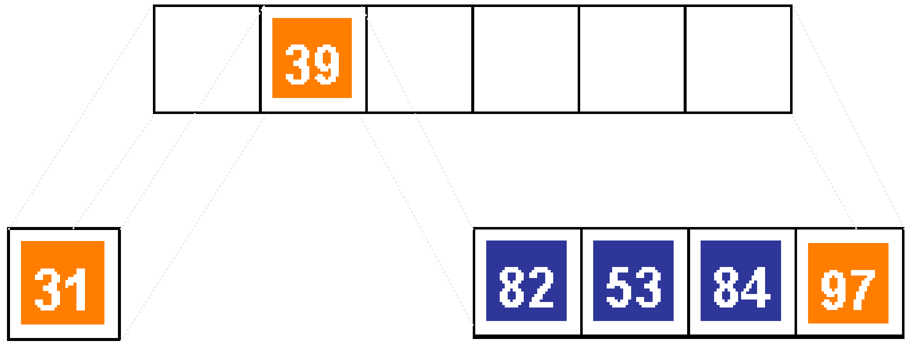


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia

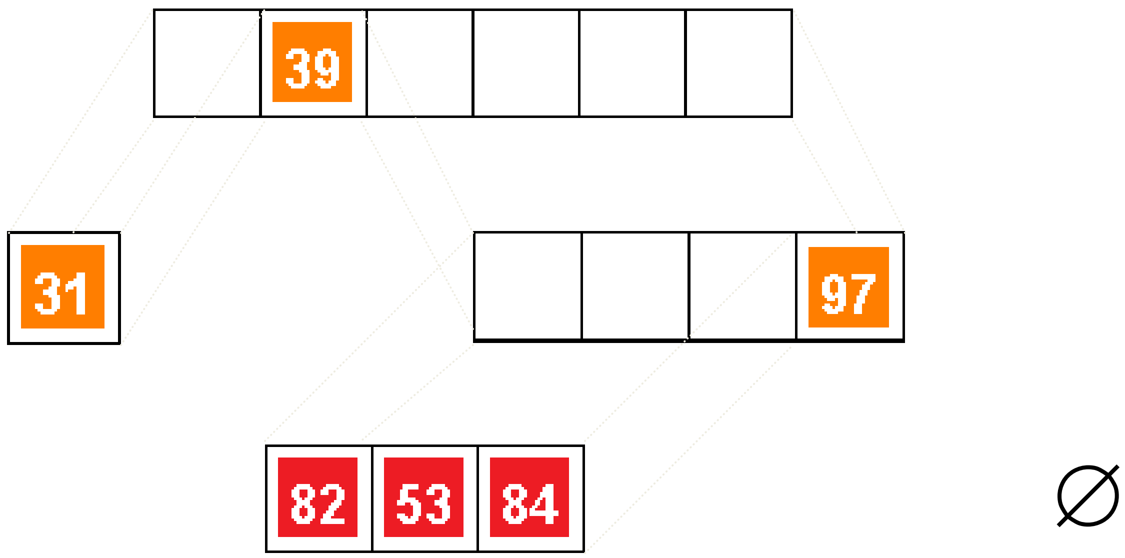


Quick Sort

Idéia

Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

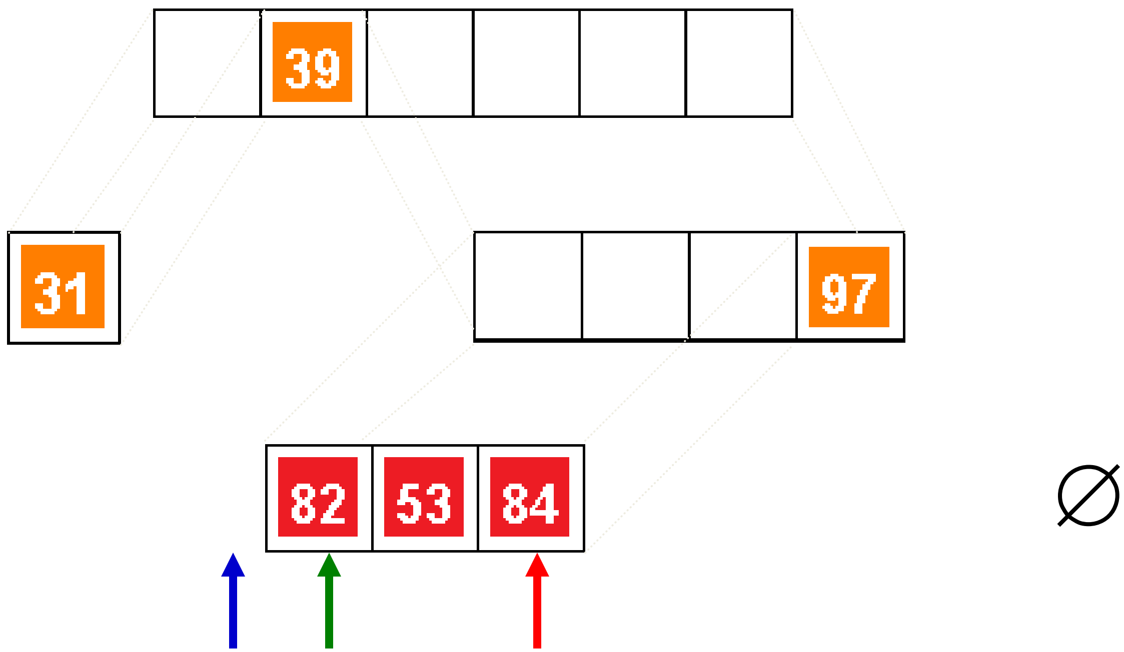


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

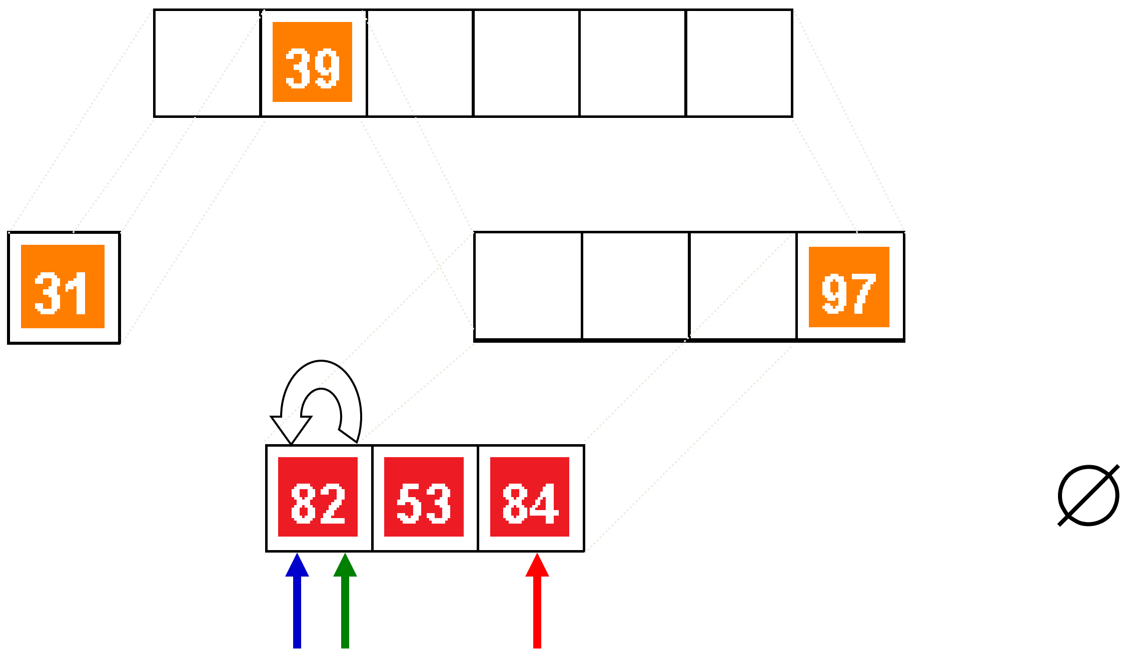


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

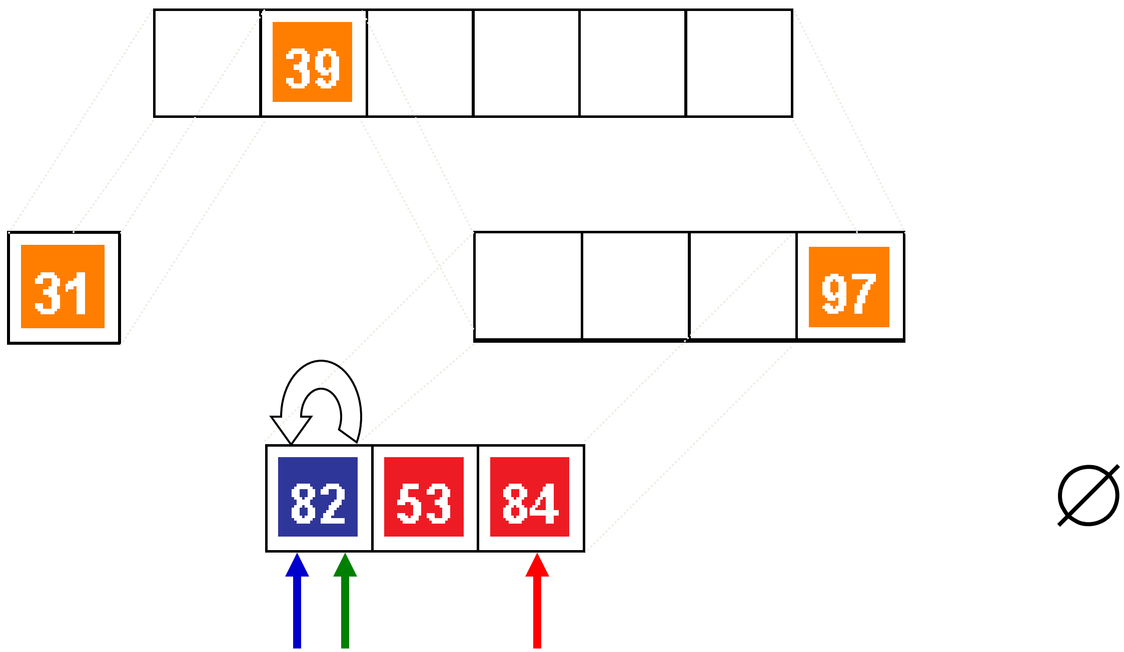
Idéia

Elemento pivô

Último menor elementos

Elemento atual

Troca

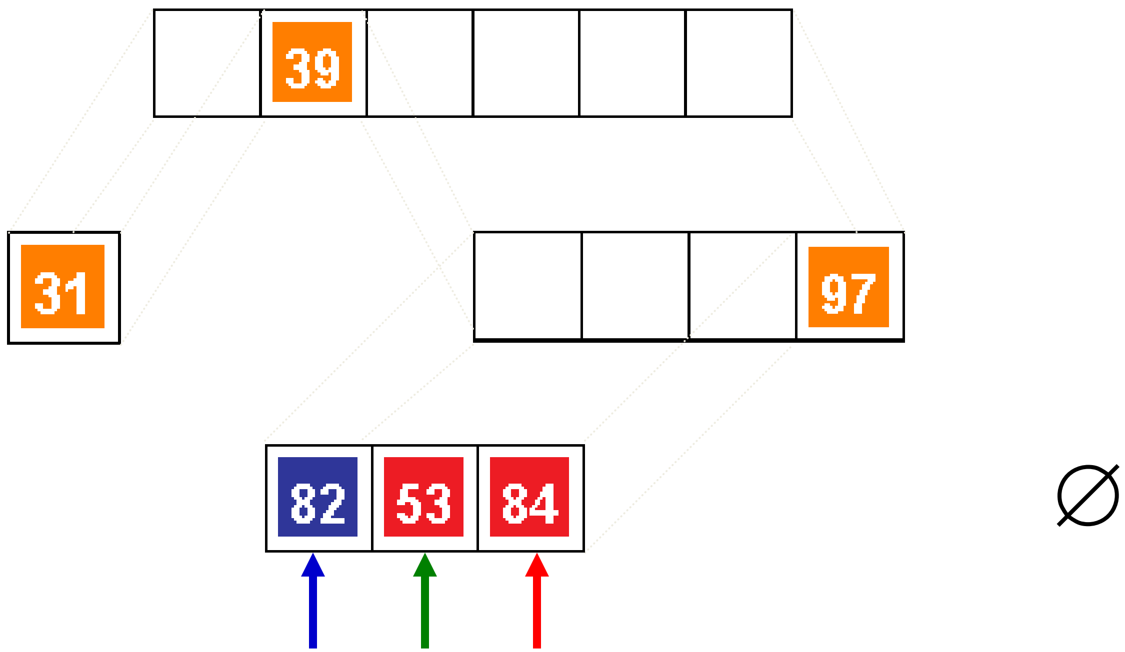


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

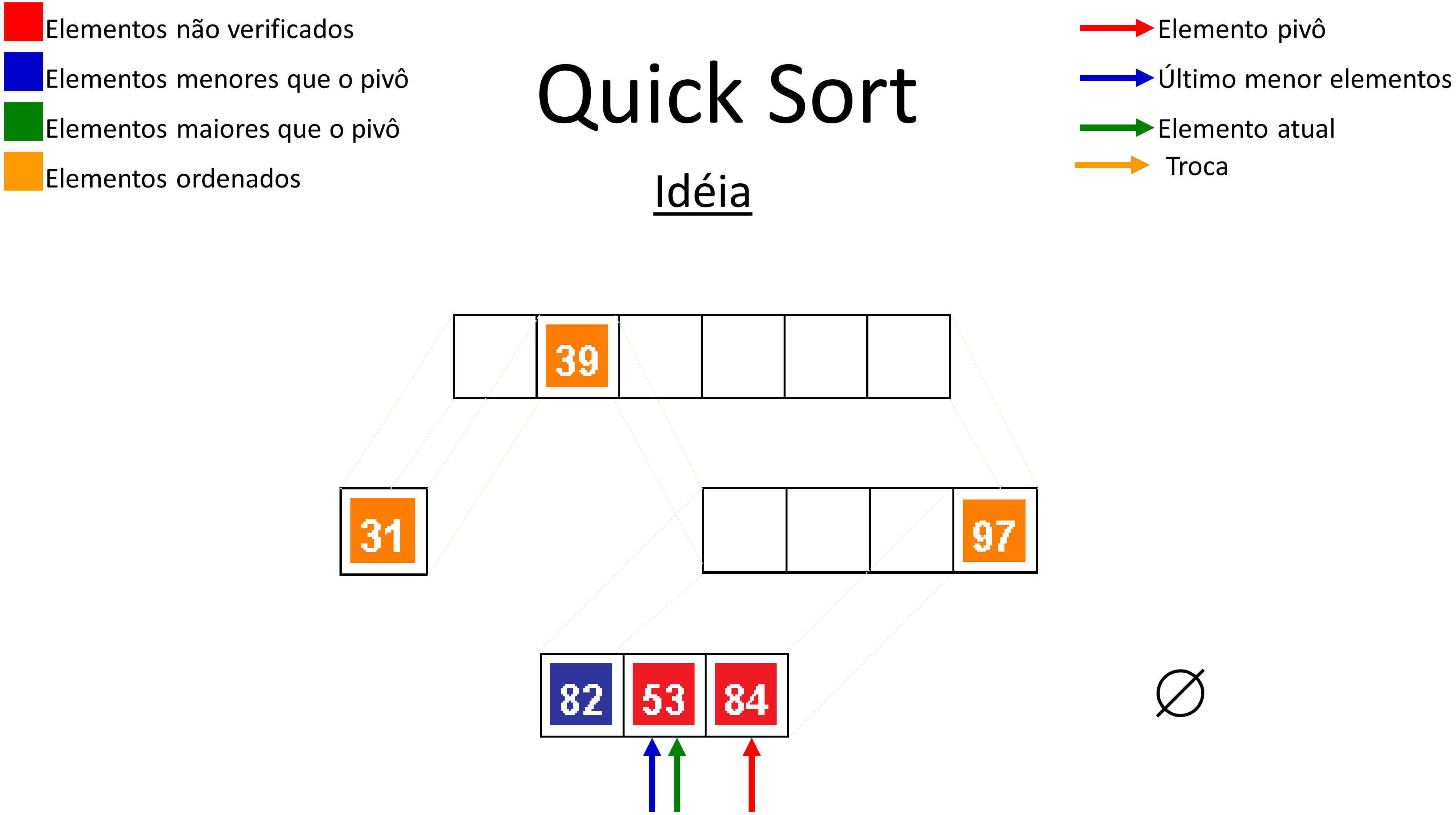
Quick Sort

Idéia



Quick Sort

Idéia

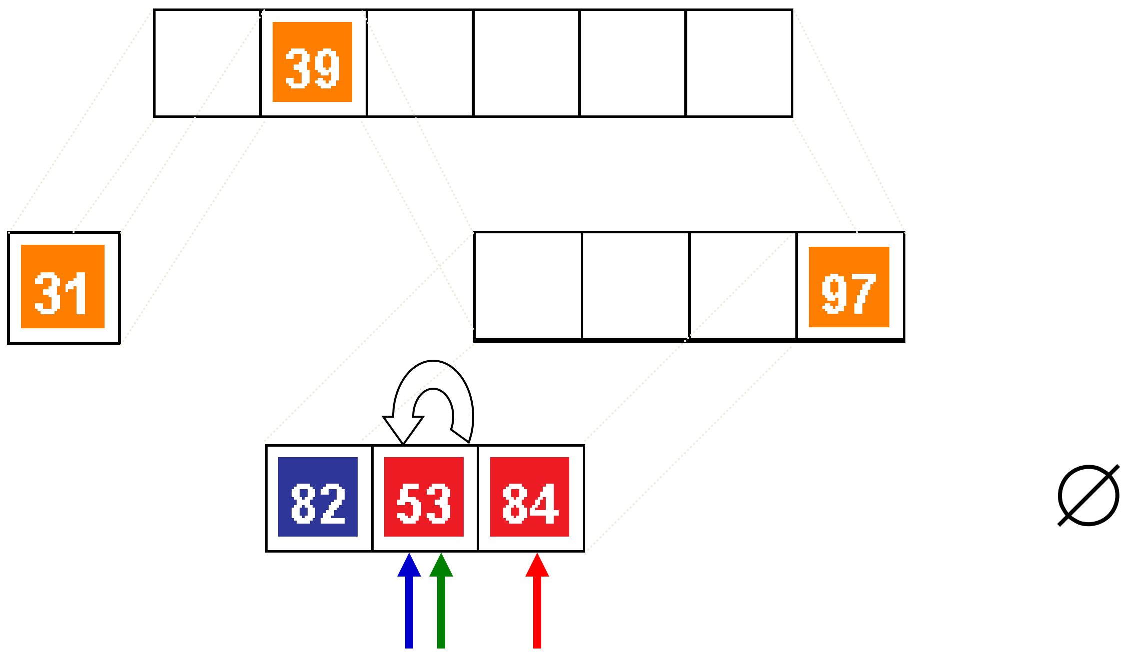


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

Quick Sort

Idéia



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

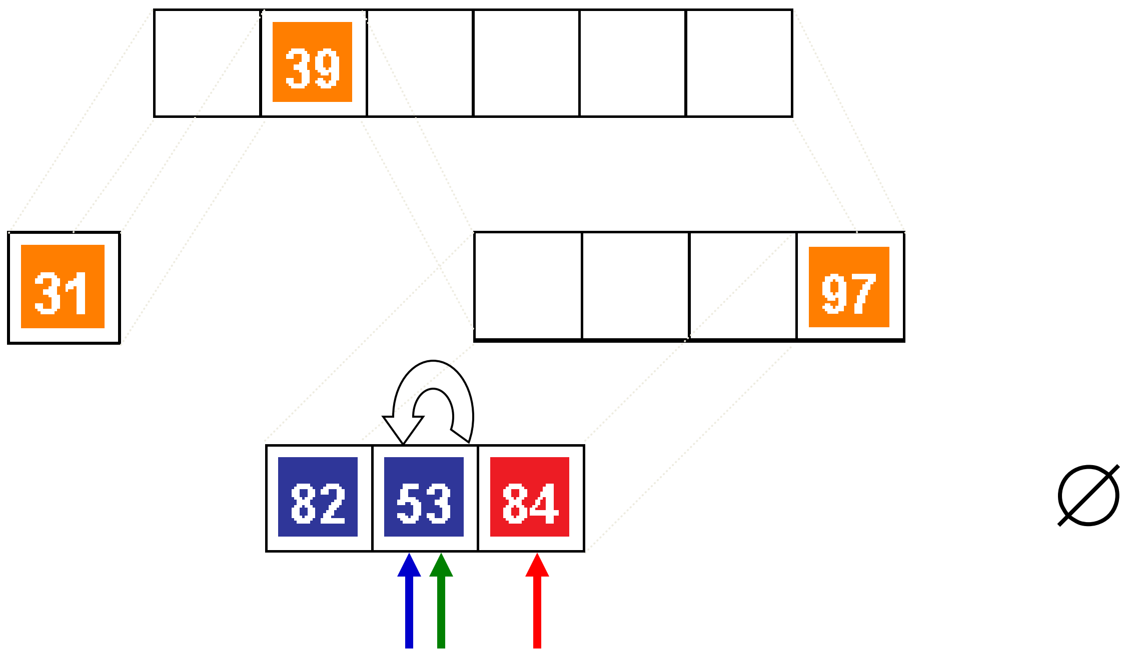
Elemento pivô

Último menor elementos

Elemento atual

Troca

Idéia

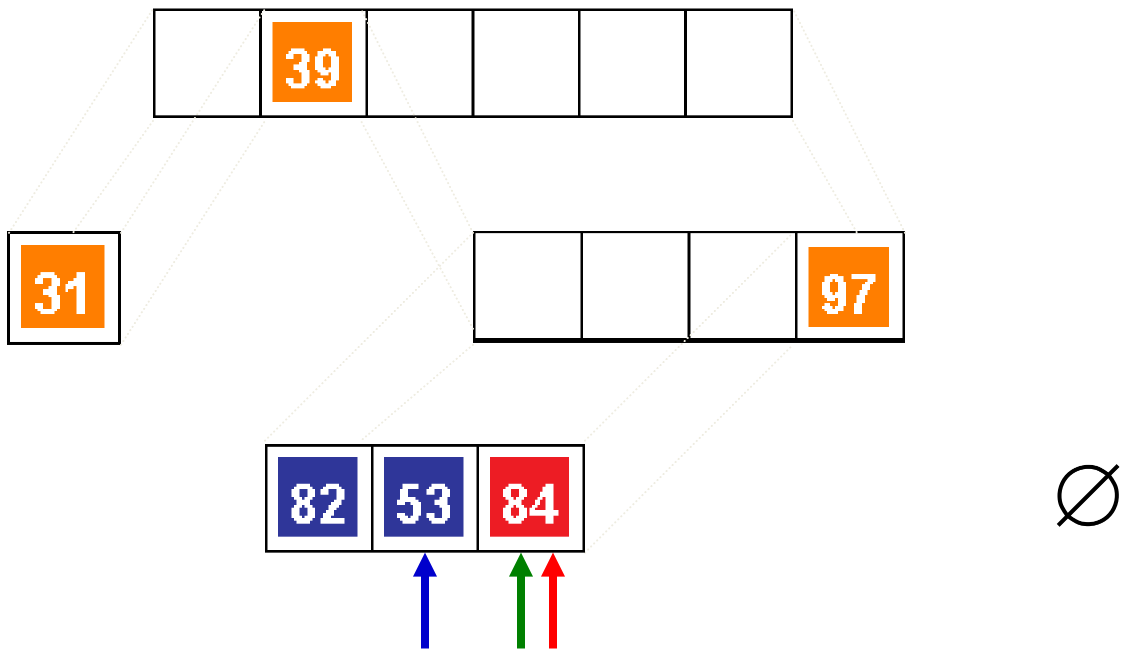


Quick Sort

Idéia

Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

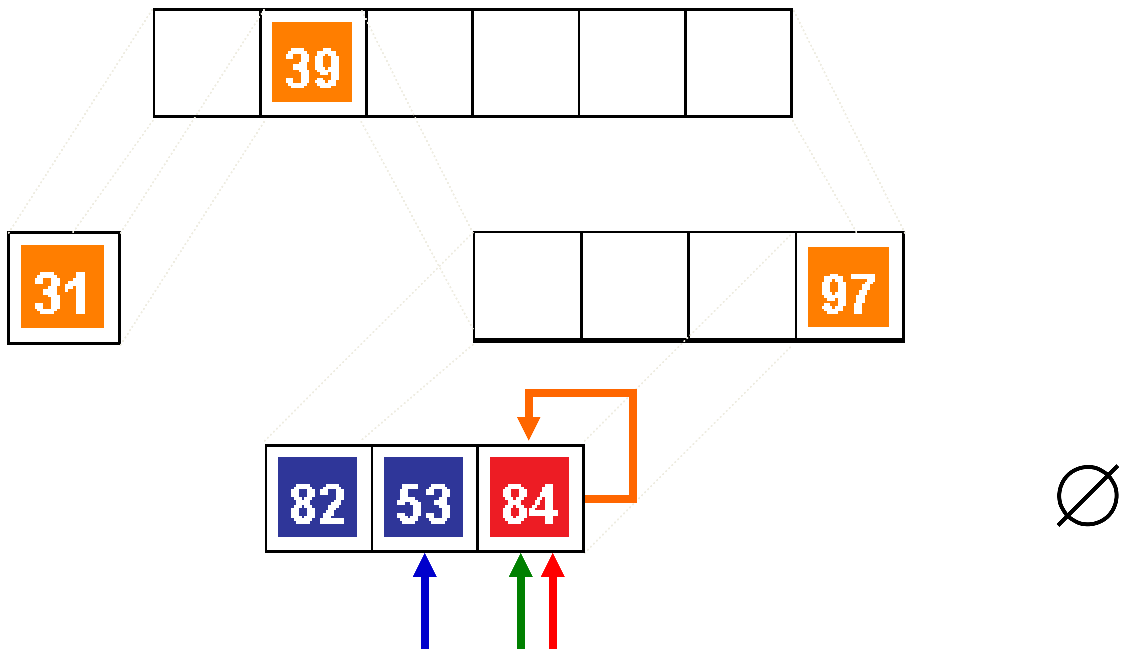


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

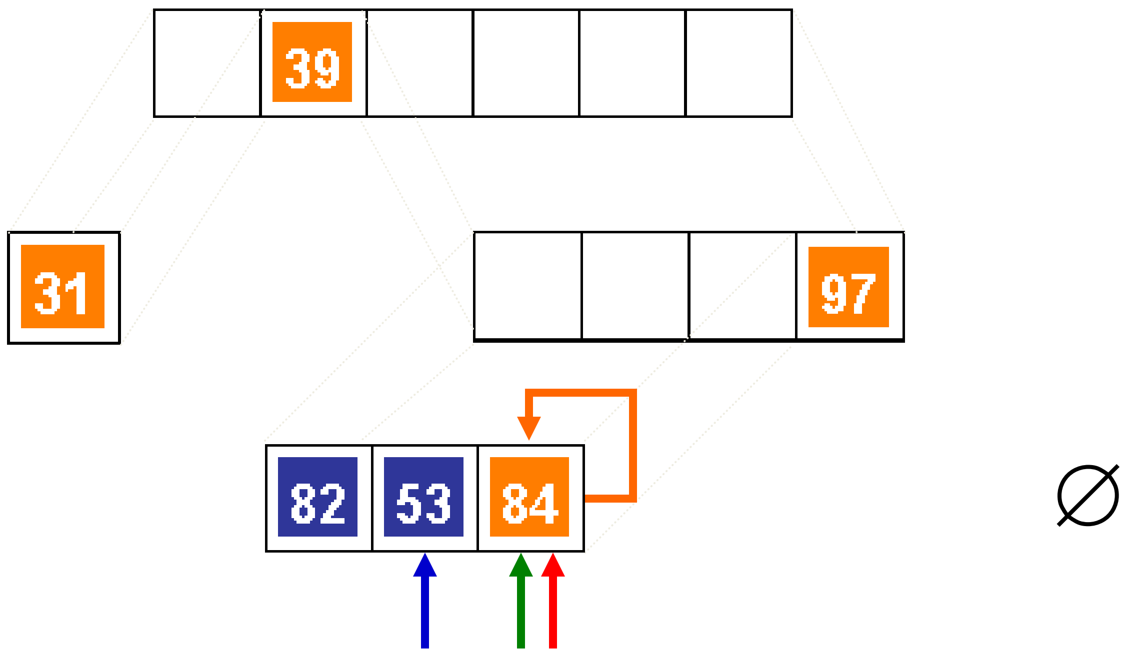


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

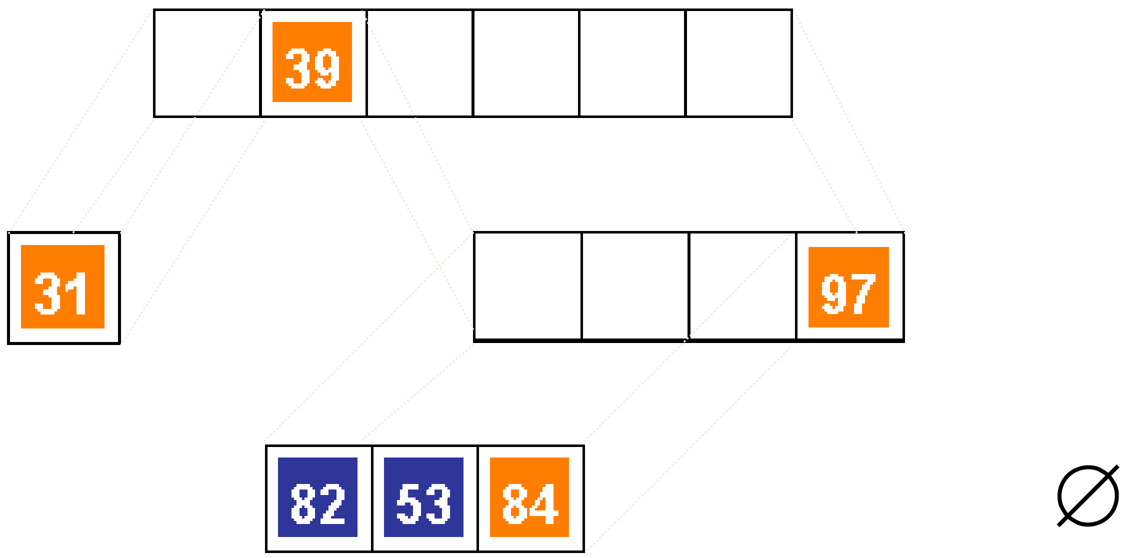


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia

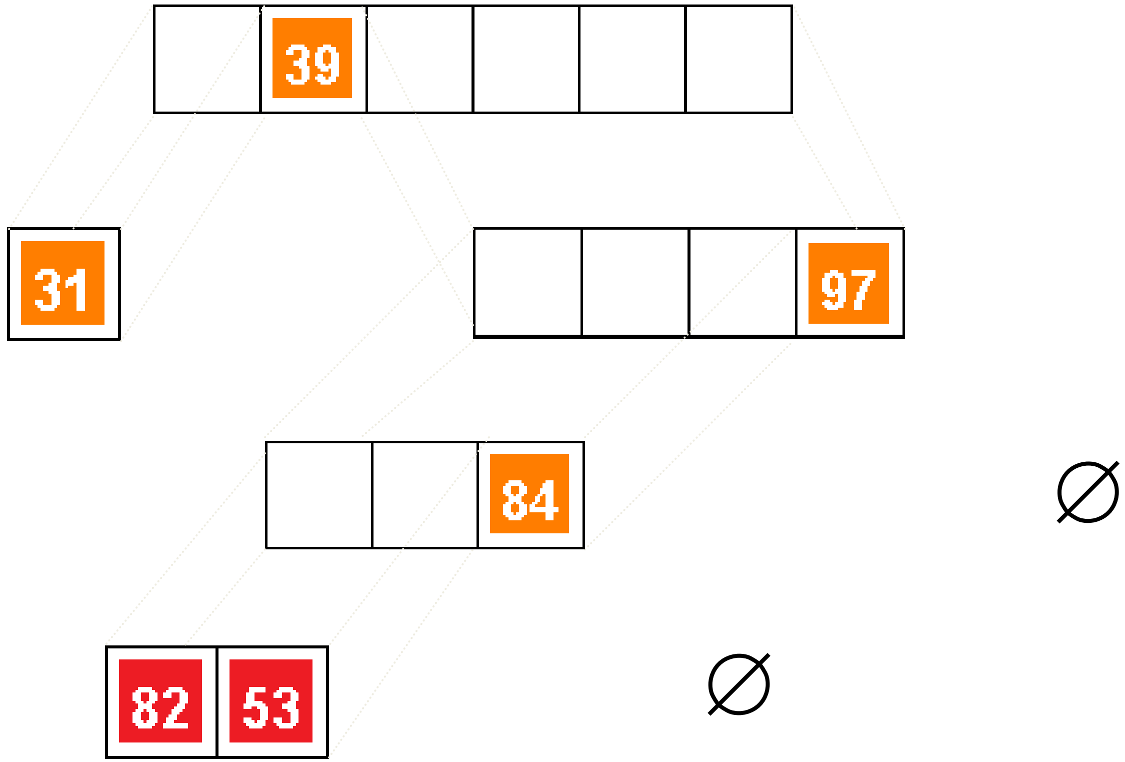


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

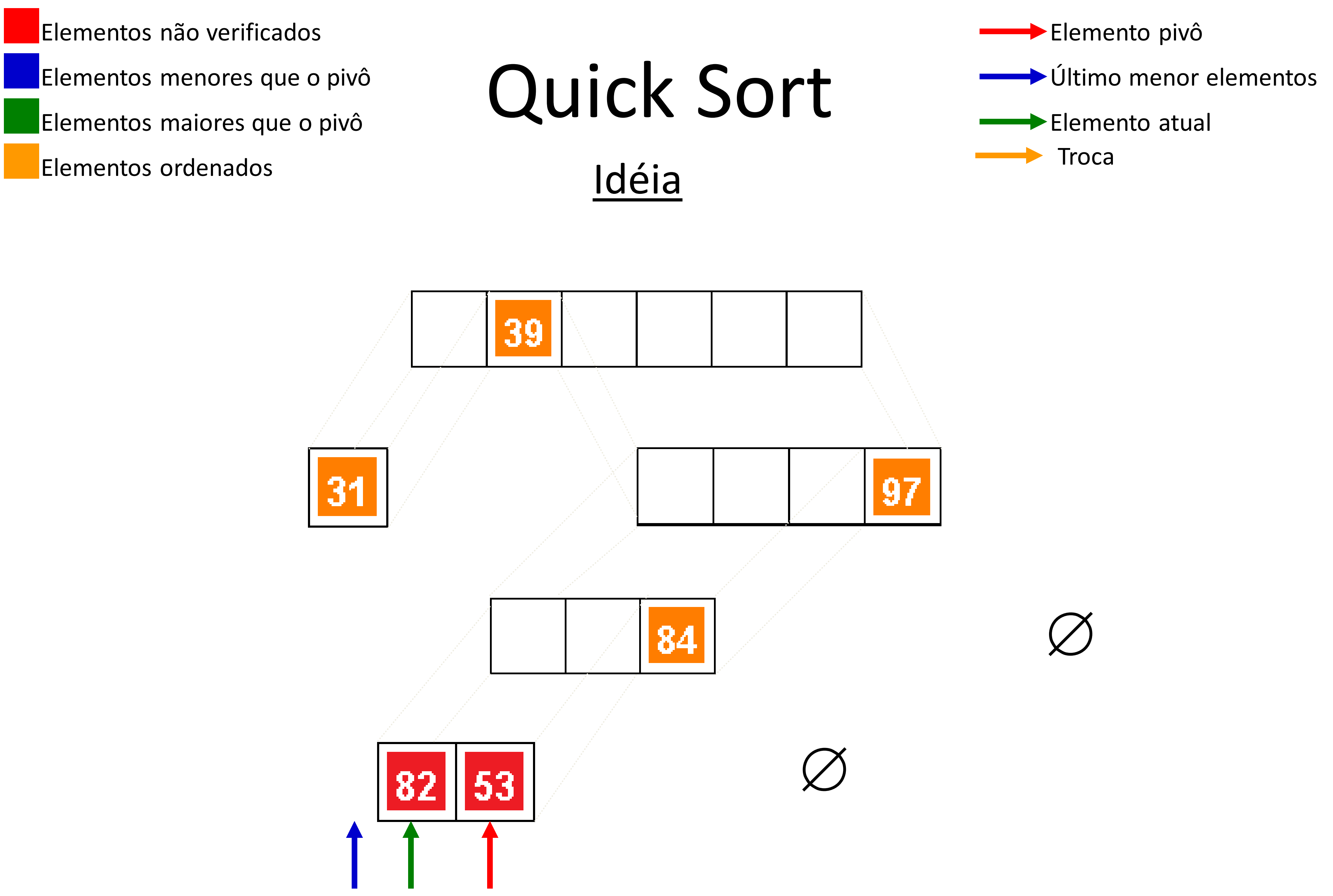
Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Quick Sort

Idéia

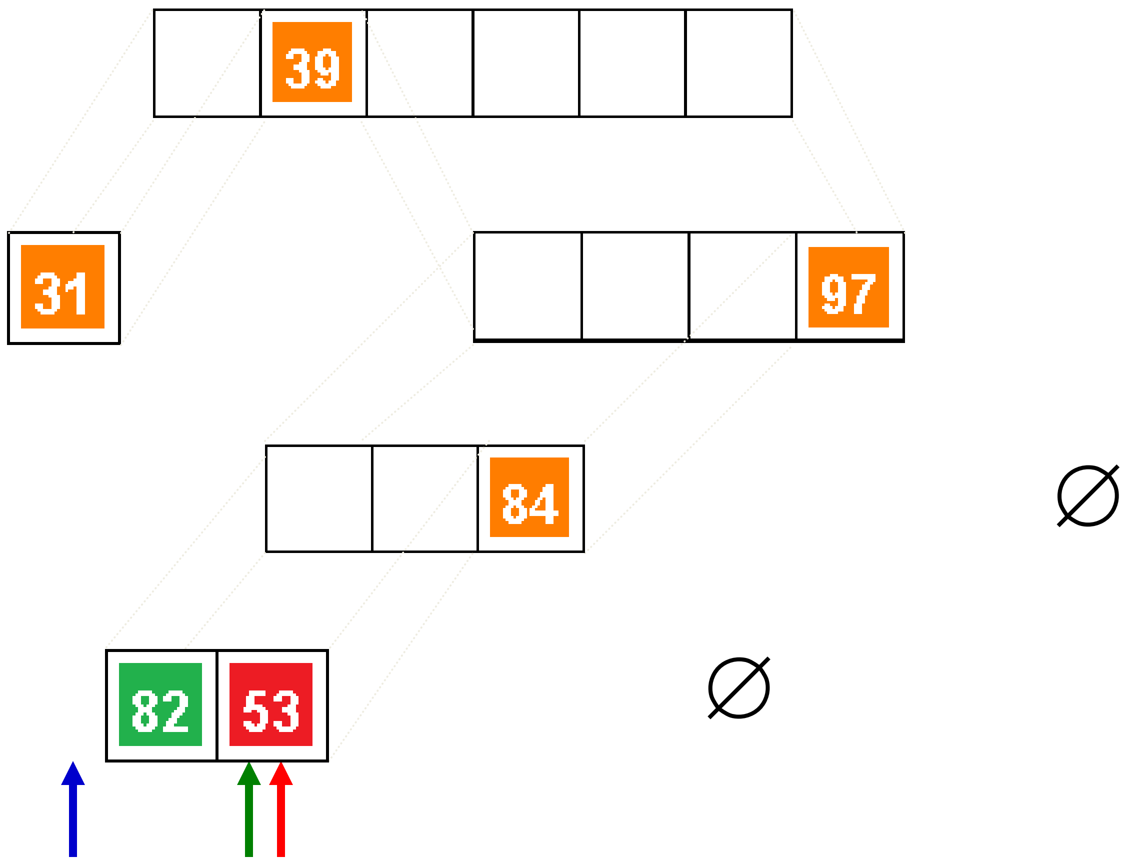


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

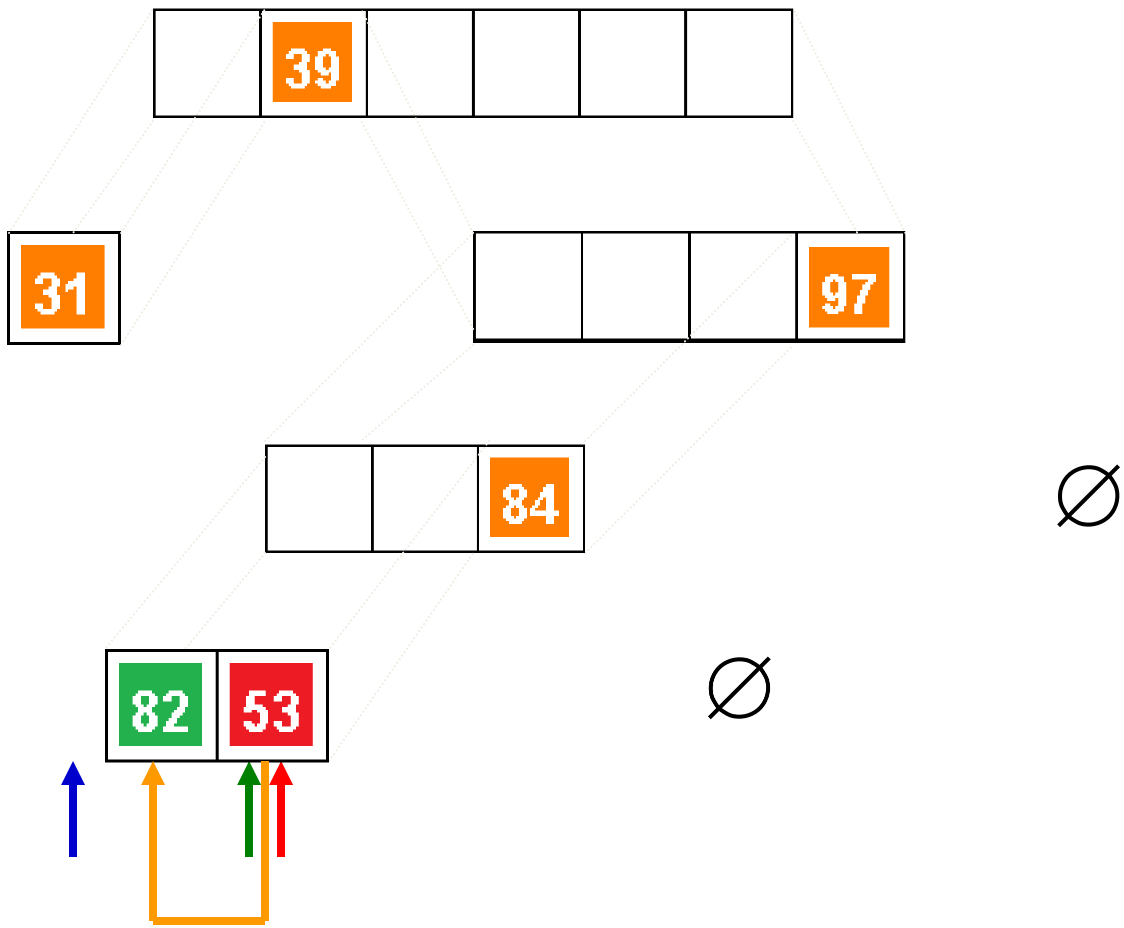


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

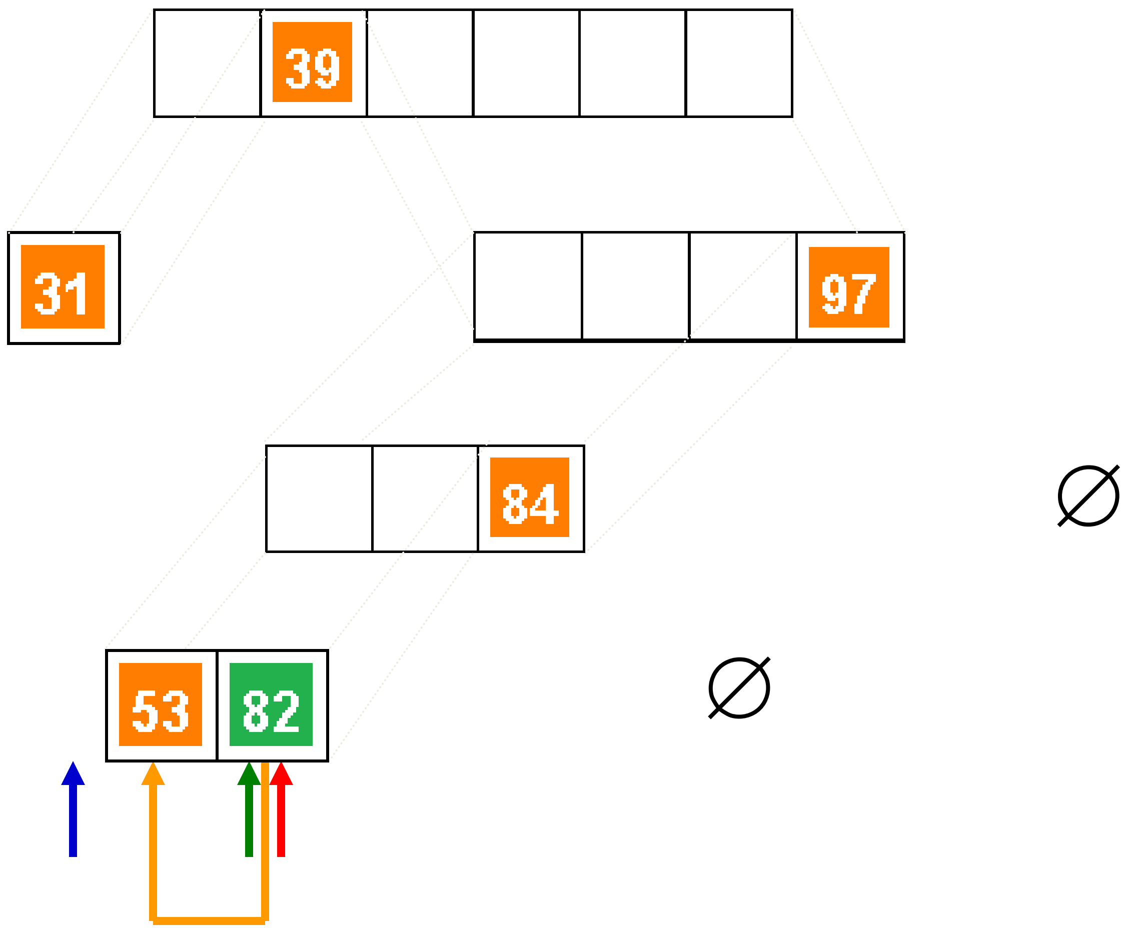


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

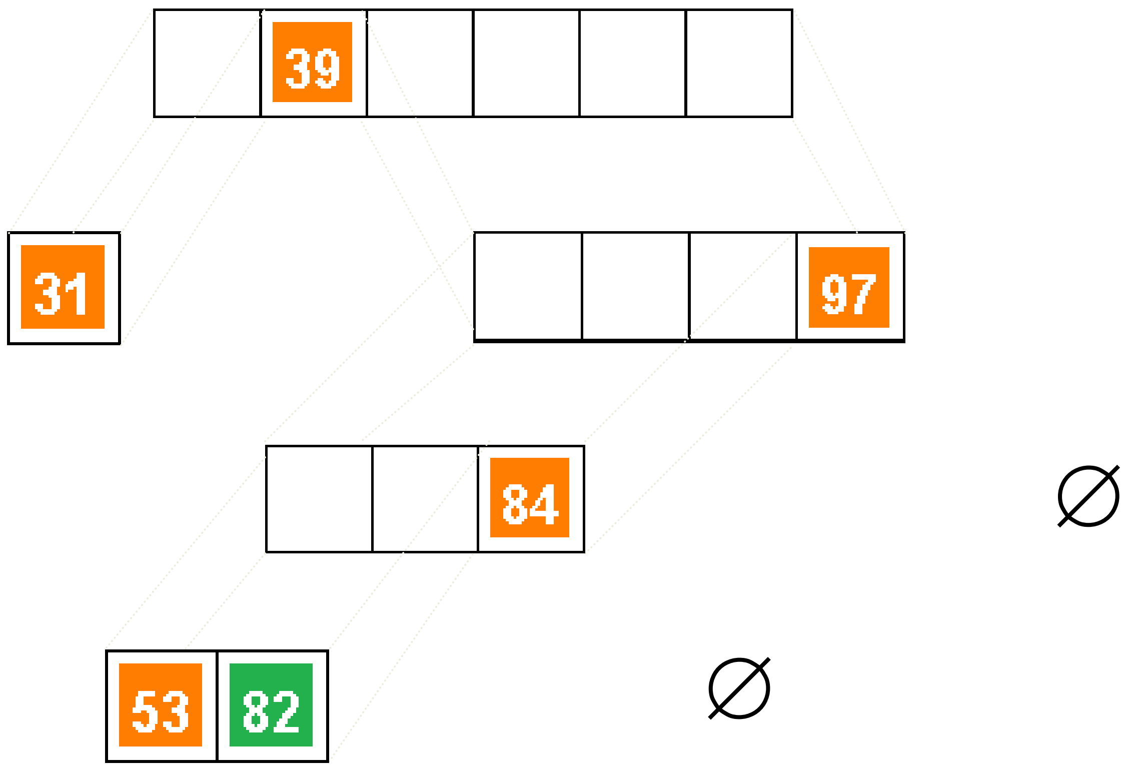


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

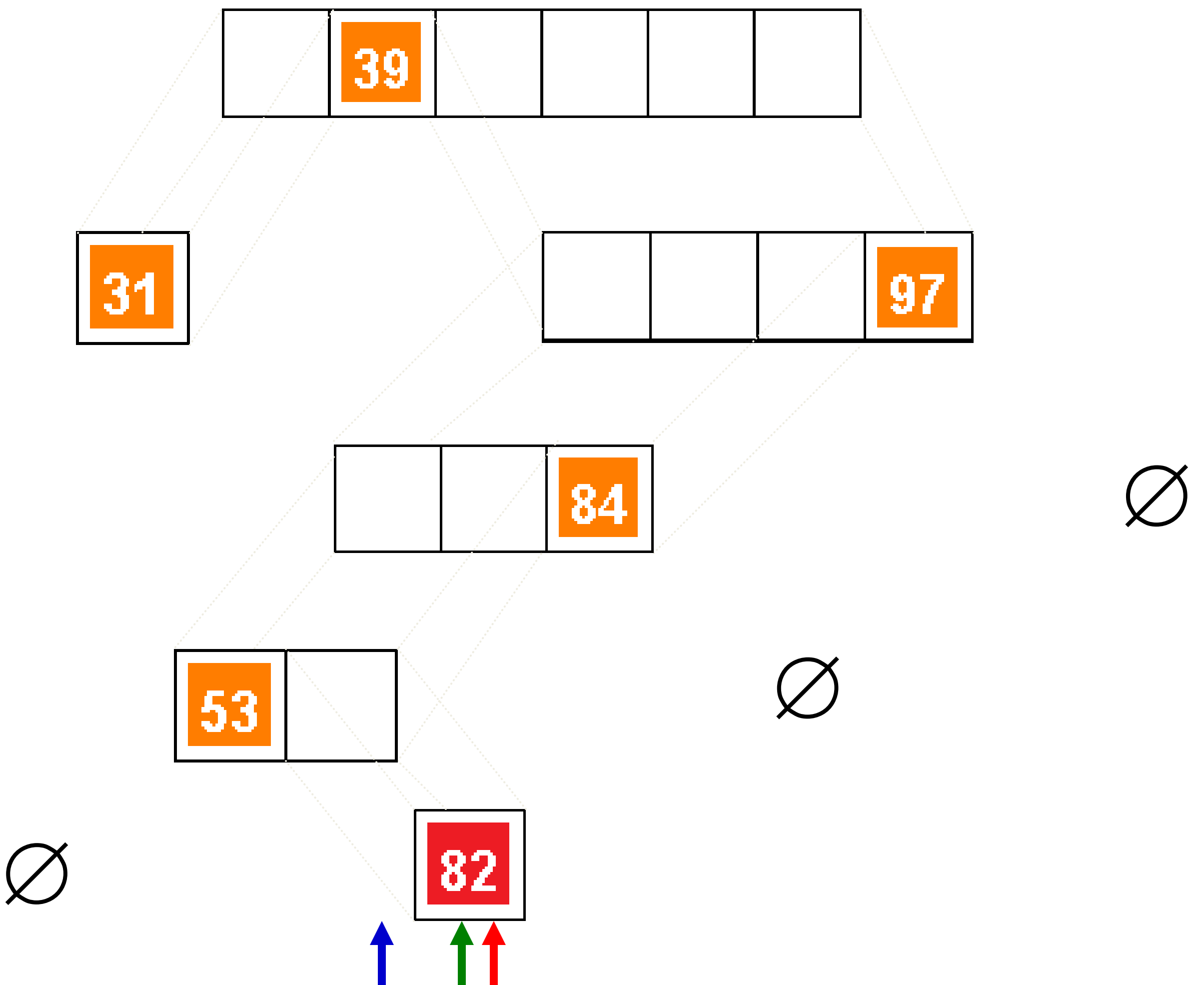


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

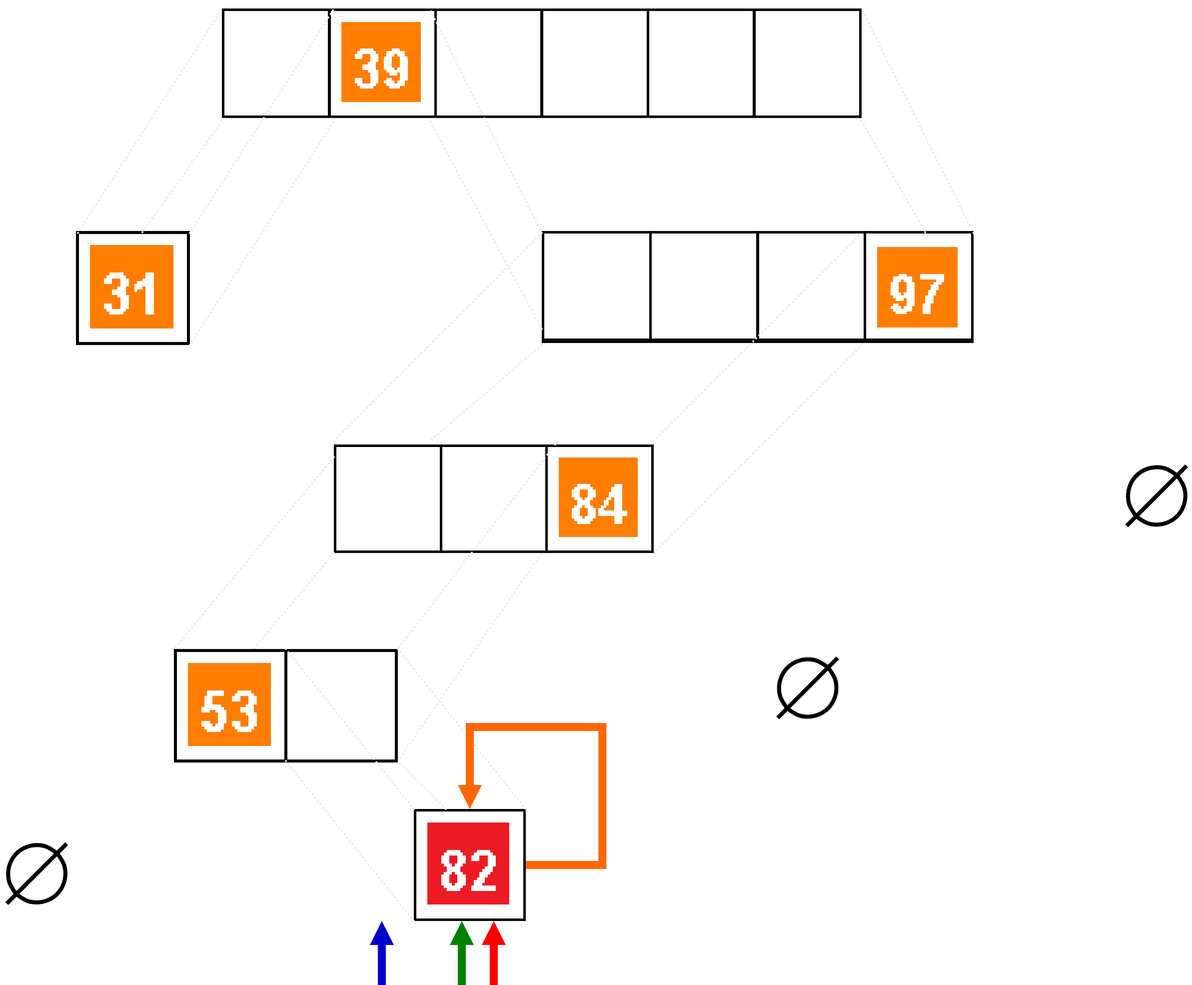


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

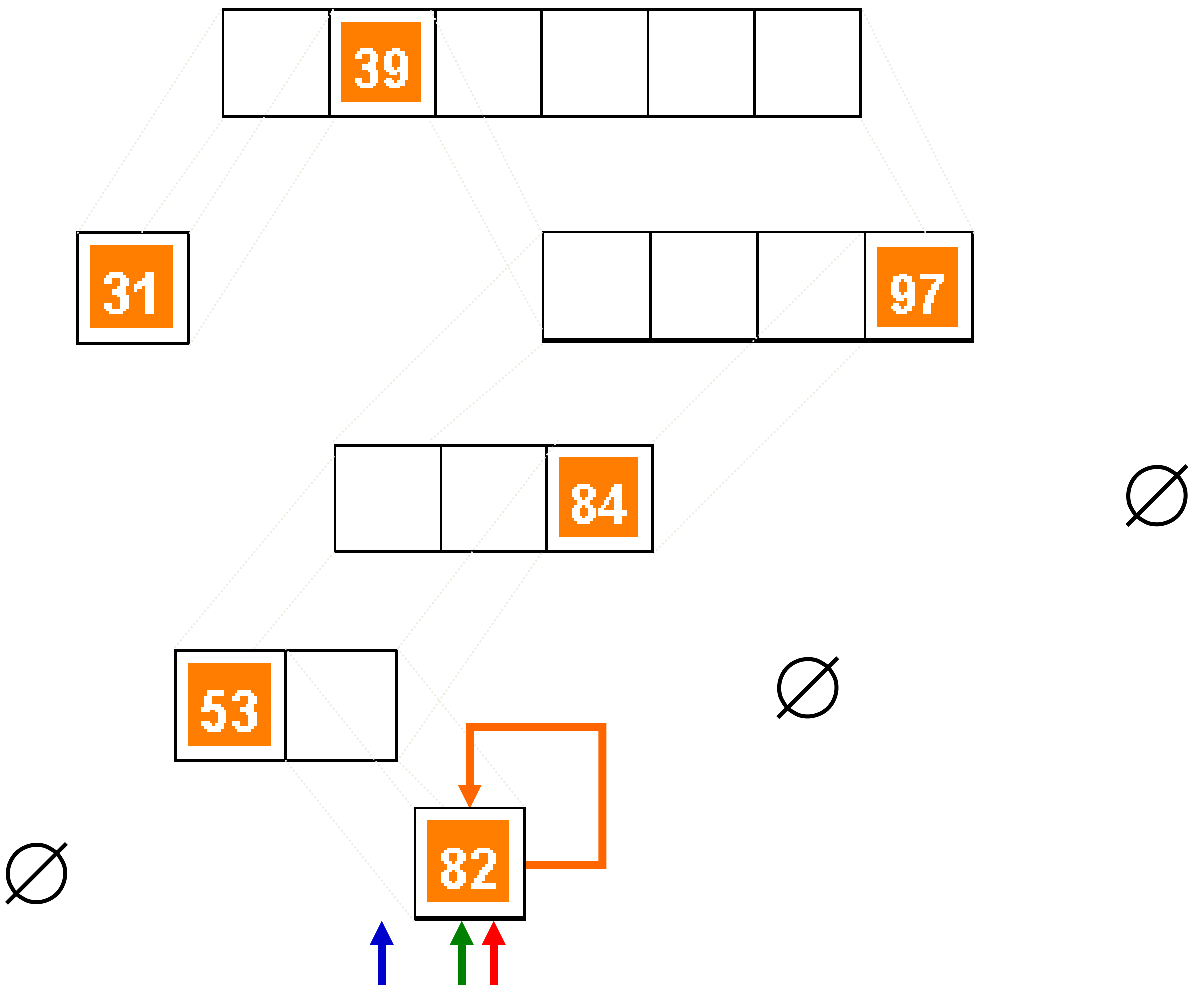


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

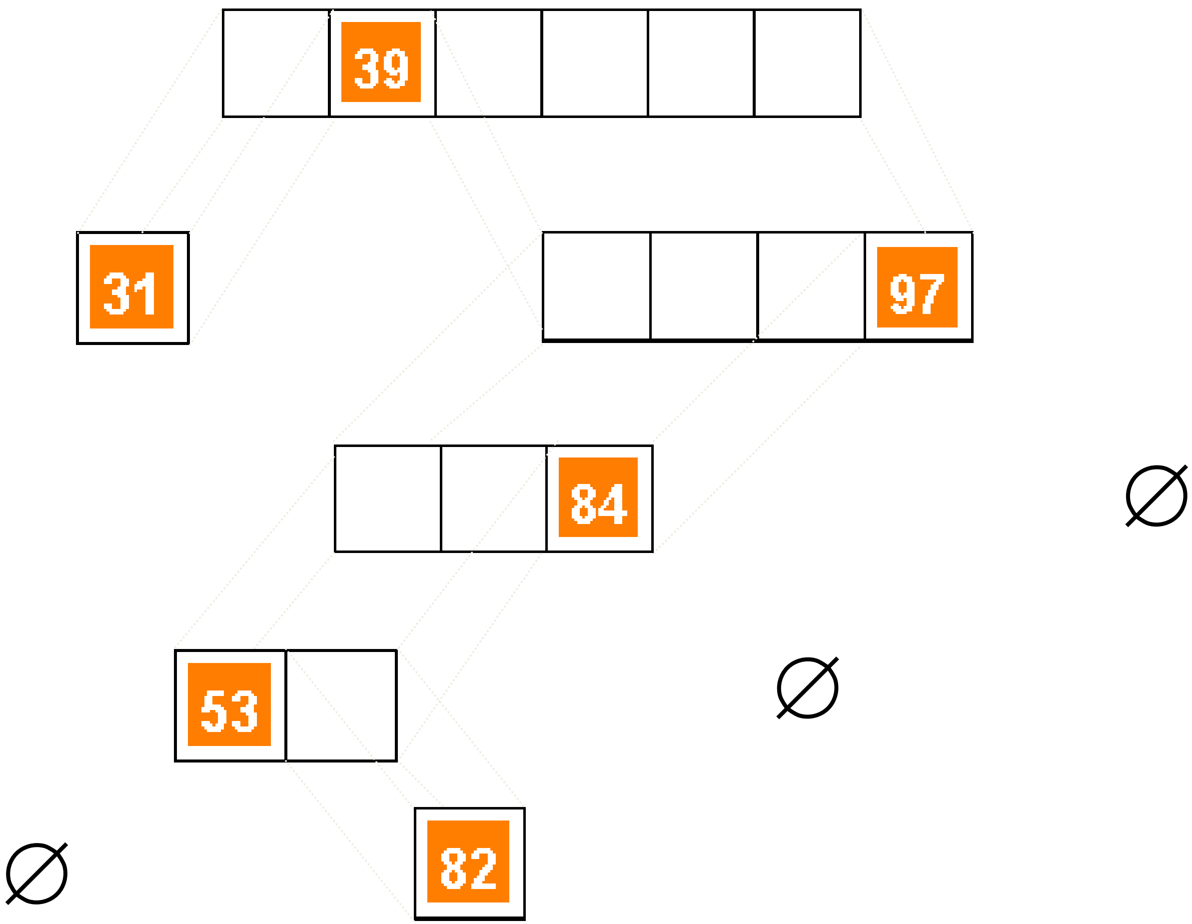


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Elemento pivô
Último menor elementos
Elemento atual
Troca

Quick Sort

Idéia

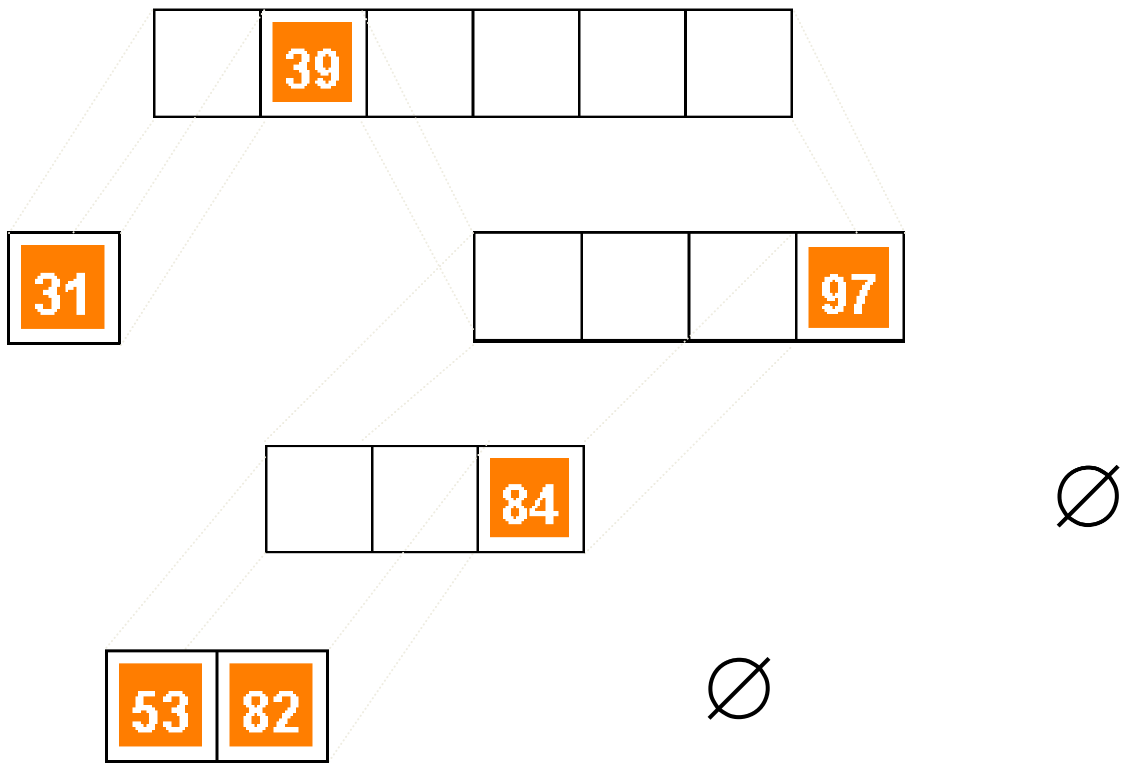


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Elemento pivô
Último menor elementos
Elemento atual
Troca

Idéia

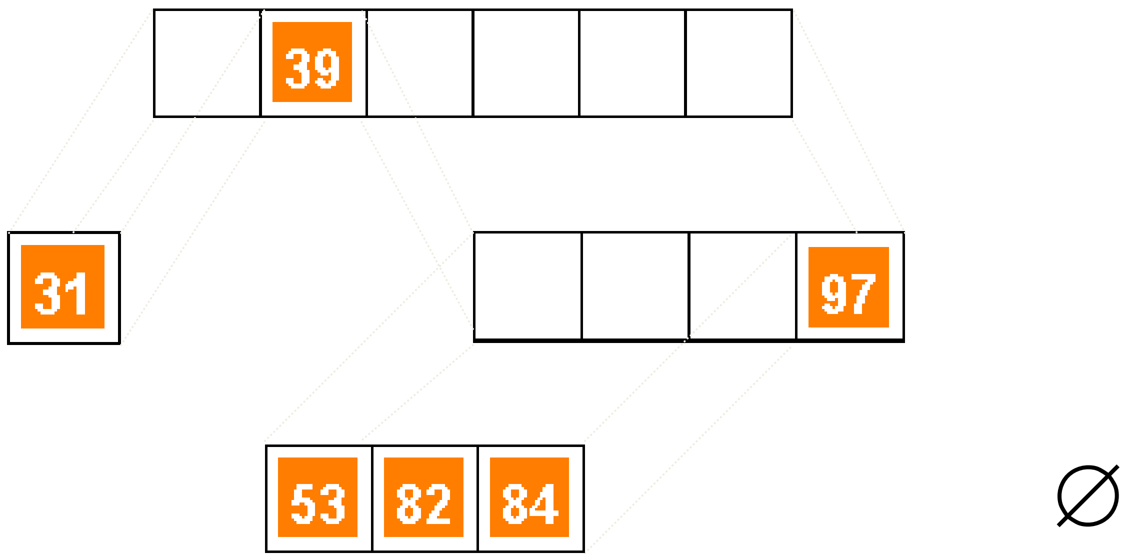


Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca



Elementos não verificados

Elementos menores que o pivô

Elementos maiores que o pivô

Elementos ordenados

Quick Sort

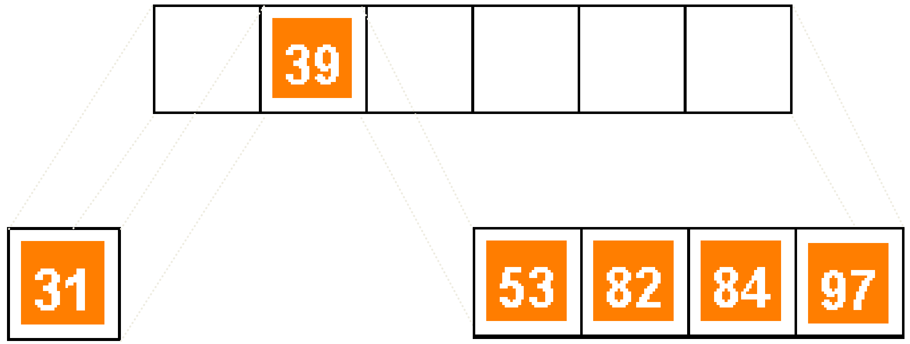
Elemento pivô

Último menor elementos

Elemento atual

Troca

Idéia



Elementos não verificados
Elementos menores que o pivô
Elementos maiores que o pivô
Elementos ordenados

Quick Sort

Idéia

Elemento pivô
Último menor elementos
Elemento atual
Troca

31	39	53	82	84	97

Particiona

```c
int particiona(int *v, int e, int d)  {
  int pm=e-1, i, aux;
  for(i=e; i<d; i++)
  {
         if(v[i]<=v[d])
         {
                pm++;
                aux = v[pm];
                v[pm] = v[i];
                v[i] = aux;
         }
  }
  aux = v[pm+1];
  v[pm+1] = v[d];
  v[d] = aux;
  return pm+1;
}
```

Quick Sort

```
void quickSort(int *v, int e, int d)  {
    int p;
    if(e < d)
    {
            p = particiona(v, e, d);
            quickSort(v, e, p-1);
            quickSort(v, p+1, d);
    }
}
```

Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

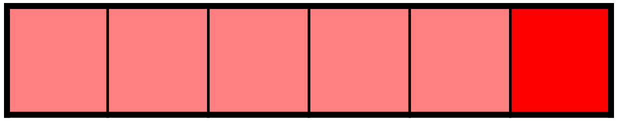
A cada iteração pivô parte o arranjo em duas metades desbalanceadas.

Elementos não ordenados

Elemento pivô

Elementos ordenados

Pivô ordenado

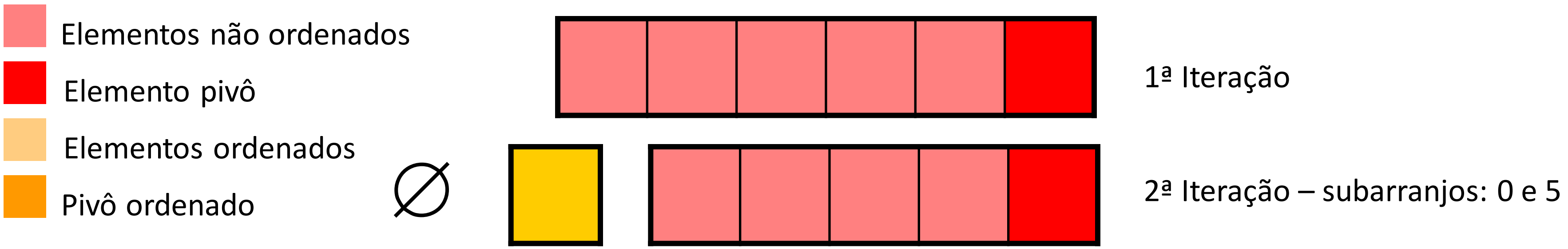


1ª Iteração

Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

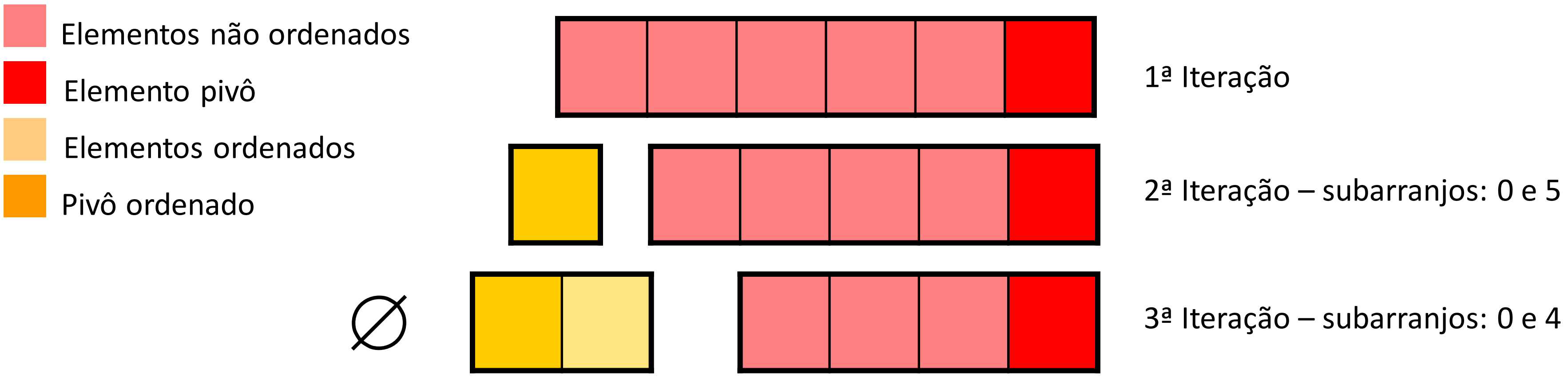
A cada iteração pivô parte o arranjo em duas metades desbalanceadas.



Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

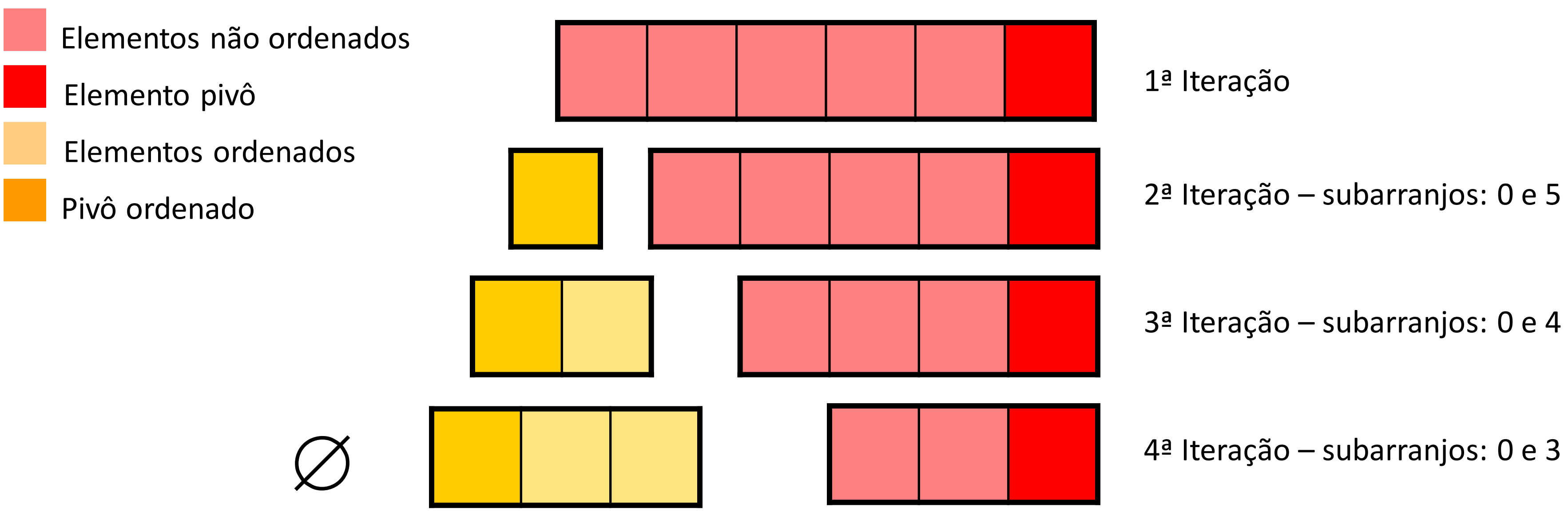
A cada iteração pivô parte o arranjo em duas metades desbalanceadas.



Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

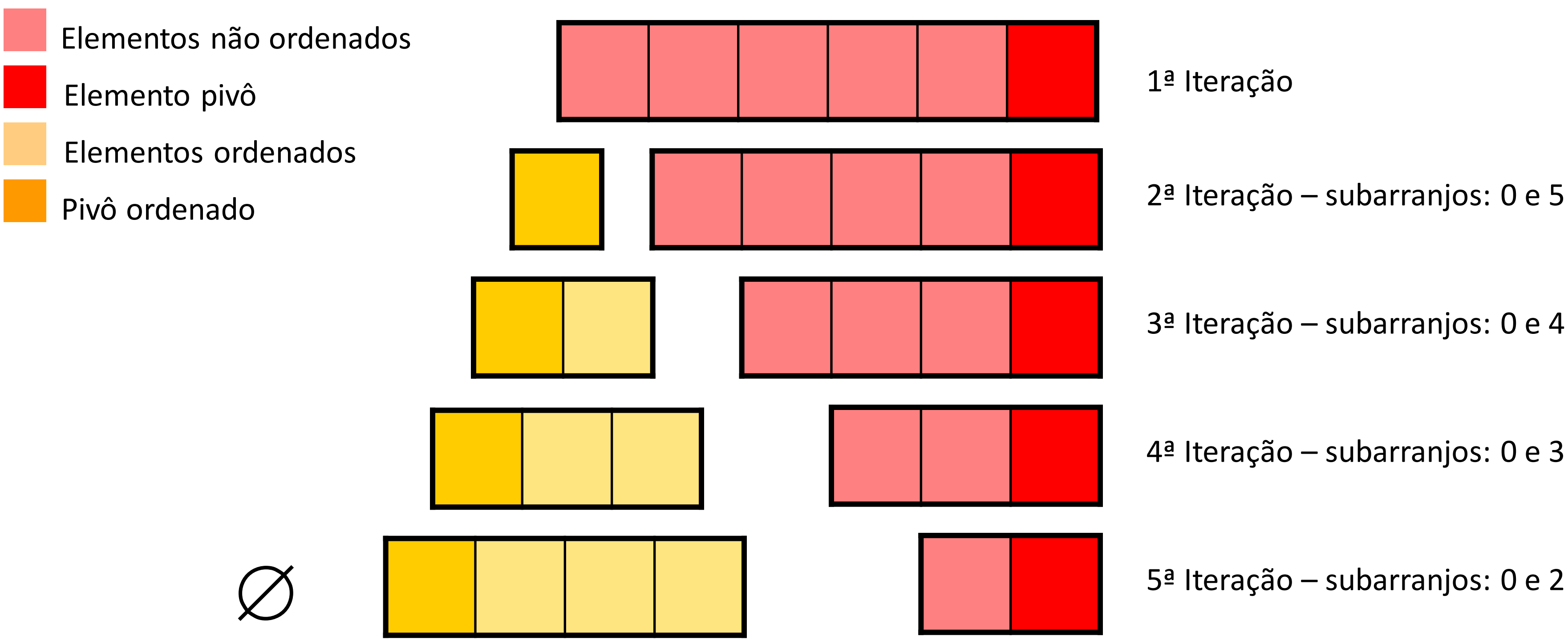
A cada iteração pivô parte o arranjo em duas metades desbalanceadas.



Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

A cada iteração pivô parte o arranjo em duas metades desbalanceadas.



Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

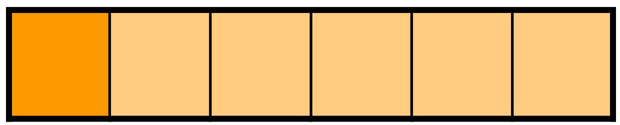
A cada iteração pivô parte o arranjo em duas metades desbalanceadas.

Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

A cada iteração pivô parte o arranjo em duas metades desbalanceadas.

- Elementos não ordenados
- Elemento pivô
- Elementos ordenados
- Pivô ordenado



7ª Iteração – Ordenado após 6 chamadas

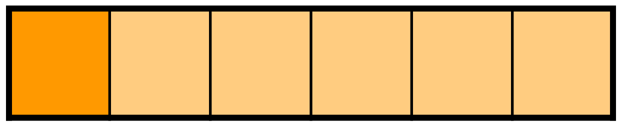
A cada iteração o vetor é particionado em n-1 elementos.

Quick Sort

O pior caso possível do Quick Sort: Particiona Desbalanceado

A cada iteração pivô parte o arranjo em duas metades desbalanceadas.

- Elementos não ordenados
- Elemento pivô
- Elementos ordenados
- Pivô ordenado



7ª Iteração – Ordenado após 6 chamadas

A cada iteração o vetor é particionado em n-1 elementos.

$$T(1) = 1$$

$$T(n) = T(n-1) + n$$

Quick Sort

- Análise de Pior Caso:

$$T(1) = 1$$

$$T(n) = T(n-1) + n$$

$$T(n) = T(n-1) + n$$

$$T(n) = T(n-2) + n + (n-1)$$

$$T(n) = T(n-3) + n + (n-1) + (n-2)$$

$$T(n) = T(n-i) + n + (n-1) + ... + (n-(i-1))$$

Quick Sort

- Análise de Pior Caso:
  - Para i = n-1

$$T(n) = T(1) + n + (n-1) + \ldots + (n-((n-1)-1))$$

$$T(n) = 1 + n + (n-1) + \ldots + 2$$

$$T(n) = n + (n-1) + \ldots + 2 + 1$$

$$T(n) = \frac{n \cdot (n+1)}{2}$$

$$T(n) = \frac{n^2 + n}{2}$$

Quick Sort

- Análise de Pior Caso:
  - Verificando através do método da substituição:
    - Para n=1

$$T(1) = 1$$

$$T(n) = \frac{n^2 + n}{2}$$

$$T(1) = \frac{1^2 + 1}{2}$$

$$T(1) = \frac{1+1}{2} = \frac{2}{2} = 1$$

Quick Sort

- Análise de Pior Caso:
  - Verificando através do método da substituição:
    - Para n

$$T(n) = T(n-1) + n$$

$$T(n) = \frac{n^2 + n}{2}$$

$$T(n) = \left(\frac{(n-1)^2 + (n-1)}{2} \right) + n$$

$$T(n) = \left(\frac{n^2 - 2n + 1 + n - 1}{2} \right) + n$$

$$T(n) = \left(\frac{n^2 - n}{2} \right) + n$$

$$T(n) = \frac{n^2 - n + 2n}{2}$$

$$T(n) = \frac{n^2 + n}{2}$$

Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

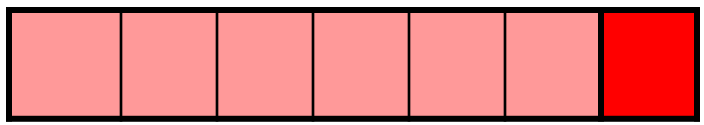
A cada iteração pivô parte o arranjo em duas metades iguais.

Elementos não ordenados

Elemento pivô

Elementos ordenados

Pivô ordenado



1ª Iteração

Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

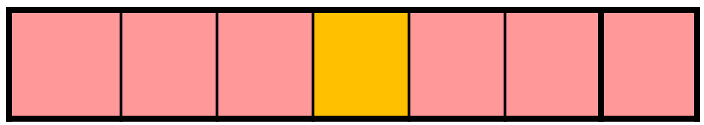
A cada iteração pivô parte o arranjo em duas metades iguais.

Elementos não ordenados

Elemento pivô

Elementos ordenados

Pivô ordenado

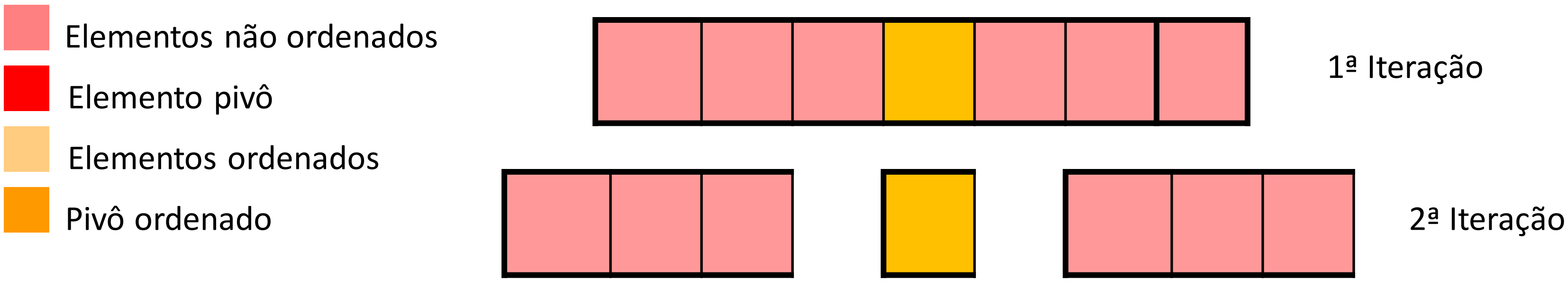


1ª Iteração

Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

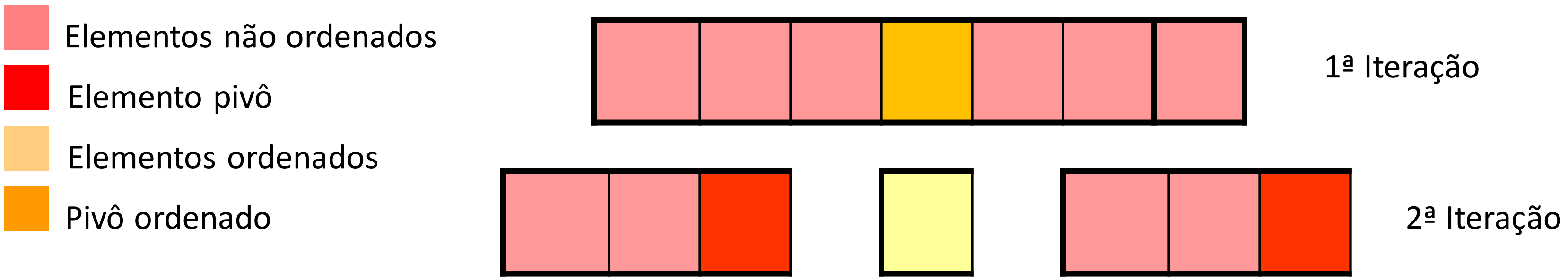
A cada iteração pivô parte o arranjo em duas metades iguais.



Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

A cada iteração pivô parte o arranjo em duas metades iguais.

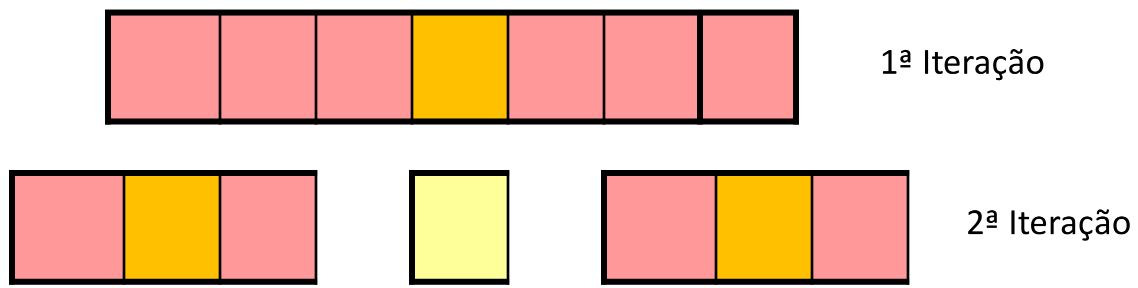


Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

A cada iteração pivô parte o arranjo em duas metades iguais.

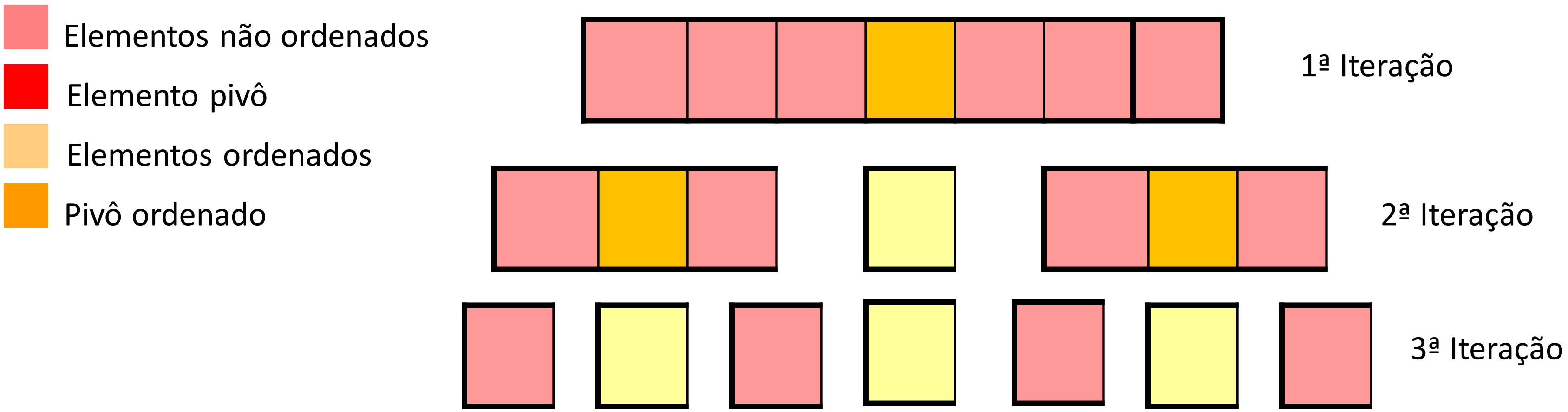
- Elementos não ordenados
- Elemento pivô
- Elementos ordenados
- Pivô ordenado



Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

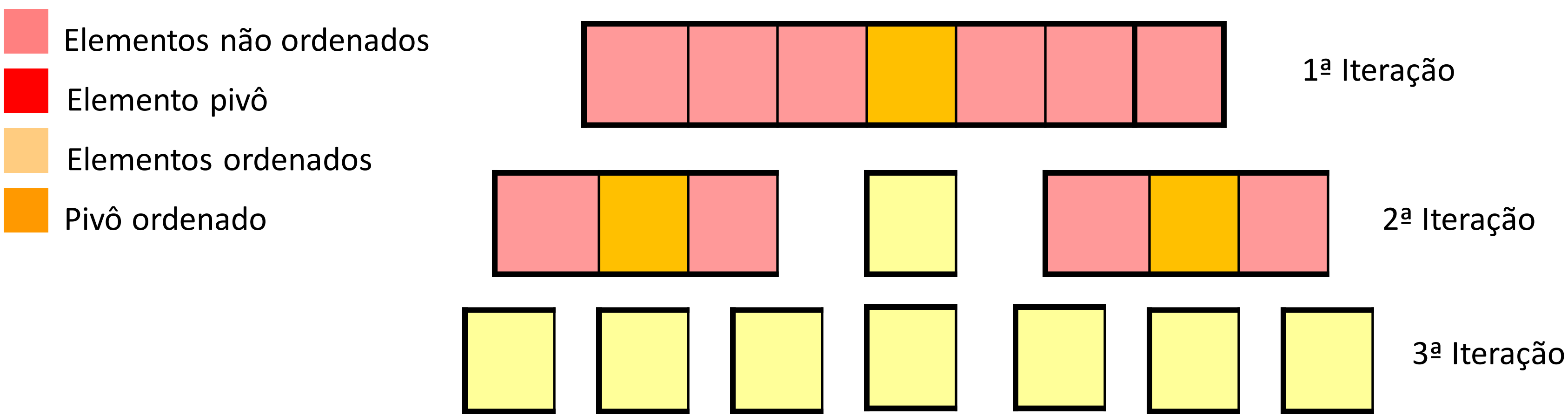
A cada iteração pivô parte o arranjo em duas metades iguais.



Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

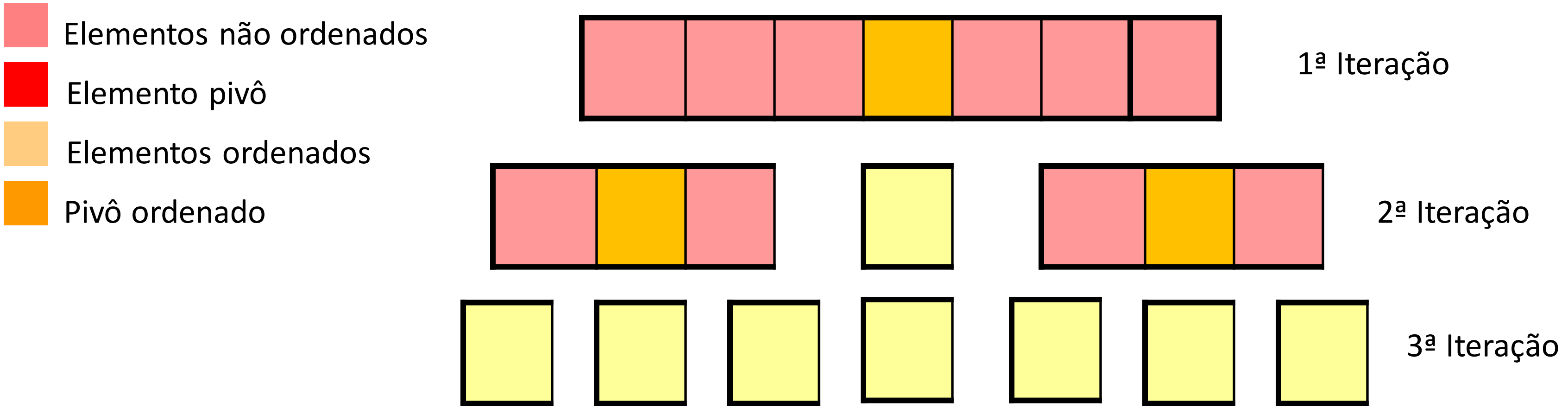
A cada iteração pivô parte o arranjo em duas metades iguais.



Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

A cada iteração pivô parte o arranjo em duas metades iguais.

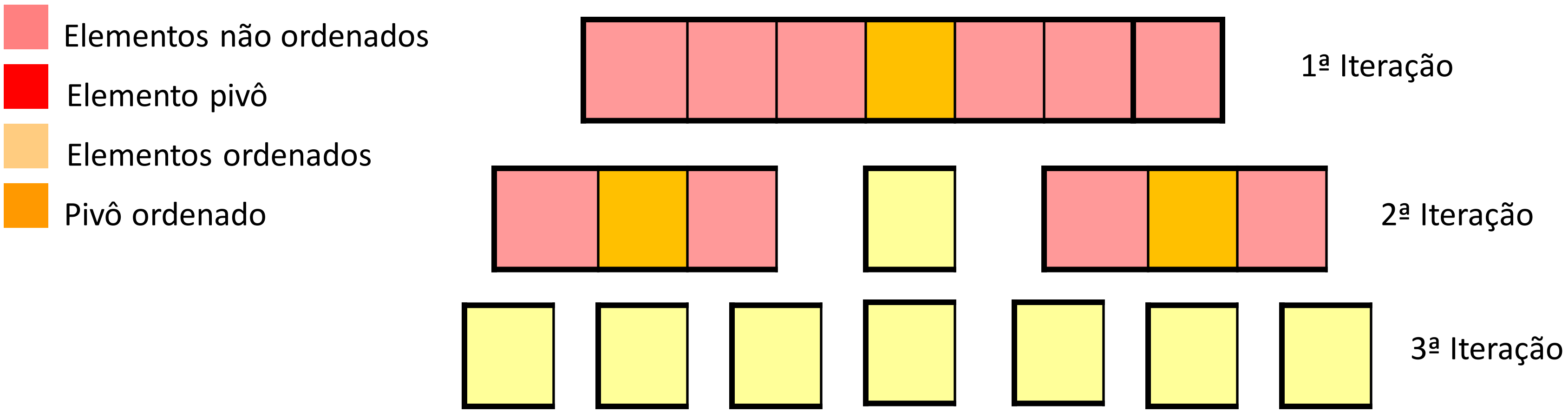


A cada iteração o vetor é particionado em n/2 elementos.

Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

A cada iteração pivô parte o arranjo em duas metades iguais.



A cada iteração o vetor é particionado em n/2 elementos.

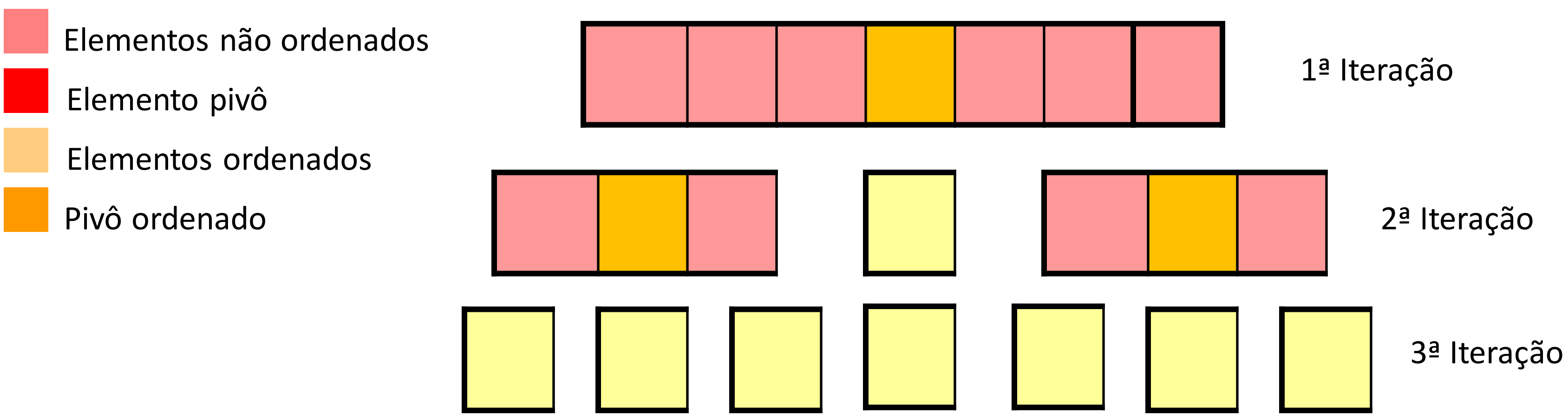
$$T(1) = 1$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$

Quick Sort

O melhor caso do Quick Sort: Particiona Balanceado

A cada iteração pivô parte o arranjo em duas metades iguais.



A cada iteração o vetor é particionado em n/2 elementos.

$$T(1) = 1$$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$

Exatamente igual ao Merge Sort

Portanto: **$O(n \log_2 n)$**

Quick Sort

- No Caso Médio o Quick Sort é **O(n $\log_2$ n)**
- Prova: Sedgewick Cap. 7 Pg. 311
- No caso médio o número de comparações é cerca de **39%** maior que no melhor caso.