Algoritmos de ordenação em arrays 1D

Ordenação

Definição

- Ordenar por **ordem crescente** significa alterar a ordem pela qual surgem os elementos de uma sequência, tal que
  - o primeiro seja **menor** do que o segundo,
  - o segundo **menor** do que o terceiro,
  - e assim sucessivamente.
- Ordenar por **ordem decrescente** significa alterar a ordem pela qual surgem os elementos de uma sequência, tal que
  - o primeiro seja **maior** do que o segundo,
  - o segundo **maior** do que o terceiro,
  - e assim sucessivamente.

Algoritmos de Ordenação

Definição

- Consiste num algoritmo que, após a sua execução, coloca por ordem crescente ou decrescente uma dada sequência **A** de **N** elementos
  - $A[0] \leq A[1] \leq A[2] \leq ... \leq A[N-1]$ (ordem crescente)
  - $A[0] \geq A[1] \geq A[2] \geq ... \geq A[N-1]$ (ordem decrescente)

Utilidade

- Estes algoritmos são úteis para a resolução de problemas como
  - pesquisa numa lista (por exemplo, telefónica)
  - agrupar elementos repetidos de uma sequência:

    (7, 1, 2, 7, 2, 5) $\rightarrow$ (1, 2, 2, 5, 7, 7) ou (7, 7, 5, 2, 2, 1)

Algoritmos de Ordenação

Implementação

- Na implementação dos algoritmos que serão estudados, os elementos da sequência são guardados num array 1D (arrays de 1 dimensão)

Nota

- Daqui para a frente, sempre que omitido, considera-se que
  - a sequência (array 1D) é composta apenas por **elementos** do tipo **inteiro**
  - pretende-se ordenar a sequência (array 1D) por **ordem crescente**

Ordenação por Seleção

Definição

- Algoritmo iterativo (não recursivo)
- Algoritmo que consiste em organizar um array 1D **A** da seguinte forma
  - 1ª posição (A[0]): colocar o menor elemento
  - 2ª posição (A[1]): colocar o 2º menor elemento

    ...
  - k-ésima posição (A[k-1]): colocar o k-ésimo menor elemento

    ...
  - N-ésima (última) posição (A[N-1]): colocar o maior elemento

Estratégia

- Para a 1ª posição (A[0])
  - determinar a posição do menor elemento (*posMenor*) de A[0] a A[N-1]
  - trocar o elemento da posição *posMenor* com o elemento da 1ª posição
- Para a 2ª posição (A[1])
  - determinar a posição do menor elemento (*posMenor*) de A[1] a A[N-1]
  - trocar o elemento da posição *posMenor* com o elemento da 2ª posição

  ...
- Para a (N-1)-ésima posição (A[N-2])
  - determinar a posição do menor elemento (*posMenor*) de A[N-2] a A[N-1]
  - trocar o elemento da posição *posMenor* com o elemento da (N-2)-ésima posição
- Para a N-ésima posição (A[N-1])
  - como só falta um elemento para ordenar, A[N-1], então
    encontra-se automaticamente ordenado (é o maior elemento de todos)

Implementação de algoritmo auxiliar em C

Operação: trocar o conteúdo de duas variáveis do tipo inteiro

```
void trocar (int *a, int *b)
{
  int  aux;
  aux = *a;
  *a = *b;
  *b = aux;
}
```

Implementação do algoritmo em C

Operação: ordenar por ordem crescente um array **A** com **N** inteiros

```
void ordenarSeleccao (int A[], int N)
{
  int  k, j, posMenor;
  for (k = 0; k < N-1; k++) {
      posMenor = k;
      for (j = k+1; j < N; j++) {
          if (A[j] < A[posMenor])
              posMenor = j;
      }
      if (posMenor != k)      // trocar A[posMenor] com A[k]
          trocar(&A[posMenor], &A[k]);
  }
}
```

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 1:

  determinar o menor elemento e colocá-lo na posição 1 (índice 0)

  N = 8

	0	1	2	3	4	5	6	7
A	8	3	5	4	9	6	2	7

  k ← 0

  j ← 1 (k+1), 2, 3, 4, 5, 6, 7

  posMenor ← 0 (k), 1, 6

  como (posMenor=6 ≠ k=0), então trocar A[k=0] com A[posMenor=6]

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 2:

  determinar o 2º menor elemento e colocá-lo na posição 2 (índice 1)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	5	4	9	6	8	7

  k ← 0, 1

    j ← 2 (k+1), 3, 4, 5, 6, 7

    posMenor ← 1 (k)

    como (posMenor=1 = k=1), então não trocar

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 3:

  determinar o 3º menor elemento e colocá-lo na posição 3 (índice 2)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	5	4	9	6	8	7

  k ← 0, 1, 2

    j ← 3 (k+1), 4, 5, 6, 7

    posMenor ← 2 (k), 3

    como (posMenor=3 ≠ k=2), então trocar A[k=2] com A[posMenor=3]

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 4:

  determinar o 4º menor elemento e colocá-lo na posição 4 (índice 3)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	4	5	9	6	8	7

  k ← 0, 1, 2, 3

    j ← 4 (k+1), 5, 6, 7

    posMenor ← 3 (k)

    como (posMenor=3 = k=3), então não trocar

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 5:

  determinar o 5º menor elemento e colocá-lo na posição 5 (índice 4)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	4	5	9	6	8	7

  k ← 0, 1, 2, 3, 4

  j ← 5 (k+1), 6, 7

  posMenor ← 4 (k), 5

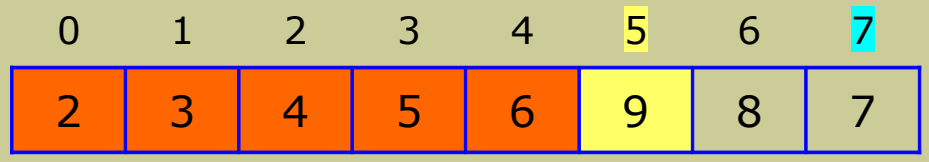
  como (posMenor=5 ≠ k=4), então trocar A[k=4] com A[posMenor=5]

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 6:

  determinar o 6º menor elemento e colocá-lo na posição 6 (índice 5)

  N = 8



  k ← 0, 1, 2, 3, 4, 5

  j ← 6 (k+1), 7

  posMenor ← 5 (k), 6, 7

  como (posMenor=7 ≠ k=5), então trocar A[k=5] com A[posMenor=7]

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 7:

  determinar o 7º menor elemento e colocá-lo na posição 7 (índice 6)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	4	5	6	7	8	9

  k ← 0, 1, 2, 3, 4, 5, 6

  j ← 7 (k+1)

  posMenor ← 6 (k)

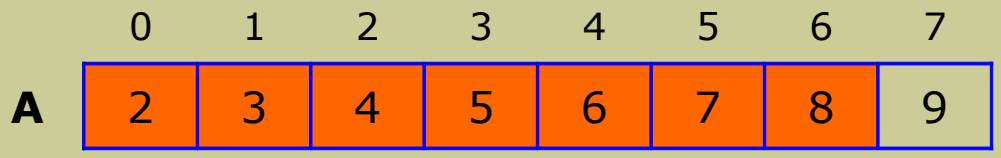
  como (posMenor=6 = k=6), então não trocar

Exemplo: ordenar por ordem crescente (simulação)

- Iteração 8:

  determinar o 8º menor elemento e colocá-lo na posição 8 (índice 7)

  N = 8



  k ← 0, 1, 2, 3, 4, 5, 6, 7

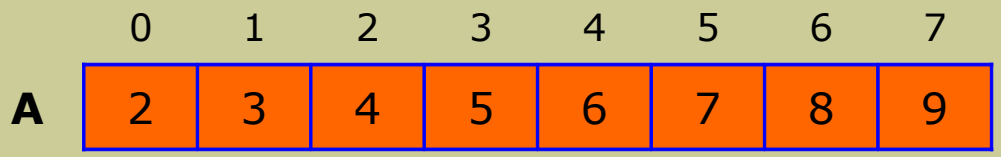
  - Como (k=7 < N-1=7) é falso, então termina o processo de ordenação
  - O elemento da última posição fica automaticamente ordenado

Exemplo: ordenar por ordem crescente (simulação)

- Terminado a execução do algoritmo

  estado final do array 1D **A** (ordenado)

  N = 8



Ordenação por Borbulhagem (“Bubble Sort”)

Definição

- Algoritmo iterativo (não recursivo)
- Este algoritmo consiste no seguinte:
  - comparar dois elementos consecutivos do array 1D,
    - se estiverem desordenados então trocá-los entre si
  - desta forma,
    - os maiores elementos tendem a deslocar-se para a direita e os menores para a esquerda do array 1D (ordenação por ordem crescente)
    - os maiores elementos tendem a deslocar-se para a esquerda e os menores para a direita do array 1D (ordenação por ordem decrescente)
  - o array 1D fica ordenado quando, após várias passagem pelo array 1D com pelo menos uma troca, não há qualquer troca na atual passagem

Colocar por ordem crescente

- Seja um array 1D **A** com **N** elementos
  - **primeira** passagem pelo array **A**:
    - coloca o **maior** elemento na N-ésima (última) posição
  - **segunda** passagem pelo array **A**:
    - coloca o **segundo maior** elemento na (N-1)-ésima (penúltima) posição
  - **terceira** passagem pelo array **A**:
    - coloca o **terceiro maior** elemento na (N-2)-ésima (antepenúltima) posição
  - ...
  - **k-ésima** passagem pelo array **A**:
    - coloca o **k-ésimo maior** elemento na (N-k)-ésima posição
  - ...
    - ...
  - **última** passagem pelo array **A**:
    - coloca o **menor** elemento na primeira posição

Colocar por ordem decrescente

- Seja um array 1D **A** com **N** elementos
  - **primeira** passagem pelo array **A**:
    - coloca o **menor** elemento na N-ésima (última) posição
  - **segunda** passagem pelo array **A**:
    - coloca o **segundo menor** elemento na (N-1)-ésima (penúltima) posição
  - **terceira** passagem pelo array **A**:
    - coloca o **terceiro menor** elemento na (N-2)-ésima (antepenúltima) posição
  - ...
  - **k-ésima** passagem pelo array **A**:
    - coloca o **k-ésimo menor** elemento na (N-(k-1))-ésima posição
  - ...
    - ...
  - **última** passagem pelo array **A**:
    - coloca o **maior** elemento na primeira posição

Implementação do algoritmo em C

Operação: ordenar por ordem crescente um array **A** com **N** inteiros

```
void ordenarBorbulhagem (int A[], int N)
{
  int  k, fim, numTrocas;
  fim = N – 1;
  do{
     numTrocas = 0;
     for (k = 0; k < fim; k++)
        if (A[k] > A[k+1]) {
           trocar(&A[k], &A[k+1]);
           numTrocas++;
        }
     fim = fim – 1;
  }while (numTrocas > 0);
}
```

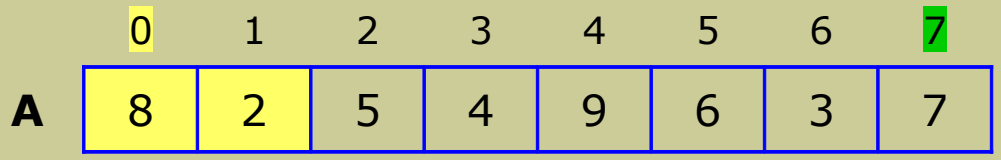
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 1:

  N = 8

  fim ← 7 (N - 1)



  k ← 0

  como (A[k=0]=8 > A[k+1=1]=2) então trocá-los

  numTrocas ← 0, 1

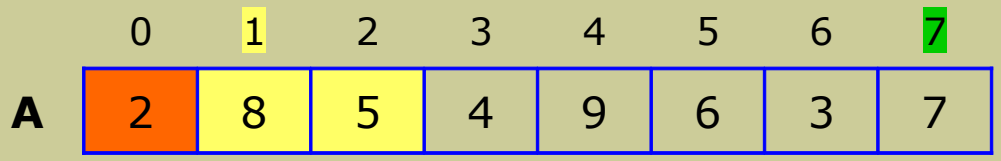
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 2:

  N = 8

  fim = 7



  k ← 0, 1

  como (A[k=1]=8 > A[k+1=2]=5) então trocá-los

  numTrocas ← 1, 2

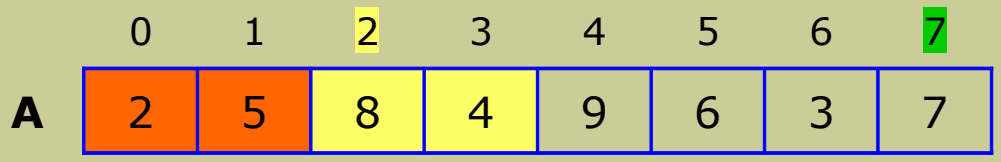
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 3:

  N = 8

  fim = 7



  k ← 0, 1, 2

  como (A[k=2]=8 > A[k+1=3]=4) então trocá-los

  numTrocas ← 2, 3

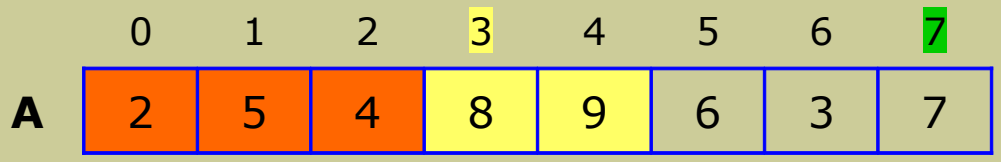
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 4:

  N = 8

  fim = 7



  k ← 0, 1, 2, 3

  como (A[k=3]=8 ≤ A[k+1=4]=9) então não trocar

  numTrocas ← 3

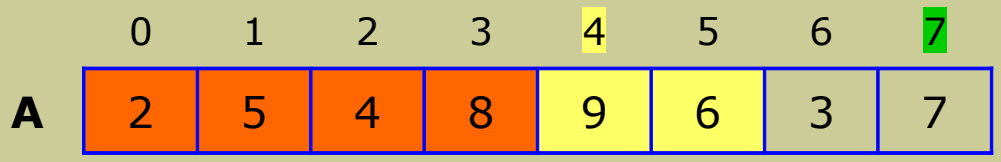
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 5:

  N = 8

  fim = 7



  k ← 0, 1, 2, 3, 4

  como (A[k=4]=9 > A[k+1=5]=6) então trocá-los

  numTrocas ← 3, 4

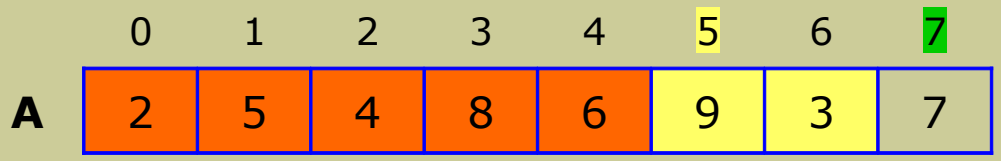
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 6:

  N = 8

  fim = 7



  k ← 0, 1, 2, 3, 4, 5

  como (A[k=5]=9 > A[k+1=6]=3) então trocá-los

  numTrocas ← 4, 5

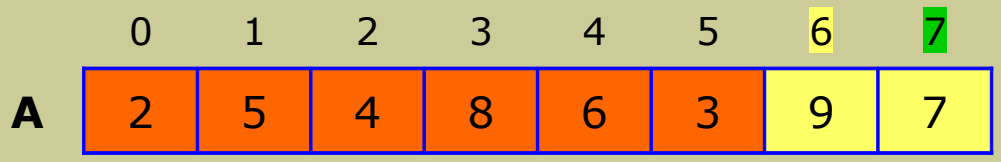
Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 7:

  N = 8

  fim = 7



  k ← 0, 1, 2, 3, 4, 5, 6

  como (A[k=6]=9 > A[k+1=7]=7) então trocá-los

  numTrocas ← 5, 6

Exemplo: ordenar por ordem crescente (simulação)

- Primeira passagem pelo array **A**

  Iteração 8:

  N = 8

  fim = 7

	0	1	2	3	4	5	6	**7**
A	2	5	4	8	6	3	7	9

  k ← 0, 1, 2, 3, 4, 5, 6, 7

  como (k=7 < 7=fim) é falso,

  então está concluída a primeira passagem

  fim ← 6 (fim−1)

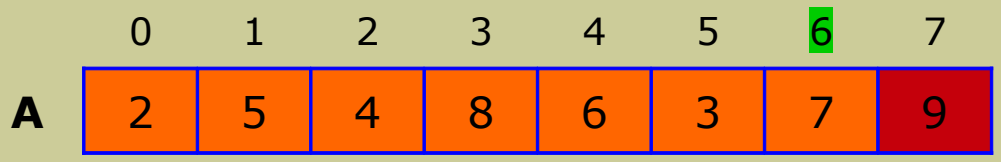
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a primeira passagem

  o elemento na posição 7 encontra-se na posição final correta (ordenado)

  N = 8

  fim = 6



- Como (numTrocas=6 > 0),

  então realizar uma nova passagem pelo array **A** atualizado

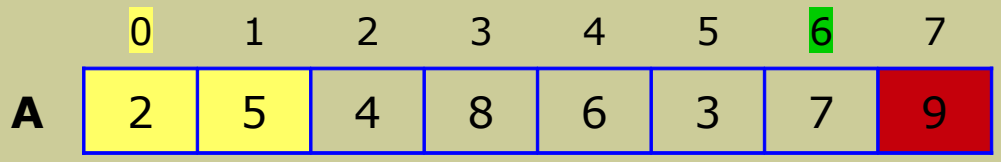
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 1:

N = 8

fim = 6



k ← 0

como (A[k=0]=2 ≤ A[k+1=1]=5) então não trocar

numTrocas ← 0

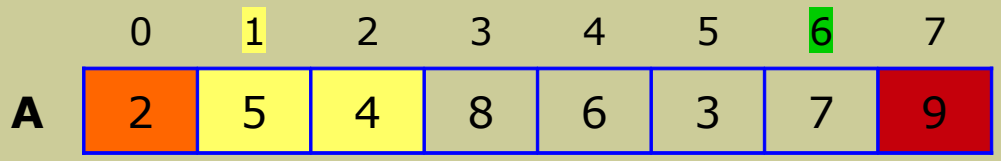
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

  Iteração 2:

  N = 8

  fim = 6



  k ← 0, 1

  como (A[k=1]=5 > A[k+1=2]=4) então trocá-los

  numTrocas ← 0, 1

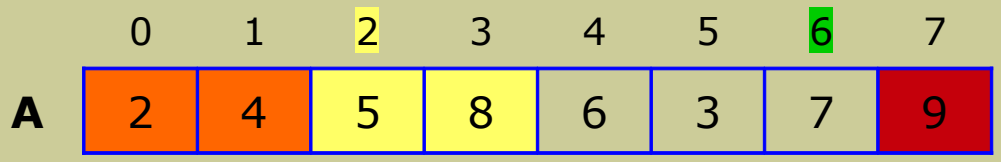
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 3:

N = 8

fim = 6



k ← 0, 1, 2

como (A[k=2]=5 ≤ A[k+1=3]=8) então não trocar

numTrocas ← 1

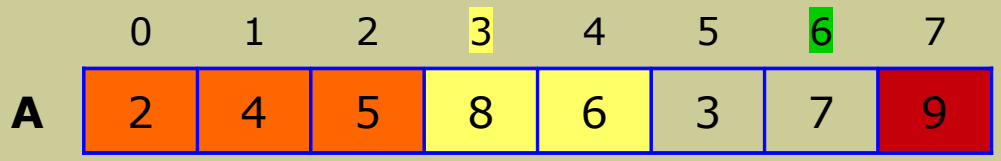
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 4:

N = 8

fim = 6



k ← 0, 1, 2, 3

como (A[k=3]=8 > A[k+1=4]=6) então trocá-los

numTrocas ← 1, 2

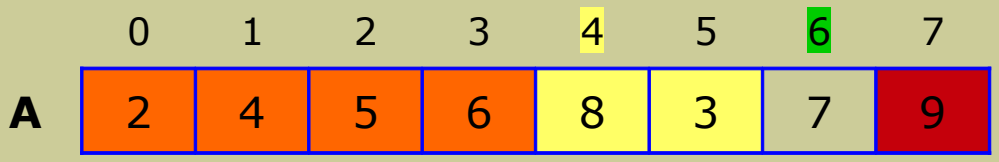
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 5:

N = 8

fim = 6



k ← 0, 1, 2, 3, 4

como (A[k=4]=8 > A[k+1=5]=3) então trocá-los

numTrocas ← 2, 3

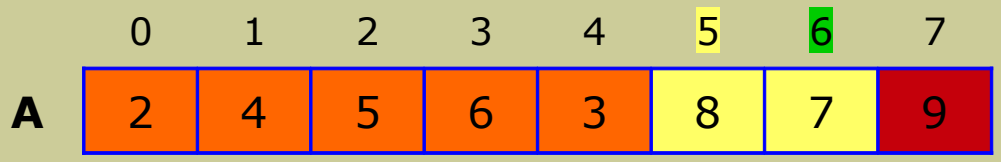
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 6:

N = 8

fim = 6



k ← 0, 1, 2, 3, 4, 5

como (A[k=5]=8 > A[k+1=6]=7) então trocá-los

numTrocas ← 3, 4

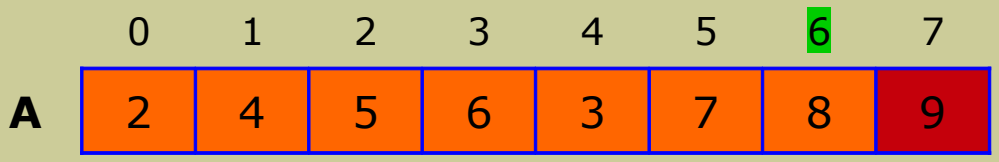
Exemplo: ordenar por ordem crescente (simulação)

- Segunda passagem pelo array **A**

Iteração 7:

N = 8

fim = 6



k ← 0, 1, 2, 3, 4, 5, 6

como (k=6 < 6=fim) é falso,

então está concluída a segunda passagem

fim ← 5 (fim−1)

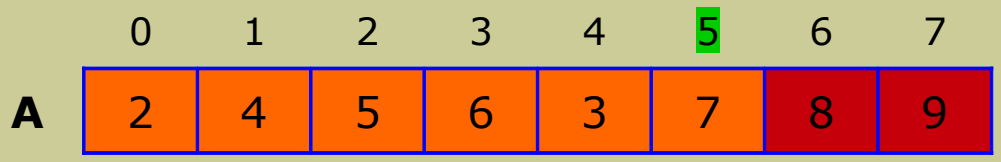
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a segunda passagem

  o elemento na posição 6 também se encontra na posição final correta (ordenado)

  N = 8

  fim = 5



- Como (numTrocas=4 > 0),

  então realizar uma nova passagem

Exemplo: ordenar por ordem crescente (simulação)

- Terceira passagem pelo array **A**

Iteração 1-3:

N = 8

fim = 5

	0	1	2	3	4	5	6	7
A	2	4	5	6	3	7	8	9

k ← 0

como (A[k=0]=2 ≤ A[k+1=1]=4) então não trocar

numTrocas ← 0

k ← 0, 1

como (A[k=1]=4 ≤ A[k+1=2]=5) então não trocar

k ← 0, 1, 2

como (A[k=2]=5 ≤ A[k+1=3]=6) então não trocar

numTrocas ← 0

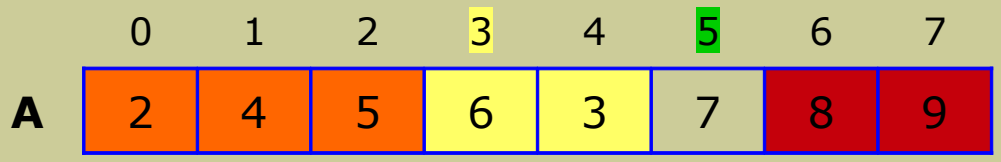
Exemplo: ordenar por ordem crescente (simulação)

- Terceira passagem pelo array **A**

  Iteração 4:

  N = 8

  fim = 5



  k ← 0, 1, 2, 3

  como (A[k=3]=6 > A[k+1=4]=3) então trocá-los

  numTrocas ← 0, 1

Exemplo: ordenar por ordem crescente (simulação)

- Terceira passagem pelo array **A**

Iteração 5:

N = 8

fim = 5

	0	1	2	3	4	5	6	7
A	2	4	5	3	6	7	8	9

k ← 0, 1, 2, 3, 4

como (A[k=4]=6 ≤ A[k+1=5]=7) então não trocar

numTrocas ← 1

Exemplo: ordenar por ordem crescente (simulação)

- Terceira passagem pelo array **A**

  Iteração 6:

  N = 8

  fim = 5

	0	1	2	3	4	5	6	7
A	2	4	5	3	6	7	8	9

  k ← 0, 1, 2, 3, 4, 5

  como (k=5 < 5=fim) é falso,

  então está concluída a terceira passagem

  fim ← 4 (fim−1)

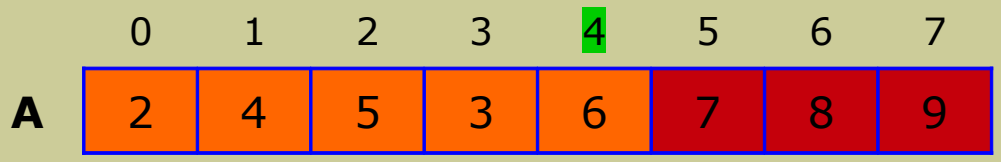
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a terceira passagem

  o elemento na posição 5 também se encontra na posição final correta (ordenado)

  N = 8

  fim = 4



- Como (numTrocas=1 > 0),

  então realizar uma nova passagem

Exemplo: ordenar por ordem crescente (simulação)

- Quarta passagem pelo array **A**

Iteração 1-2:

N = 8

fim = 4

	0	1	2	3	4	5	6	7
A	2	4	5	3	6	7	8	9

k ← 0

como (A[k=0]=2 ≤ A[k+1=1]=4), então não trocar

numTrocas ← 0

k ← 0, 1

como (A[k=1]=4 ≤ A[k+1=2]=5), então não trocar

numTrocas ← 0

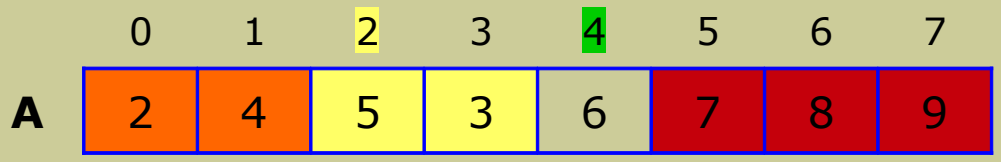
Exemplo: ordenar por ordem crescente (simulação)

- Quarta passagem pelo array **A**

  Iteração 3:

  N = 8

  fim = 4



  k ← 0, 1, 2

  como (A[k=2]=5 ≤ A[k+1=3]=3), então trocá-los

  numTrocas ← 0, 1

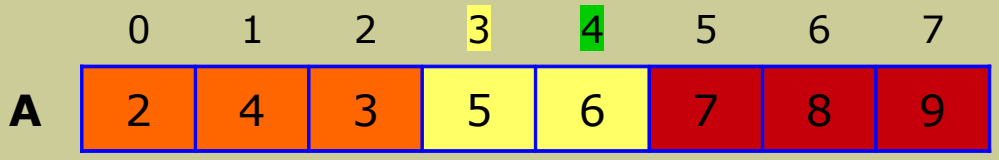
Exemplo: ordenar por ordem crescente (simulação)

- Quarta passagem pelo array **A**

  Iteração 4:

  N = 8

  fim = 4



  k ← 0, 1, 2, 3

  como (A[k=3]=5 ≤ A[k+1=4]=6), então não trocar

  numTrocas ← 1

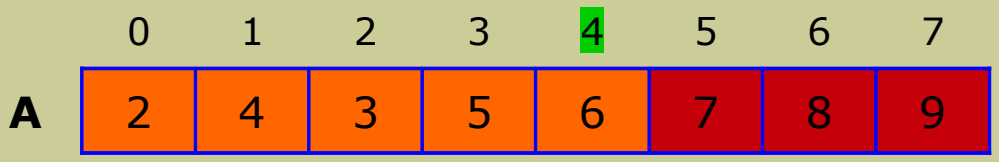
Exemplo: ordenar por ordem crescente (simulação)

- Quarta passagem pelo array **A**

  Iteração 5:

  N = 8

  fim = 4



  k ← 0, 1, 2, 3, 4

  como (k=4 < 4=fim) é falso,

  então está concluída a quarta passagem

  fim ← 3 (fim−1)

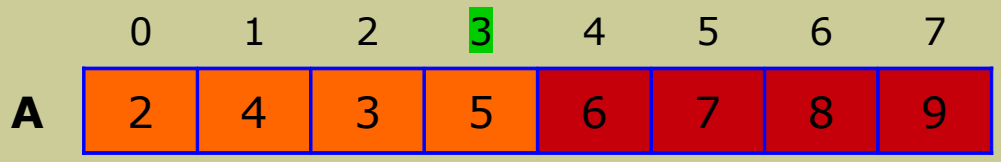
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a quarta passagem

  o elemento na posição 4 também se encontra na posição final correta (ordenado)

  N = 8

  fim = 3



- Como (numTrocas=1 > 0),

  então realizar uma nova passagem

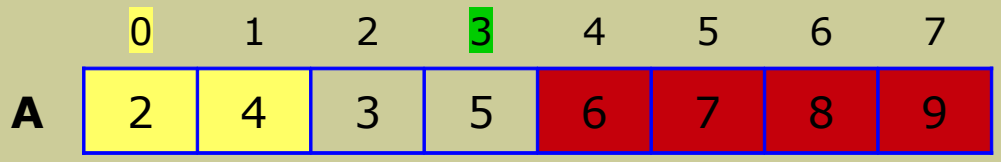
Exemplo: ordenar por ordem crescente (simulação)

- Quinta passagem pelo array **A**

  Iteração 1:

  N = 8

  fim = 3



  k ← 0

  como (A[k=0]=2 ≤ A[k+1=1]=4) então não trocar

  numTrocas ← 0

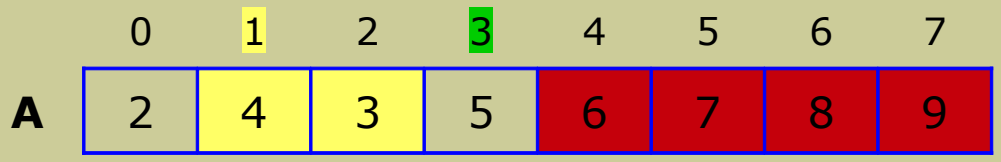
Exemplo: ordenar por ordem crescente (simulação)

- Quinta passagem pelo array **A**

  Iteração 2:

  N = 8

  fim = 3



  k ← 0, 1

  como (A[k=1]=4 > A[k+1=2]=3), então trocá-los

  numTrocas ← 0, 1

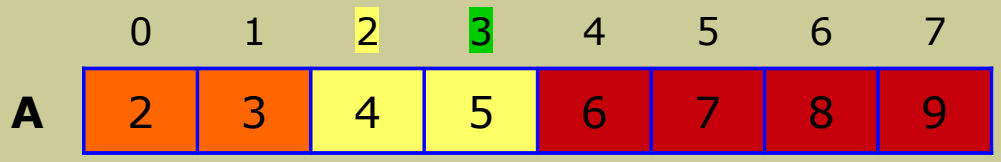
Exemplo: ordenar por ordem crescente (simulação)

- Quinta passagem pelo array **A**

  Iteração 3:

  N = 8

  fim = 3



  k ← 0, 1, 2

  como (A[k=2]=4 ≤ A[k+1=3]=5), então não trocar

  numTrocas ← 1

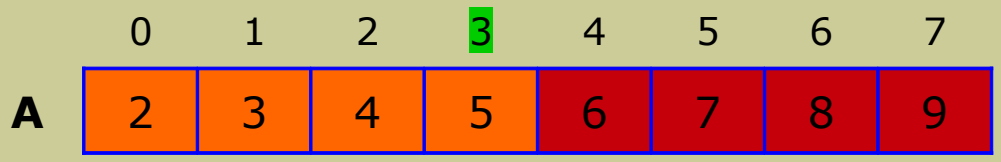
Exemplo: ordenar por ordem crescente (simulação)

- Quinta passagem pelo array **A**

Iteração 4:

N = 8

fim = 3



k ← 0, 1, 2, 3

como (k=3 < 3=fim) é falso,

então está concluída a quinta passagem

Fim ← 2 (fim−1)

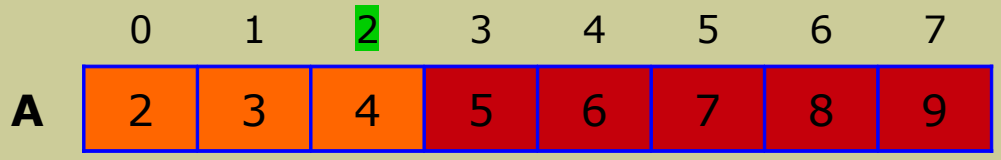
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a quinta passagem

  o elemento na posição 3 também se encontra na posição final correta (ordenado)

  N = 8

  fim = 2 (fim – 1)



- Como (numTrocas=1 > 0),

  então realizar uma nova passagem

Exemplo: ordenar por ordem crescente (simulação)

- Sexta passagem pelo array **A**

Iteração 1-2:

N = 8

fim = 2

	0	1	2	3	4	5	6	7
A	2	3	4	5	6	7	8	9

k ← 0

como (A[k=0]=2 > A[k+1=1]=3) então não trocar

numTrocas ← 0

k ← 0, 1

como (A[k=1]=3 ≤ A[k+1=2]=4), então não trocar

numTrocas ← 0

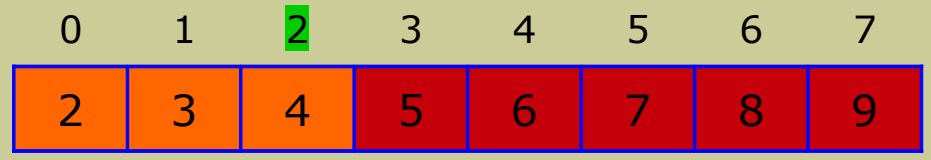
Exemplo: ordenar por ordem crescente (simulação)

- Sexta passagem pelo array **A**

Iteração 3:

N = 8

fim = 2



k ← 0, 1, 2

como (k=2 < 2=fim) é falso,

então está concluída a sexta passagem

fim ← 1 (fim – 1)

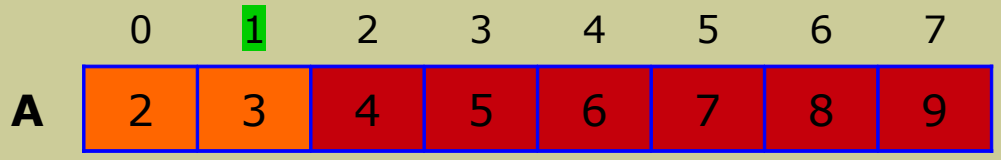
Exemplo: ordenar por ordem crescente (simulação)

- Estado final do array **A** após a sexta passagem

  o elemento na posição 2 também se encontra na posição final correta (ordenado)

  N = 8

  fim = 1 (fim – 1)



  - Como (numTrocas=0 > 0) é falso,

    então terminar processo: o array 1D A está ordenado

  - Os elementos nas posições 0 e 1 encontram-se nas posições finais corretas (ordenados)

Exemplo: ordenar por ordem crescente (simulação)

- Terminado a execução do algoritmo

  estado final do array **A** (ordenado)

  N = 8

	0	1	2	3	4	5	6	7
A	2	3	4	5	6	7	8	9

Ordenação rápida ("Quicksort")

Definição

- Algoritmo recursivo que se baseia num princípio muito simples que, quando aplicado recursivamente, ordena um array 1D **A** com **N** elementos
- Este princípio baseia-se no seguinte:
  1. escolher um elemento do array **A**, chamado de **pivot** (por exemplo, A[0])
  2. determinar posição final correta do **pivot** no array **A** ordenado
  3. colocar o **pivot** na sua posição final correta no array **A** ordenado
  4. dividir o array A em 3 partes (esquerda, pivot e direita), em que:
     - parte **esquerda** com os elementos menores do que o **pivot**
     - **pivot** na posição final correta (ordenado) no array **A**
     - parte **direita** com os elementos maiores do que o **pivot**
- Ao aplicar-se sucessivamente este princípio às partes esquerda e direita, o processo recursivo quando terminar apresenta o array **A** ordenado

Algoritmo

- Ordenar um array 1D **A** com **N** elementos entre as posições **inicio** e **fim**
  - os primeiros valores (início do algoritmo) são:
    - inicio = 0 (índice da *primeira* posição do array A)
    - fim = N - 1 (índice da *última* posição do array A)
- Passos do algoritmo

1. determinar a posição final **pos** do elemento **pivot** (A[inicio]) no array **A**
2. colocar o **pivot** na posição **pos**, ficando o array **A** dividido em 3 partes:
   - esquerda = (A[inicio], ..., A[pos-1])
   - pivot na posição **pos**
   - direita = (A[pos+1], ..., A[fim])
3. aplicar o algoritmo recursivamente à parte esquerda: (A[inicio], ..., A[pos-1])
4. aplicar o algoritmo recursivamente à parte direita: (A[pos+1], ..., A[fim])

- Algoritmo com partes
  - iterativa (passos 1 e 2) e
  - recursiva (passos 3 e 4), que chama a parte iterativa

Implementação do algoritmo principal (recursivo) em C

Operação: ordenar por ordem crescente um array de **A** entre os índices **inicio** até **fim**

```
void ordenarQuicksort (int A[], int inicio, int fim)
{
  int  pos;      //  pos = posição final do pivot A[inicio] em A ordenado
  if (inicio < fim) {
     pos = determinarPosicaoPivot(A, inicio, fim);  // colocar o pivot ordenado
     ordenarQuicksort(A, inicio, pos-1);     // ordenar a sub-array esquerdo
     ordenarQuicksort(A, pos+1, fim);        // ordenar a sub-array direito
  }
}
```

- Notas:

  Se (inicio ≥ fim) então o sub-array de A já está ordenado, pois

  - tem apenas um elemento (inicio = fim), ou
  - está vazio (inicio > fim)

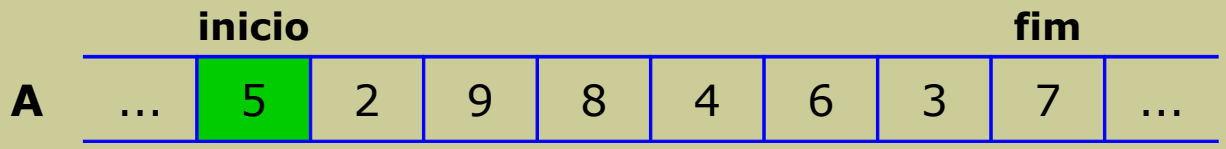
Algoritmo (parte iterativa)

- Pretende-se determinar a posição final do elemento **pivot** num array 1D **A** entre as posições **inicio** e **fim**

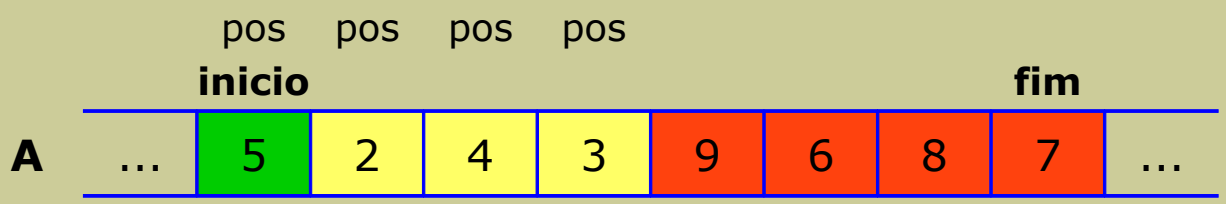
1. escolher o elemento pivot do array **A**: A[inicio]
2. determinar a posição final correta do pivot no array **A** ordenado, **pos**
   - colocar nas posições imediatamente a seguir a inicio os menores do que o **pivot**
   - colocar nas posições imediatamente antes de fim os maiores do que o **pivot**
   - desta forma, o array **A** fica dividido em 3 partes:
     - **pivot** na posição inicio
     - todos os elementos *menores* do que o **pivot** nas posições entre inicio+1 e pos
     - todos os elementos *maiores* do que o **pivot** nas posições entre pos+1 e fim
3. colocar o **pivot** na posição **pos** do array **A** ordenado (sua posição final correta)
   - trocar o elemento da posição **pos** ("ultimo" menor que o pivot) com o **pivot** (inicio)

Algoritmo – parte iterativa

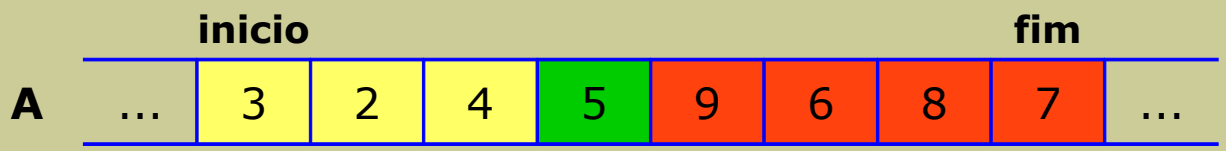
- Exemplo:
  - Passo 1:



  - Passo 2:



  - Passo 3:



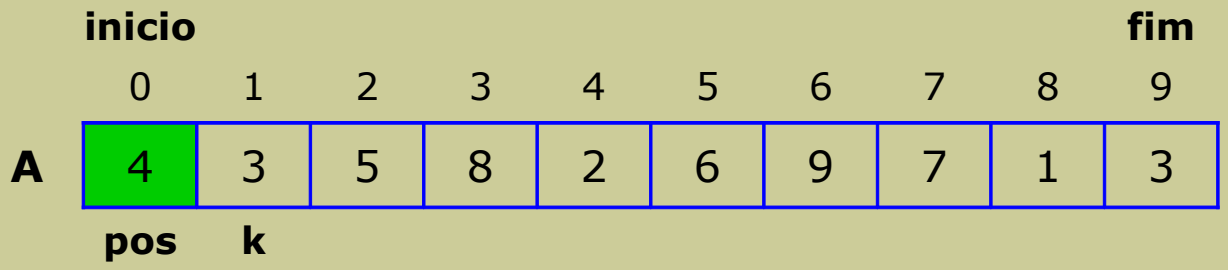
Implementação da parte iterativa do algoritmo em C

1. determinar a posição final do pivot (**pos**) no array de **A** desde **inicio** até **fim**
2. colocar o pivot na sua posição final **pos**

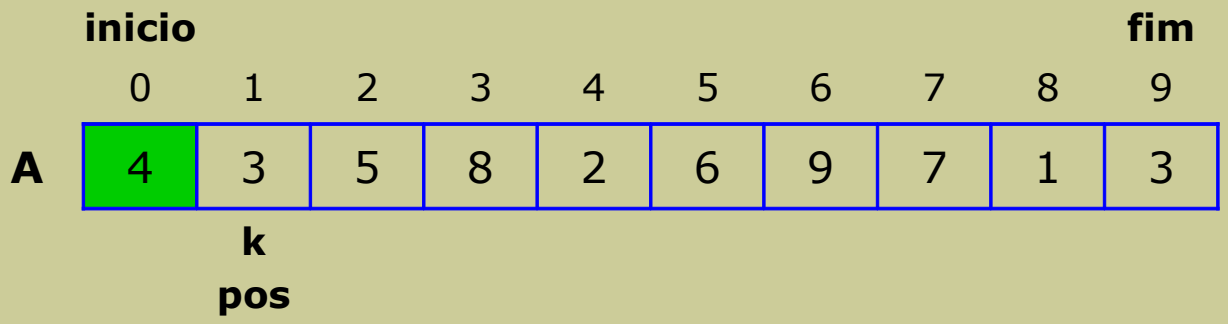
```
int determinarPosicaoPivot (int A[], int inicio, int fim)
{
  int  k, pos;
  pos = inicio;   // pos = posição final do pivot (A[inicio]) em A ordenado
  for (k = inicio + 1; k <= fim; k++) {
     if (A[k] < A[inicio]) {
        pos++;
        if (k != pos)
           trocar(&A[k], &A[pos]);
     }
  }
  trocar(&A[inicio], &A[pos]);
  return pos;
}
```

Exemplo

- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



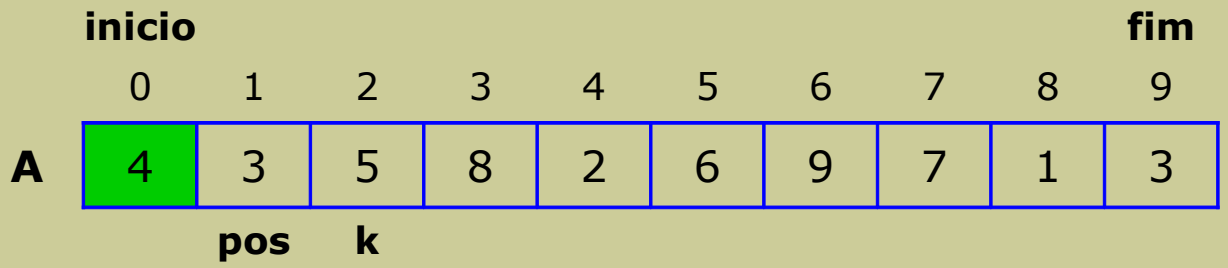
  - como (A[k=1]=3 < A[inicio=0]=4), então
    - pos = pos + 1 = 1
    - como (k=1 = pos=1) então
      - não trocar



      - passar à próxima iteração (k = 2)

Exemplo

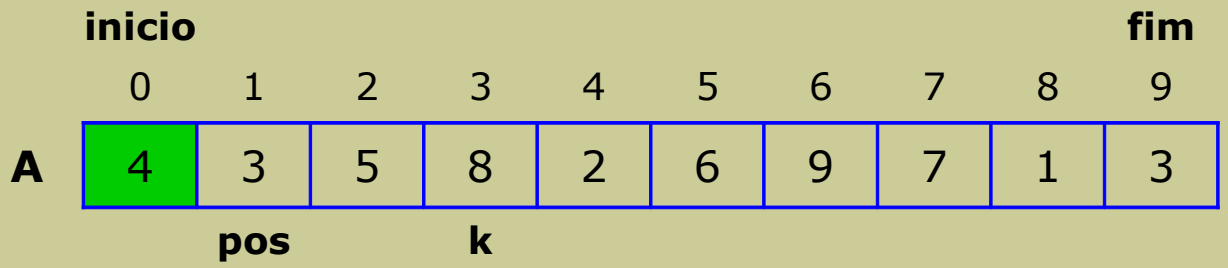
- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



  - Como (A[k=2]=5 > A[inicio=0]=4) então
    - passar à próxima iteração (k = 3)

Exemplo

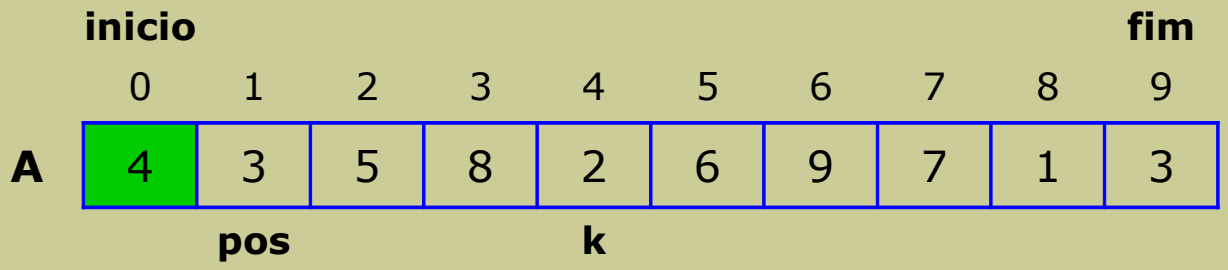
- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



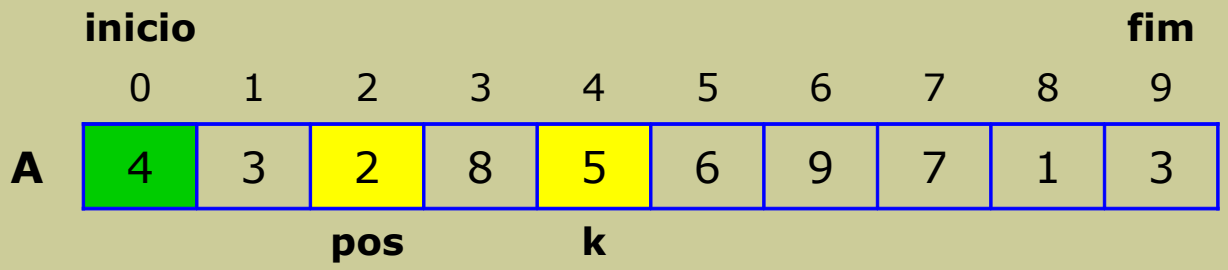
  - Como (A[k=3]=8 > A[inicio=0]=4) então
    - passar à próxima iteração (k = 4)

Exemplo

- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



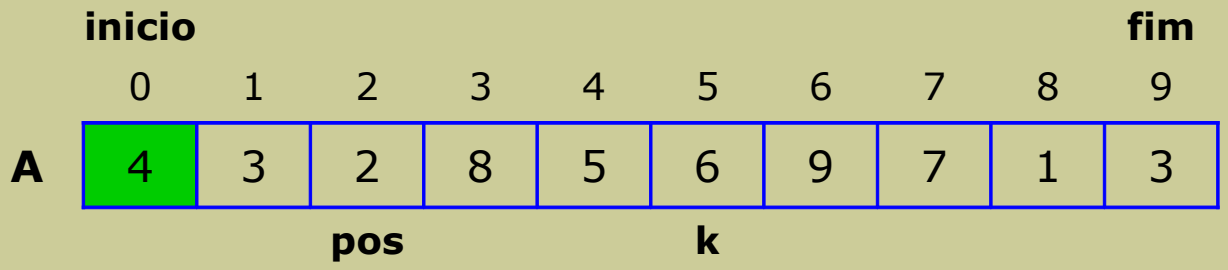
  - Como (A[k=4]=2 < A[inicio=0]=4) então
    - pos = pos + 1 = 2
    - Como (k=4 ≠ pos=2) então
      - trocar(A[k], A[pos])



      - passar à próxima iteração (k = 5)

Exemplo

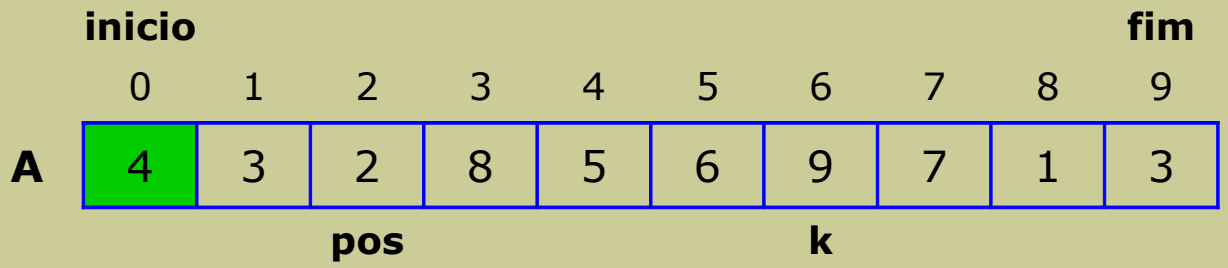
- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



  - Como (A[k=5]=6 > A[inicio=0]=4) então
    - passar à próxima iteração (k = 6)

Exemplo

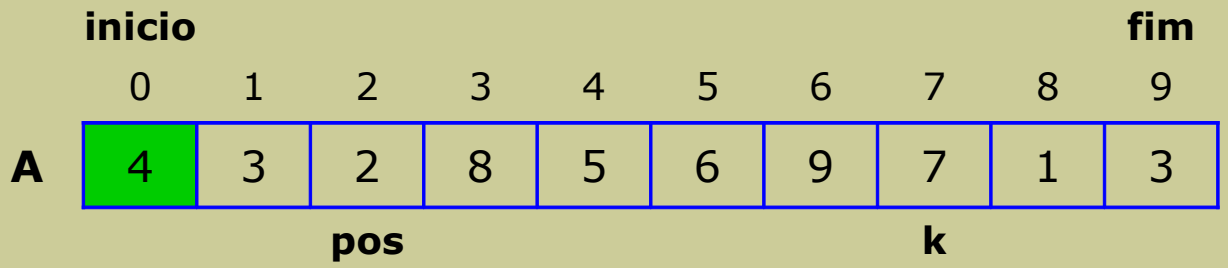
- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



  - Como (A[k=6]=9 > A[inicio=0]=4) então
    - passar à próxima iteração (k = 7)

Exemplo

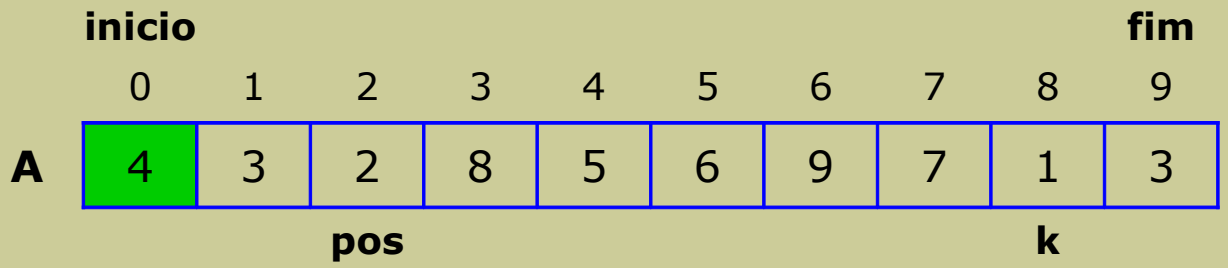
- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



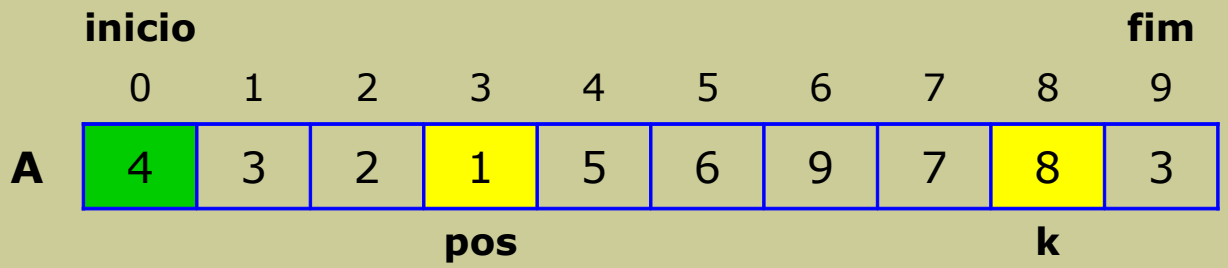
  - Como (A[k=7]=7 > A[inicio=0]=4) então
    - passar à próxima iteração (k = 8)

Exemplo

- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



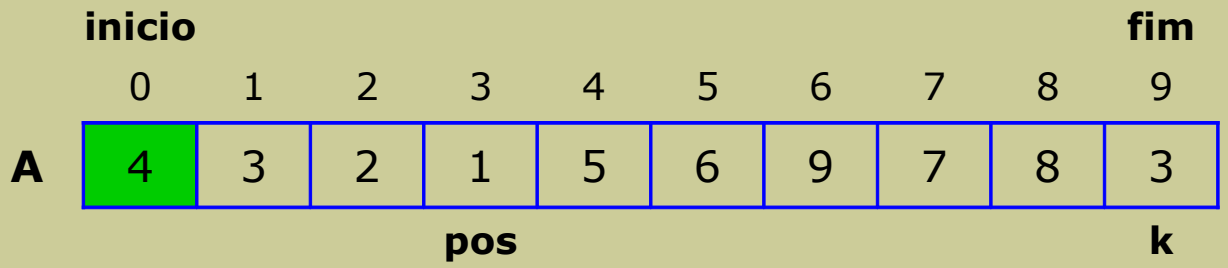
  - Como (A[k=8]=1 < A[inicio=0]=4) então
    - pos = pos + 1 = 3
    - Como (k=8 ≠ pos=3) então
      - trocar(A[k=8], A[pos=3])



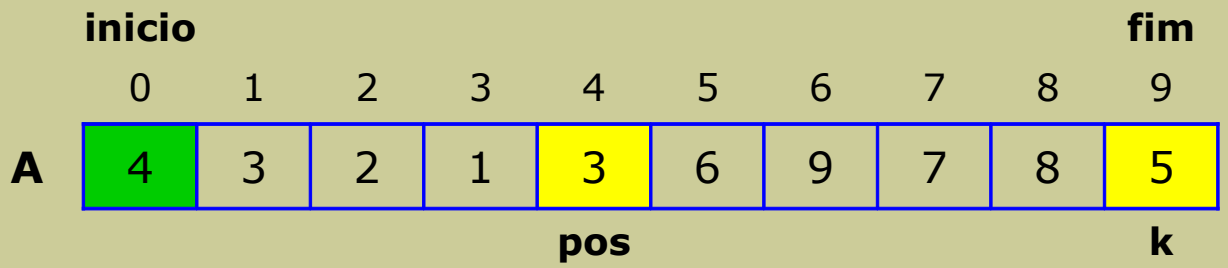
      - passar à próxima iteração (k = 9)

Exemplo

- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4



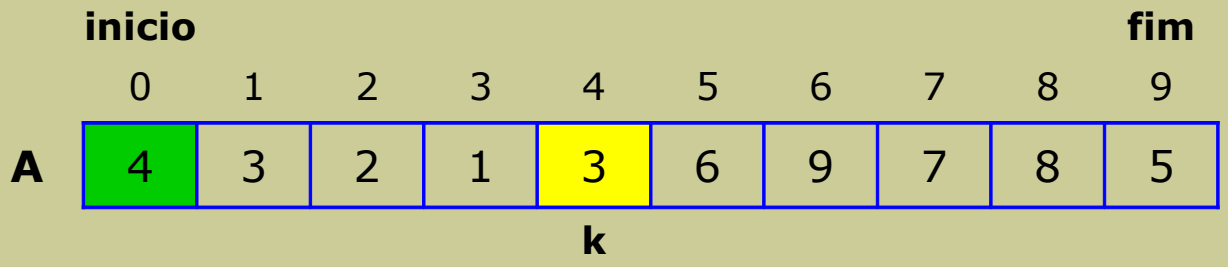
  - Como (A[k=9]=3 < A[inicio=0]=4) então
    - pos = pos + 1 = 4
    - Como (k=9 ≠ pos=4) então
      - trocar(A[k=9], A[pos=4])



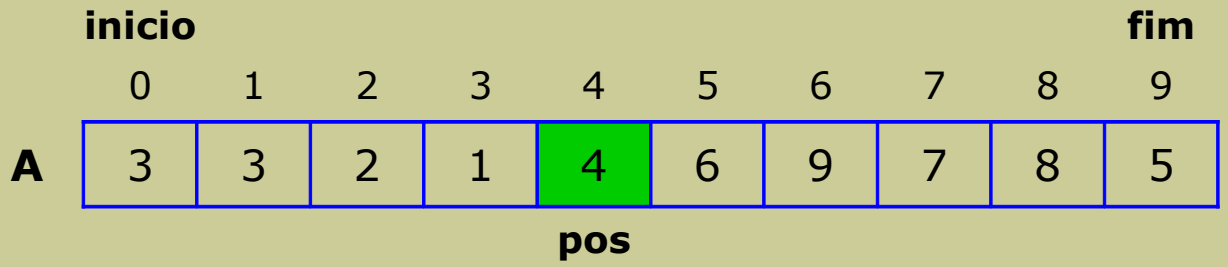
      - passar à próxima iteração (k = 10)

Exemplo

- Simular o subprograma auxiliar determinarPosicaoPivot
  - pivot = A[inicio=0] = 4

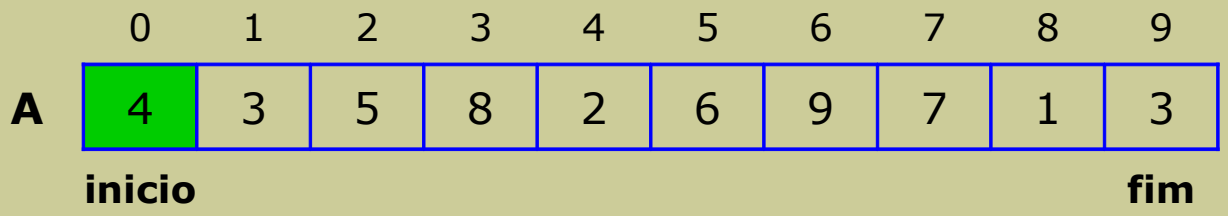


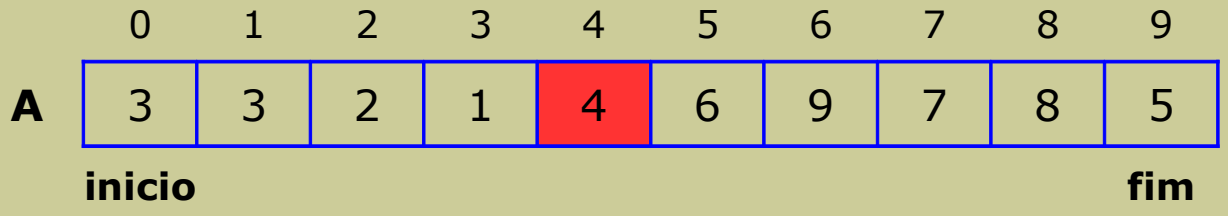
  - Como (k=10 > fim=9) então
    - terminar a terminar esta fase
    - trocar(A[inicio], A[pos])



Exemplo

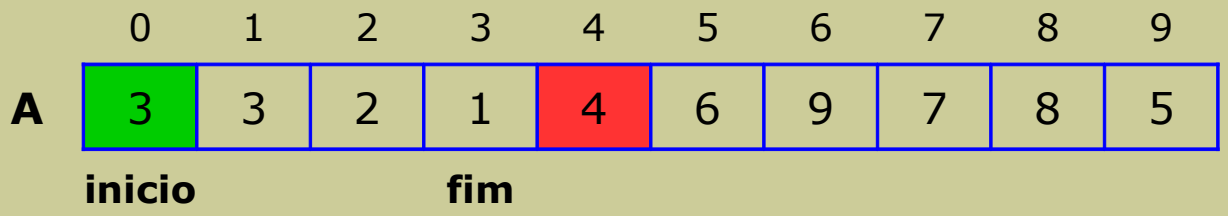
- Simular o subprograma principal ordenarQuicksort

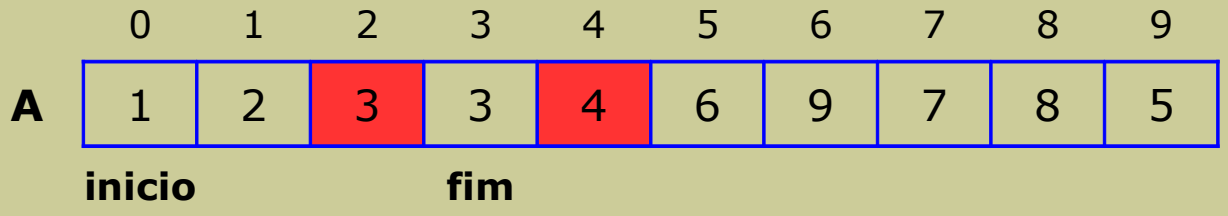




Exemplo

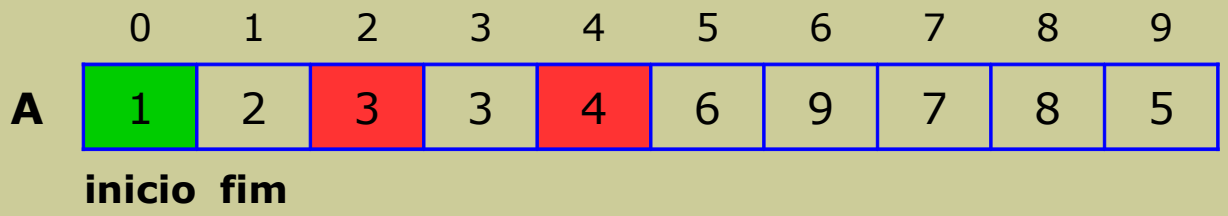
- Simular o subprograma principal ordenarQuicksort

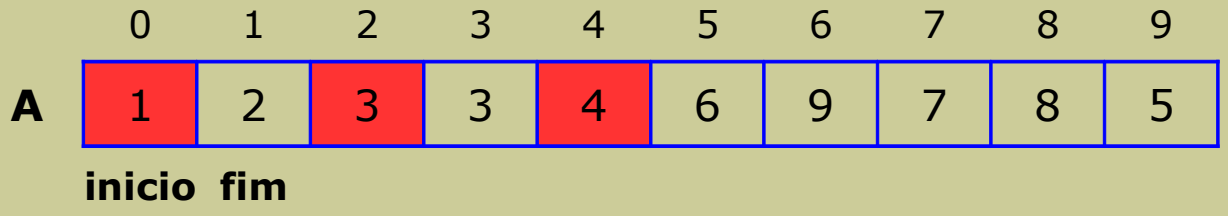




Exemplo

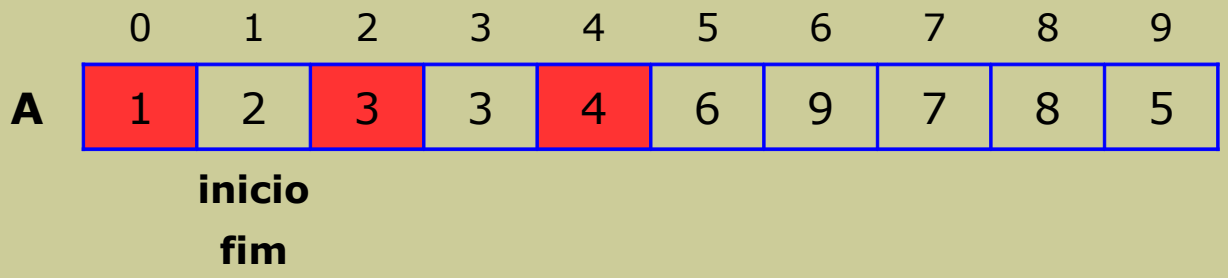
- Simular o subprograma principal ordenarQuicksort

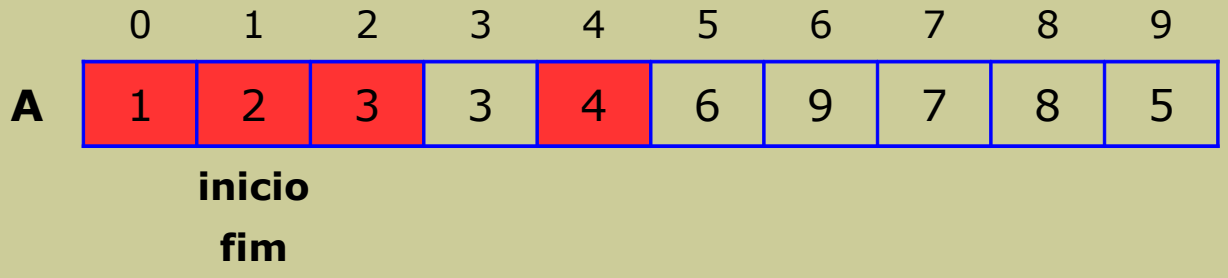




Exemplo

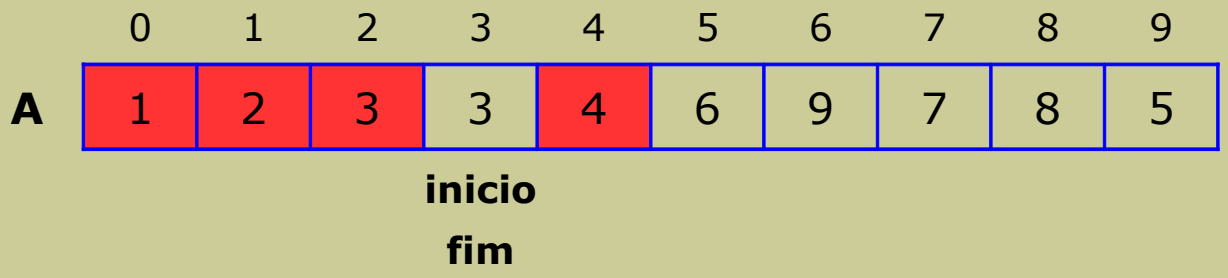
- Simular o subprograma principal ordenarQuicksort

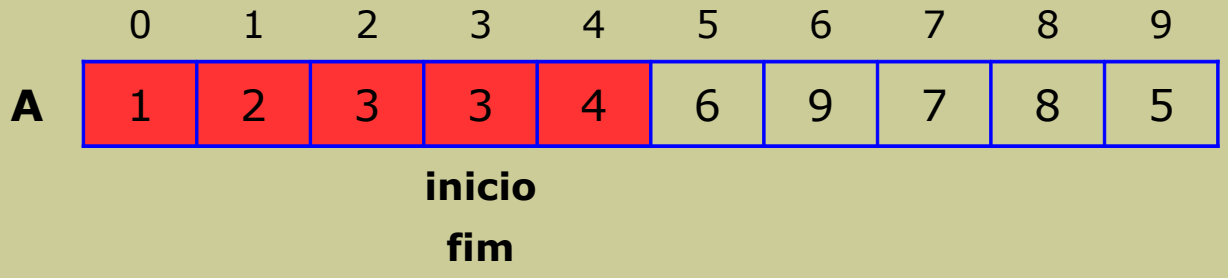




Exemplo

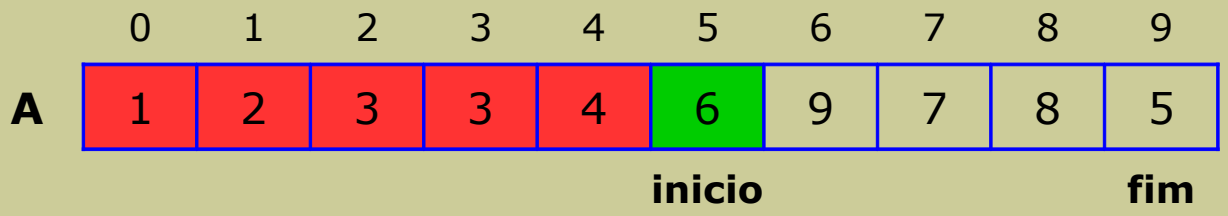
- Simular o subprograma principal ordenarQuicksort

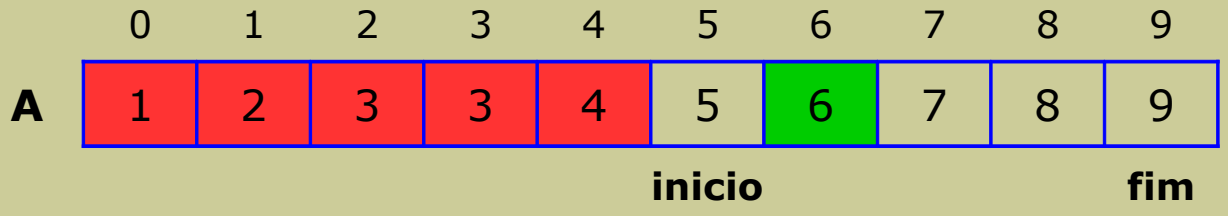




Exemplo

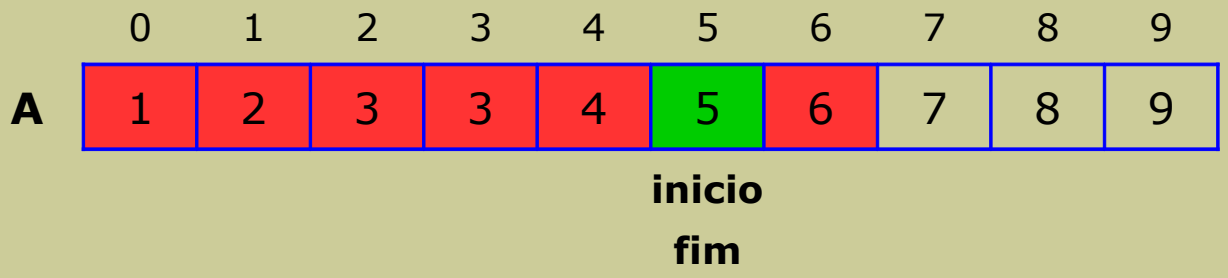
- Simular o subprograma principal ordenarQuicksort

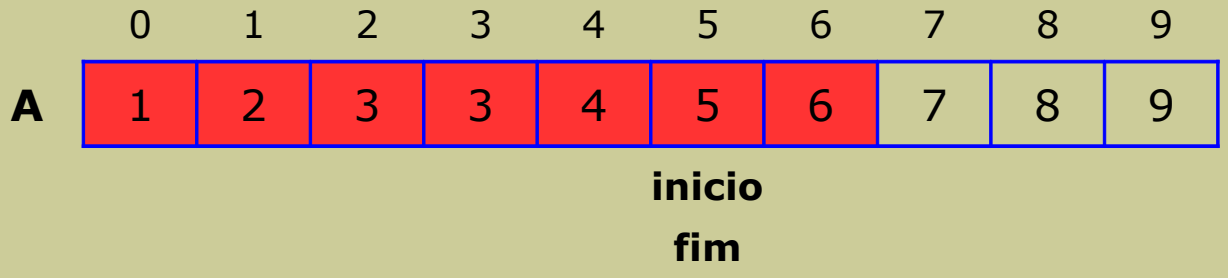




Exemplo

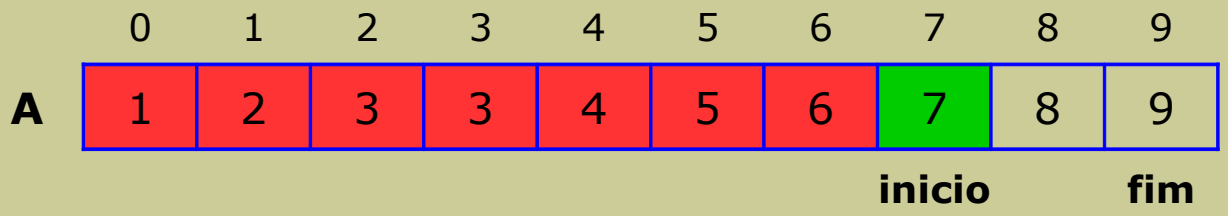
- Simular o subprograma principal ordenarQuicksort

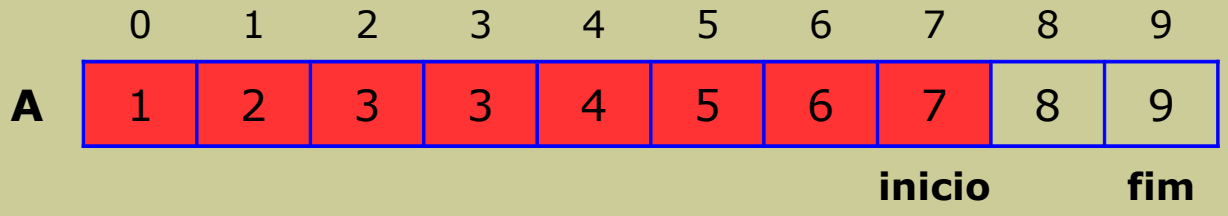




Exemplo

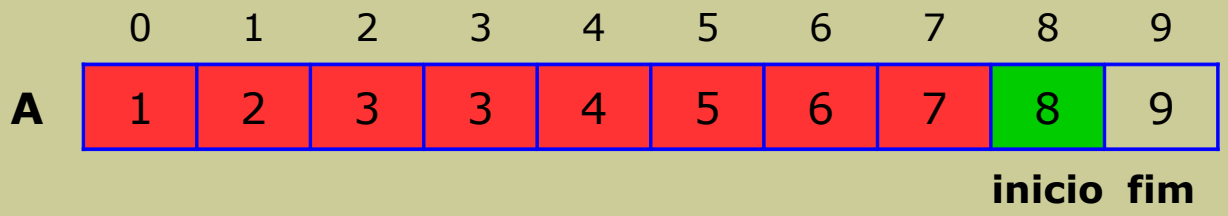
- Simular o subprograma principal ordenarQuicksort

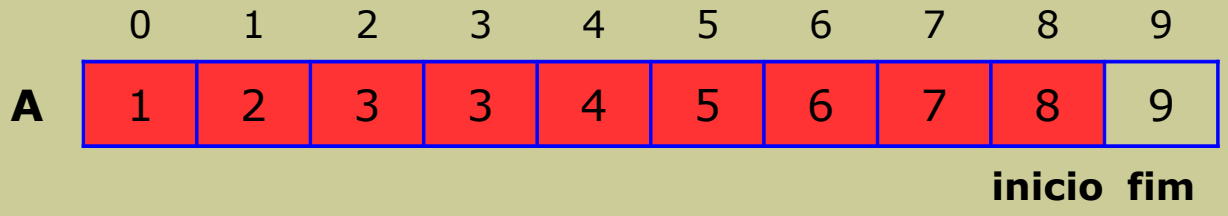




Exemplo

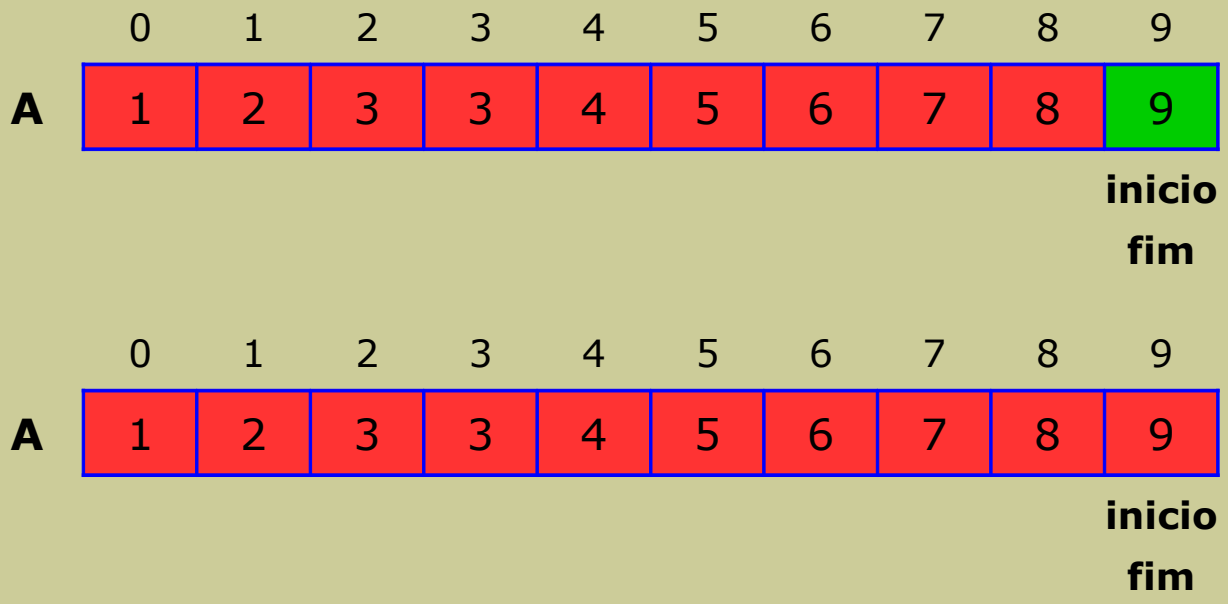
- Simular o subprograma principal ordenarQuicksort





Exemplo

- Simular o subprograma principal ordenarQuicksort



Ordenação por fusão (“Merge Sort”)

Definição

- Algoritmo recursivo
- A ideia é a seguinte
  - dividir o array 1D em vários sub-arrays de menor dimensão,
  - ordenar estes sub-arrays separadamente e,
  - fundir os dois sub-arrays num array único
- O algoritmo consiste no seguinte
  - dividir sucessivamente o array 1D ao meio até que se obtenham sub-arrays com apenas um elemento (que está ordenado)
  - fundir os sub-arrays num array único

Algoritmo

- Pretende-se ordenar um array 1D **A** com **N** elementos entre as posições **a** e **c**
- O algoritmo é composto por duas partes

  Parte 1: ordenar o array **A** entre as posições

  - **a** e **b** (A[a], A[a+1], ..., A[b]), e
  - **b+1** e **c** (A[b+1], A[b+2], ..., A[c])

  Parte 2: fundir o array **A** entre as posições **a** e **c** (**A** está ordenado entre estas posições)

  - **a** e **b** (A[a] ≤ A[a+1] ≤ ... ≤ A[b]), e
  - **b+1** e **c** (A[b+1] ≤ A[b+2] ≤ ... ≤ A[c])

Subprograma principal em C

Operação: ordenar por ordem crescente um array **A** entre os índices **inicio** e **fim**

```
void ordenarFusao (int A[], int inicio, int fim)
{
  int  meio;
  if (inicio < fim) {
     meio = (inicio + fim) / 2;
     ordenarFusao(A, inicio, meio);
     ordenarFusao(A, meio+1, fim);
     fundir(A, inicio, meio, fim);
  }
}
```

- Notas:

  Se (inicio ≥ fim) então o array de A entre os índices inicio e fim está ordenado, pois

  - tem apenas um elemento (inicio = fim), ou
  - está vazio (inicio > fim)

Subprograma auxiliar em C

```
void fundir (int A[], int inicio, int meio, int fim)
{
  int  esq, dir, k, i, Aux[fim-inicio+1];
  esq = inicio;
  dir = meio + 1;
  k = 0;  // controlo do array Aux
  while (esq <= meio && dir <= fim) {
     if (A[esq] < A[dir]) {
        Aux[k] = A[esq];
        esq++;
     }
     else {
        Aux[k] = A[dir];
        dir++;
     }
     k++;
```

```
} // while
```

Subprograma auxiliar em C (cont)

```
  ...
  if (esq > meio)
     for (i = dir; i <= fim; i++) {
        Aux[k] = A[i];
        k++;
     }
  else  // dir > fim
     for (i = esq; i <= meio; i++) {
        Aux[k] = A[i];
        k++;
     }
  //  array Aux construído e ordenado; agora passar Aux para o array A
  for (i = 0; i < k; i++)
     A[inicio+i] = Aux[i];
}
```