INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I.

Algoritmos de ordenação VI

Prof. Eiji Adachi M. Barbosa

Objetivos

- Dar continuidade à discussão da ordenação por partição (QuickSort)

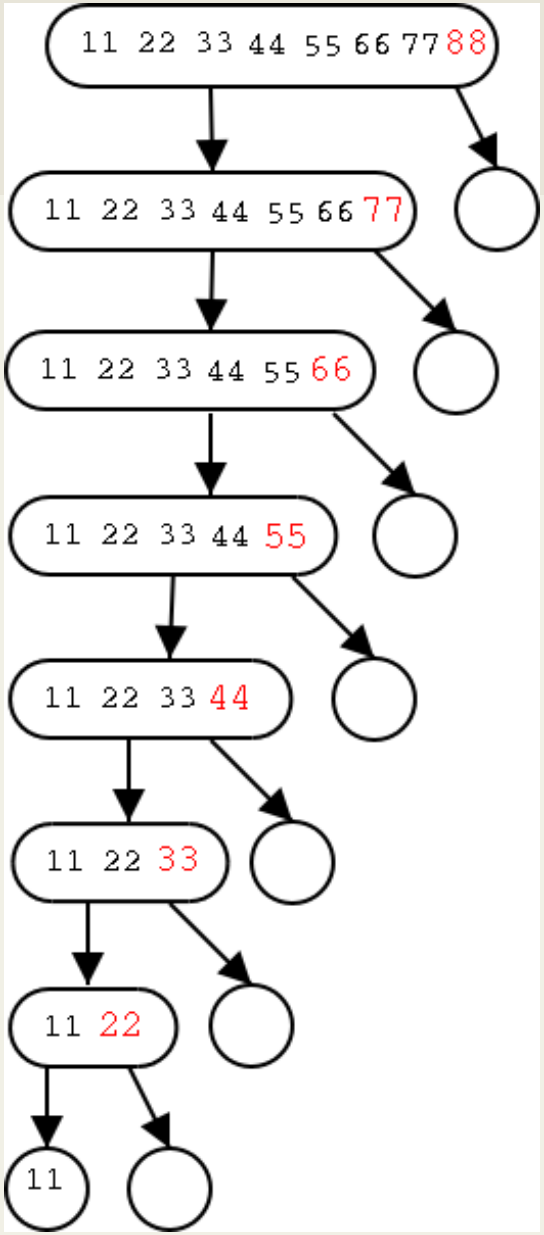
Qual era a complexidade da ordenação por partição no pior caso?

Qual era o fator crítico da ordenação por partição?

ESTRATÉGIAS PARA
ESCOLHA DO PIVÔ

Escolha do pivô

- Fator crítico para eficiência do algoritmo de ordenação por partição
  - Pior caso ocorre quando partição é “desbalanceada”



```
Particionar( v[n], esquerda, direita ) :
   pivo = v[direita]
   i = esquerda, j = direita-1
   WHILE j ≥ i :
        WHILE v[i] < pivo && j ≥ i :
               i=i+1
        END_WHILE
        WHILE v[j] > pivo && j ≥ i :
               j = j-1
        END_WHILE
        IF j ≥ i :
               Swap v[i], v[j]
        END_IF
   END_WHILE
   Swap v[i], v[direita]
   Return i
END
```

Escolha do pivô

- Estratégia usada: pivô é o último elemento

- Outra opção: escolher aleatoriamente o pivô

```
Particionar( v[n], esquerda, direita ) :
   i_pivo = Randomico( 0, n-1 )
   Swap v[i_pivo], v[direita]
   pivo = v[direita]
   i = esquerda, j = direita-1
   WHILE j ≥ i :
        WHILE v[i] < pivo && j ≥ i :
               i=i+1
        END_WHILE
        WHILE v[j] > pivo && j ≥ i :
               j = j-1
        END_WHILE
        IF j ≥ i :
               Swap v[i], v[j]
        END_IF
   END_WHILE
   Swap v[i], v[direita]
   Return i
END
```

Escolha do pivô

- Idealmente, o pivô deveria ser a mediana do vetor
  - Mediana := valor numérico que divide uma amostra de dados ao meio

```
Particionar( v[n], esquerda, direita ) :
   i_mediana = AcharMediana( v )
   Swap v[i_mediana], v[direita]
   pivo = v[direita]
   i = esquerda, j = direita-1
   WHILE j ≥ i :
        WHILE v[i] < pivo && j ≥ i :
               i=i+1
        END_WHILE
        WHILE v[j] > pivo && j ≥ i :
               j = j-1
        END_WHILE
        IF j ≥ i :
               Swap v[i], v[j]
        END_IF
   END_WHILE
   Swap v[i], v[direita]
   Return i
END
```

Escolha do pivô

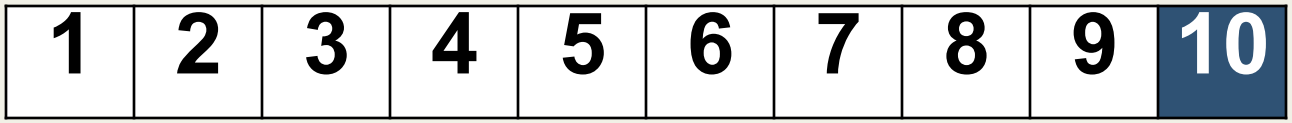
- Gerar números randômicos ou achar a mediana do vetor aumentam o custo de execução do algoritmo de partição

- Solução simples e eficiente:
  - Usar como pivô a mediana de três elementos do vetor: primeiro, último e meio

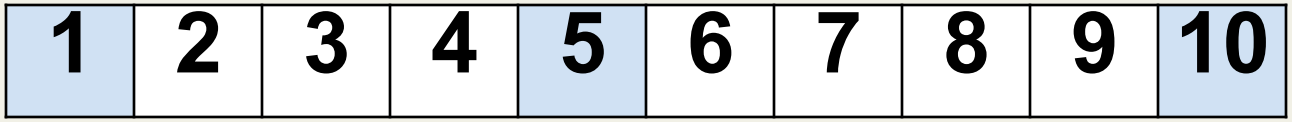
```
Particionar( v[1...n], esquerda, direita ) :
   meio = (esquerda + direita) /2
   i_mediana = AcharMediana( v, esquerda, direita, meio )
   Swap v[i_mediana], v[direita]
   pivo = v[direita]
   i = esquerda, j = direita-1
   WHILE j ≥ i :
        WHILE v[i] < pivo && j ≥ i :
               i=i+1
        END_WHILE
        WHILE v[j] > pivo && j ≥ i :
               j = j-1
        END_WHILE
        IF j ≥ i :
               Swap v[i], v[j]
        END_IF
   END_WHILE
   Swap v[i], v[direita]
   Return i
END
```

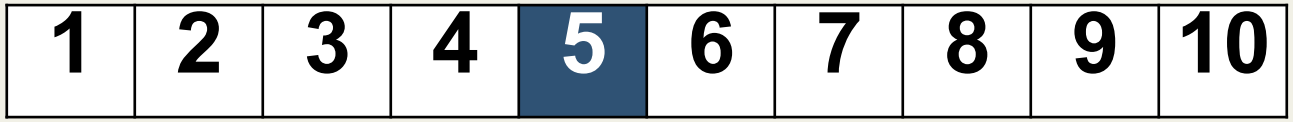
Mediana de 3

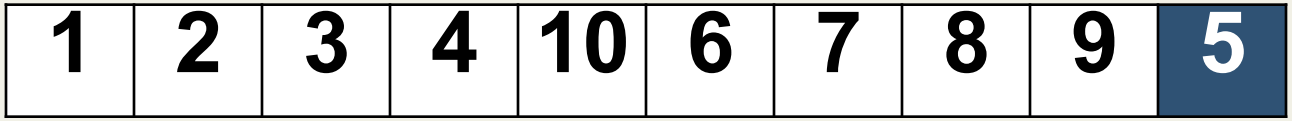
Estratégia simples: último elemento



Estratégia mediana de 3: esquerda, meio ou direita







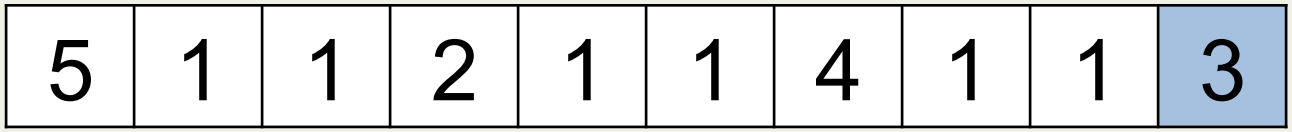
LIDANDO COM ELEMENTOS REPETIDOS

Elementos repetidos

- Ordenar vetores com elementos repetidos ocorre com bastante frequência
  - Ex.: Ordenar por ano de nascimento

- Algoritmos visto resolve ordenação com elementos repetidos, mas pode ser melhorado

Elementos repetidos



Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1								

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1								
1									

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1		1	1	1	2			
1									

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1		1	1	1	2			
1									

Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1		1	1	1	2			
1			1	1	1	2			

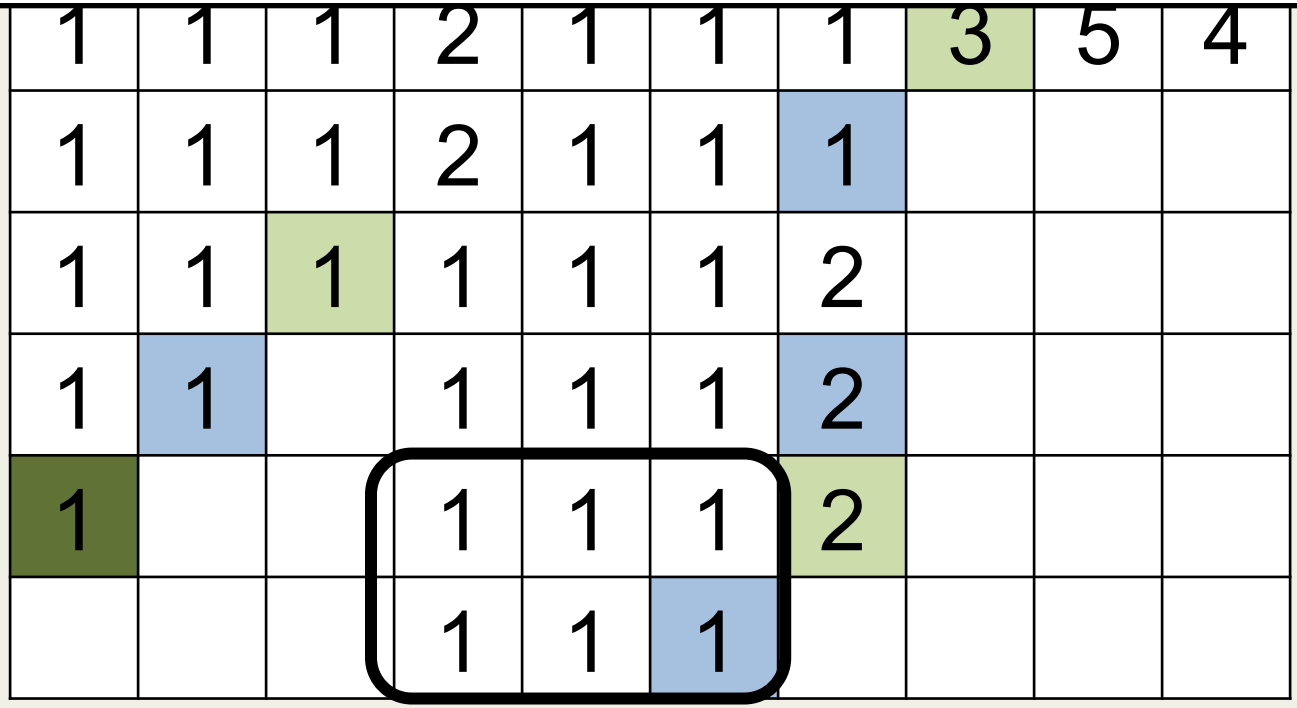
Elementos repetidos

5	1	1	2	1	1	4	1	1	3
1	1	1	2	1	1	1	3	5	4
1	1	1	2	1	1	1			
1	1	1	1	1	1	2			
1	1		1	1	1	2			
1			1	1	1	2			
			1	1	1				

...

Elementos repetidos

São feitas diversas chamadas para ordenar elementos repetidos



Como evitar fazer **chamada recursiva para** elementos repetidos?

Elementos repetidos

Primeira solução

Antes:

Durante:

Depois:

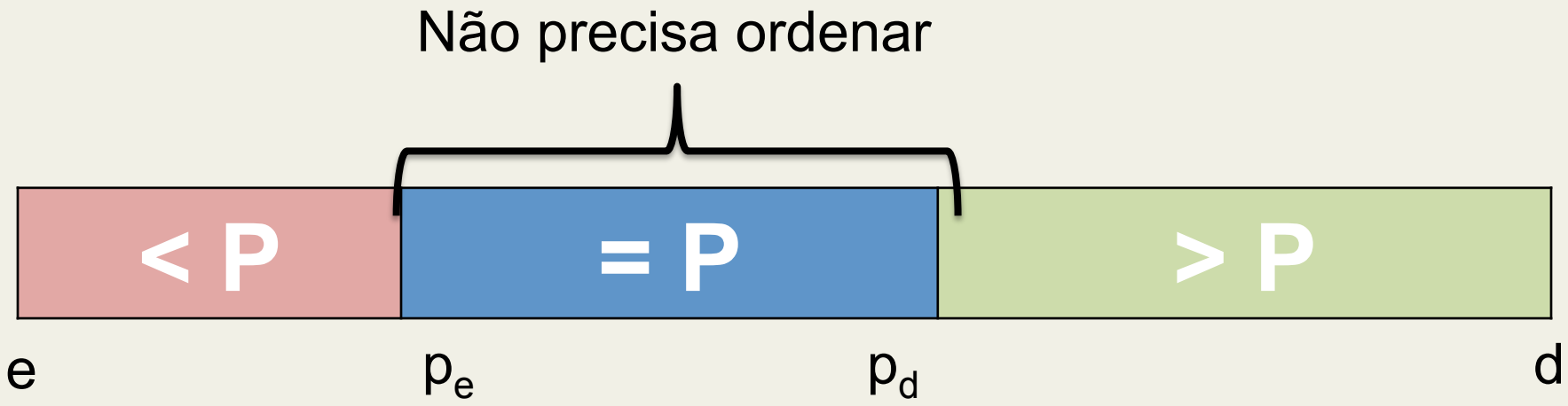
Melhoria para Elementos Repetidos

Antes:

Durante:

Depois:

Elementos repetidos

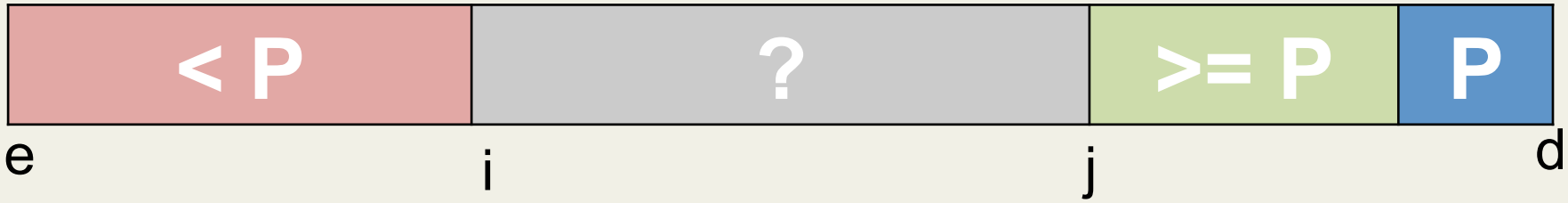



```
OrdenarParticao( v, esquerda, p_e-1 )
OrdenarParticao( v, p_d+1, direita )
```

Durante a Execução

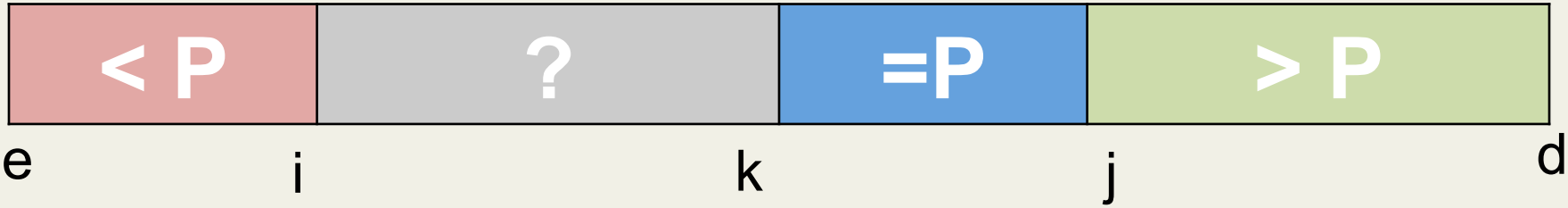
- Primeira solução

Durante:

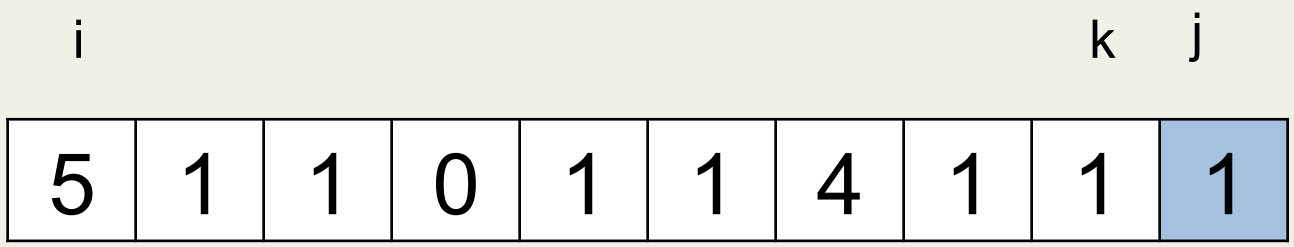


- Com melhoria para elementos repetidos

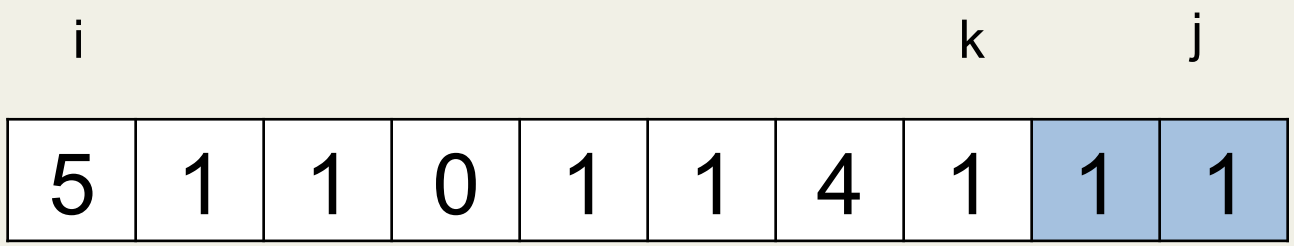
Durante:



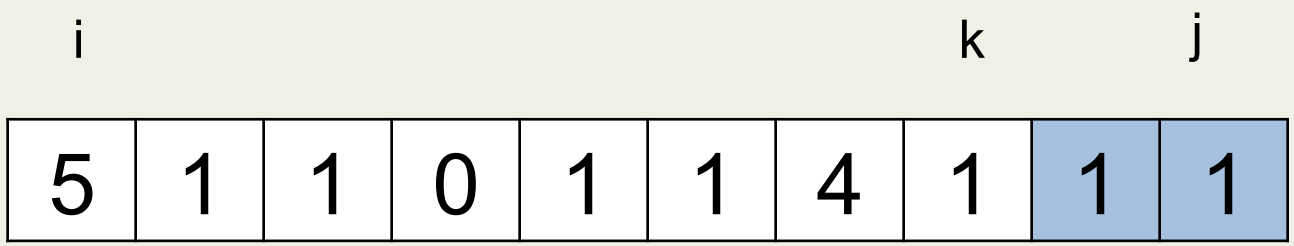
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo :
            k--
      FIM_SE
FIM_ENQUANTO
```



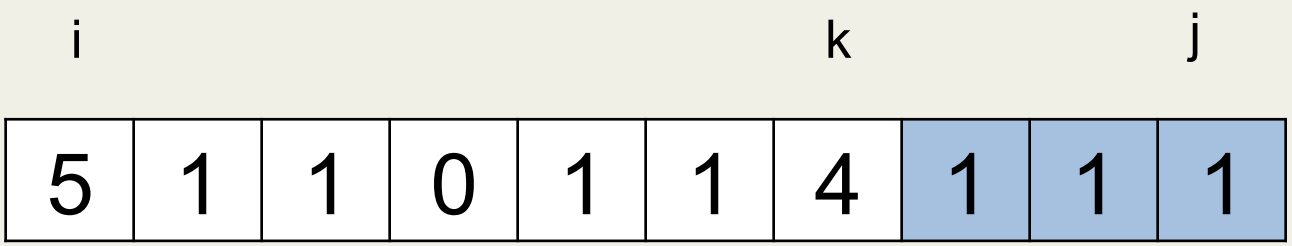
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo :
            k--
      FIM_SE
FIM_ENQUANTO
```



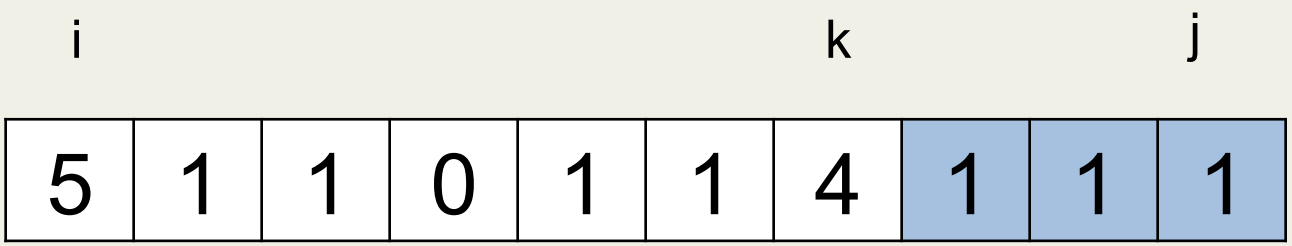
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo :
            k--
      FIM_SE
FIM_ENQUANTO
```



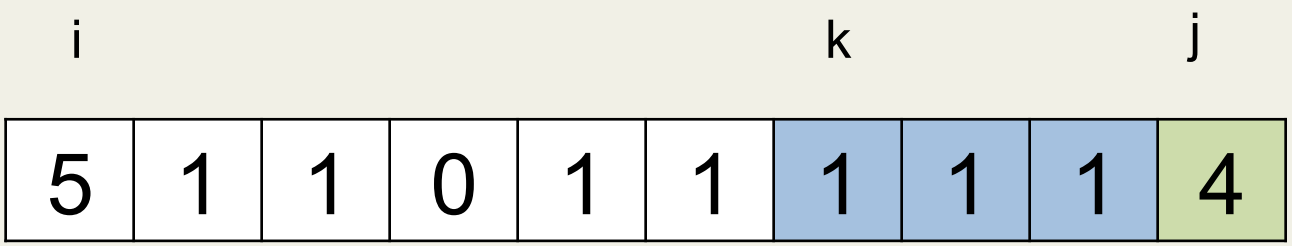
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo :
            k--
      FIM_SE
FIM_ENQUANTO
```



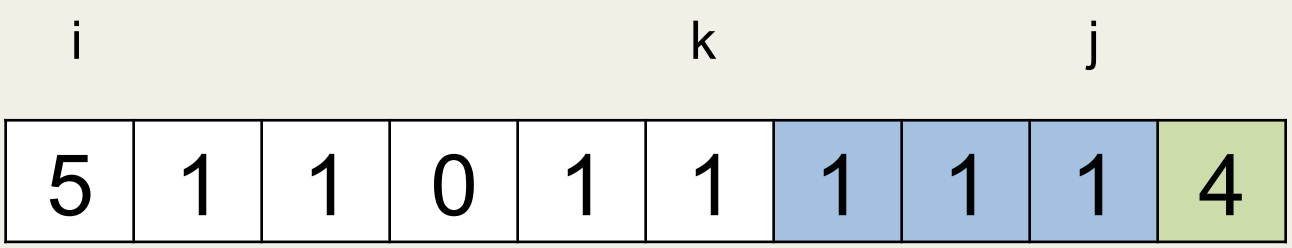
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



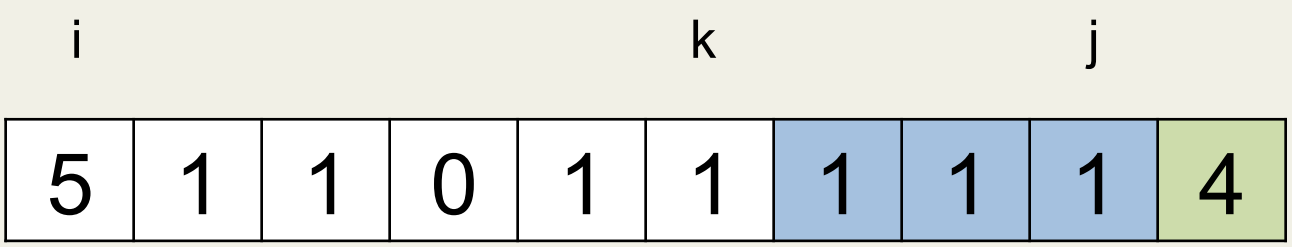
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



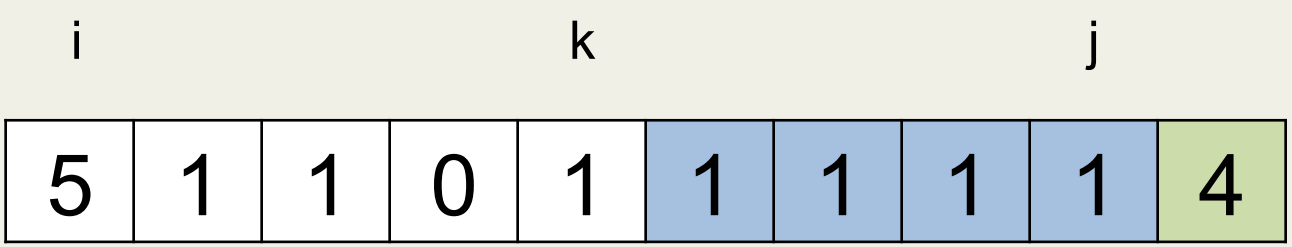
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



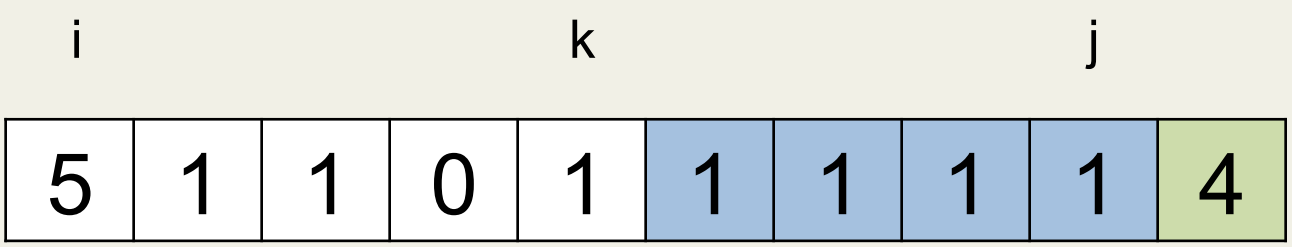
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



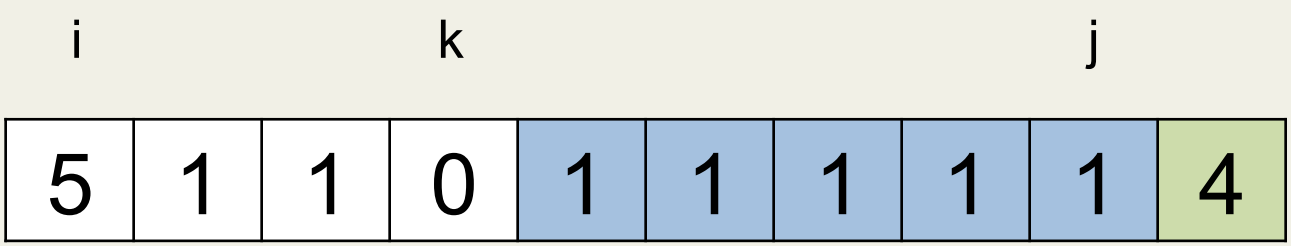
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



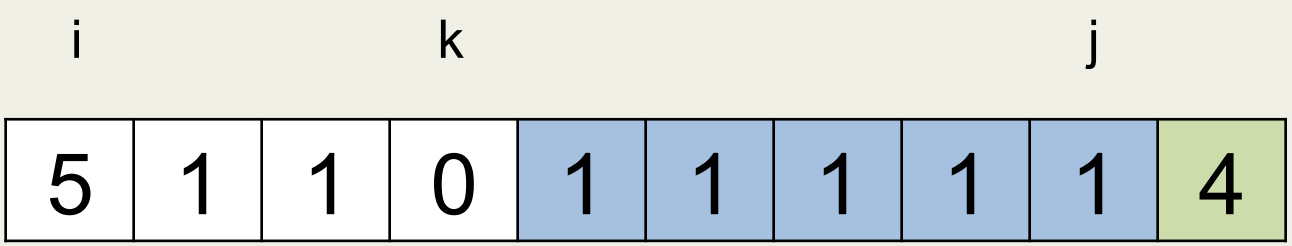
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



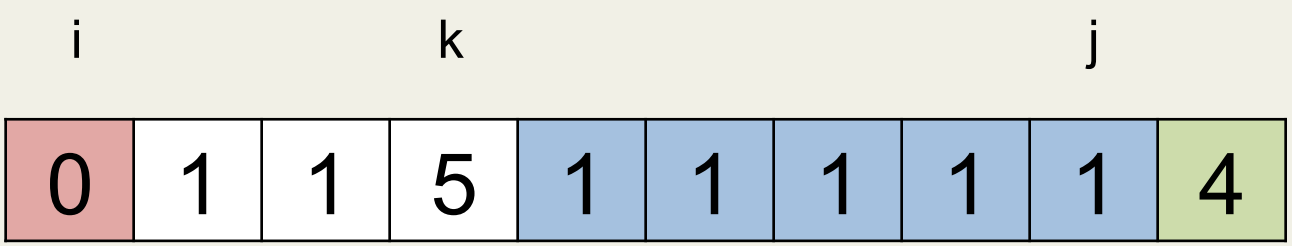
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



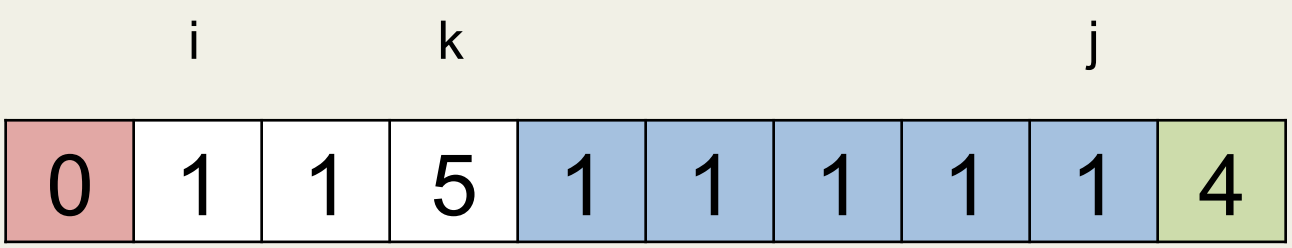
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



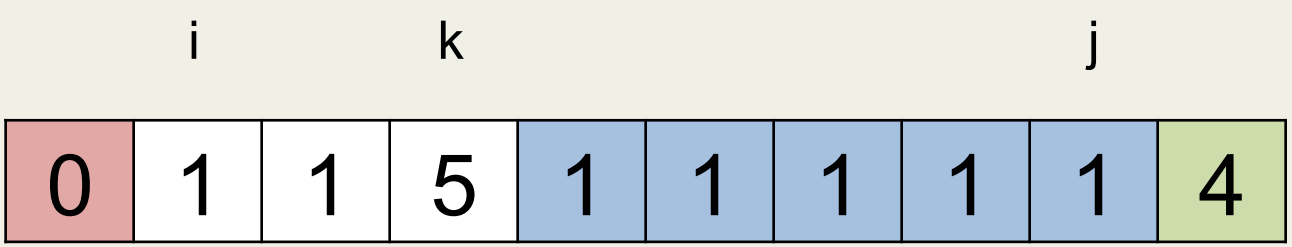
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



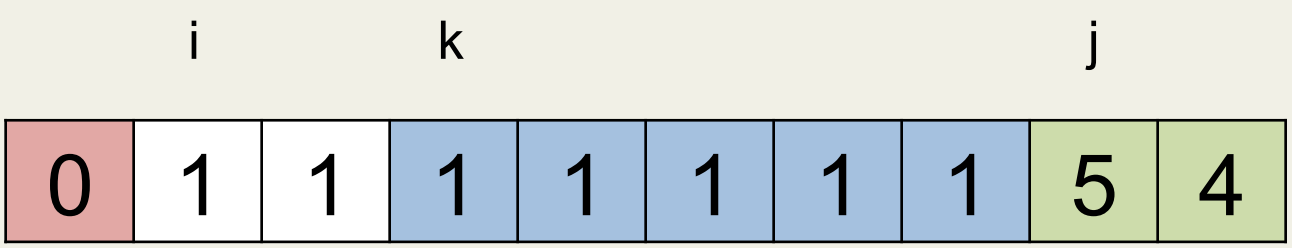
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



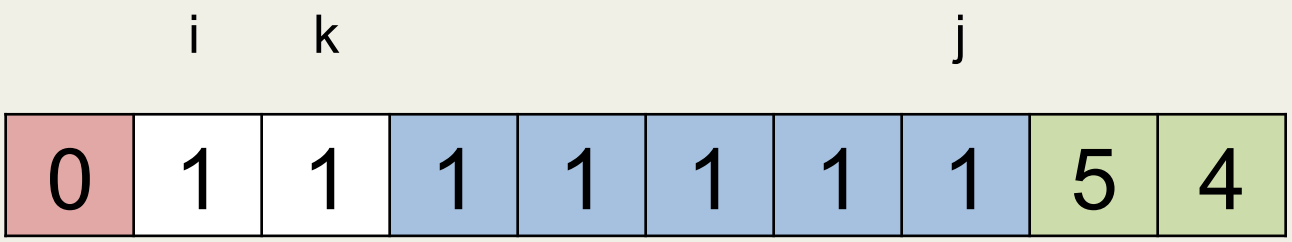
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



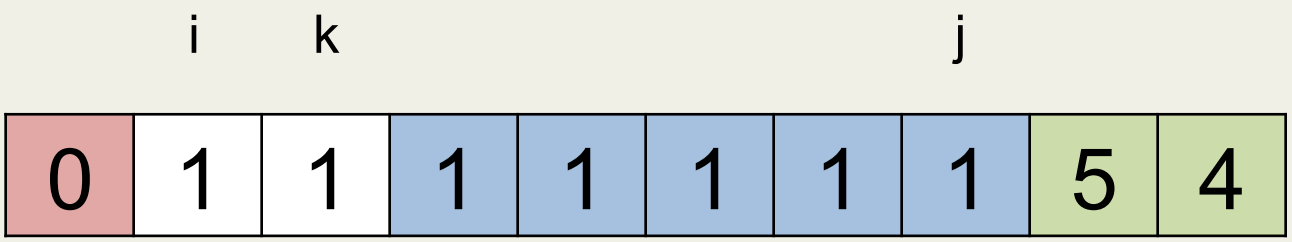
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



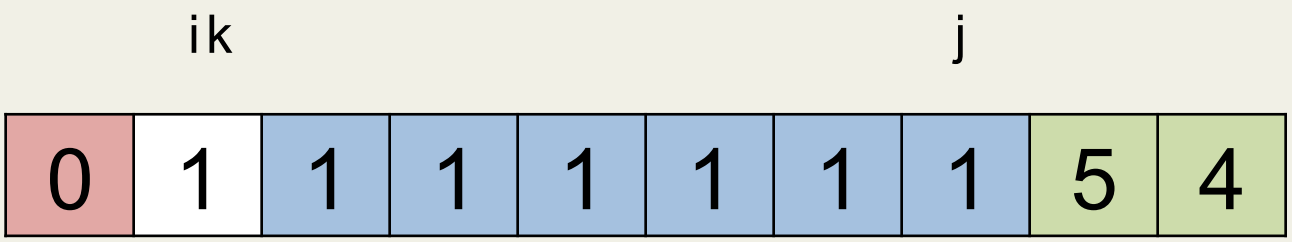
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



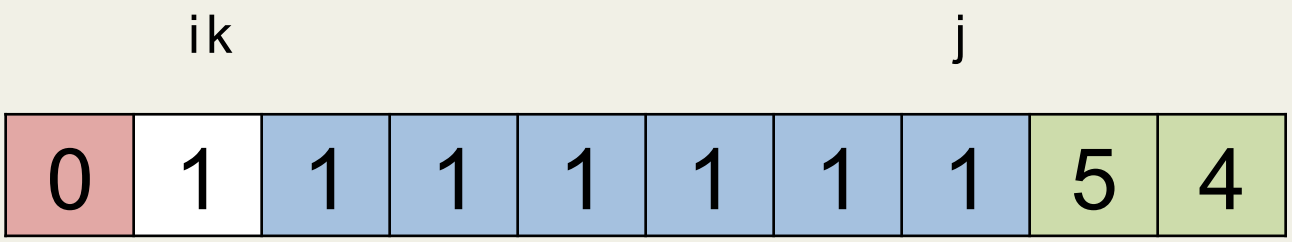
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



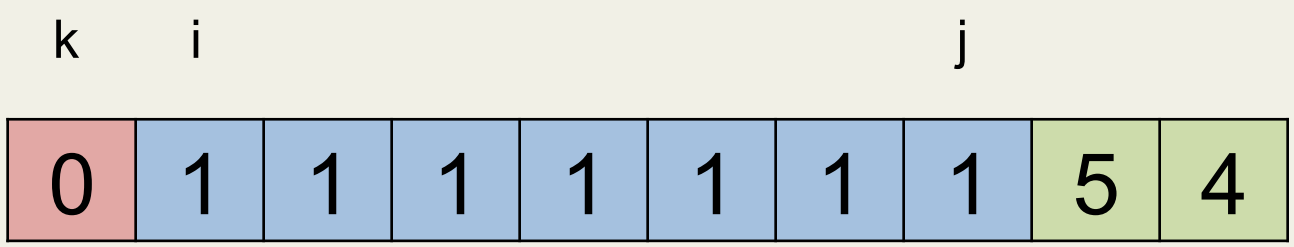
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



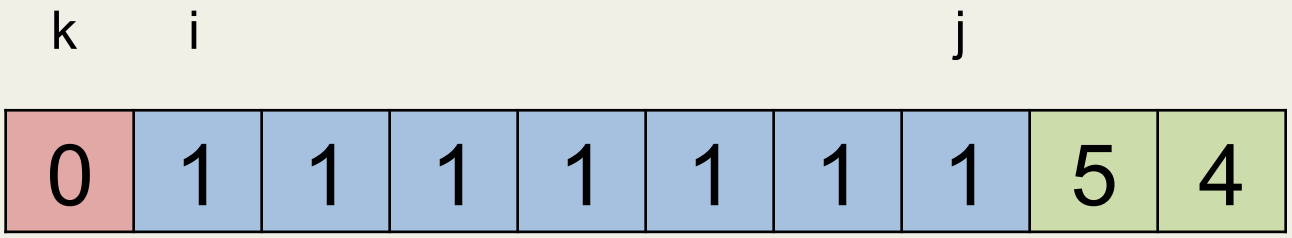
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO ?
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo
            k--
      FIM_SE
FIM_ENQUANTO
```



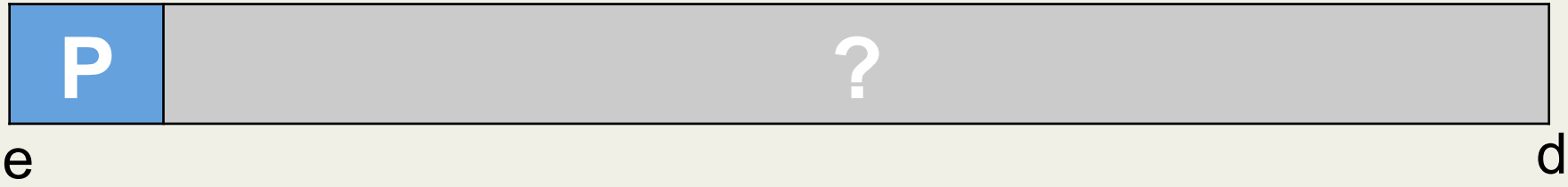
```
pivo = v[direita], k = direita-1, i =esquerda, j = direita

ENQUANTO k >= i
      SE v[k]   < pivo ENTAO
            TROQUE v[i], v[k],
            i++
      SENÃO SE v[k]   > pivo ENTAO
            TROQUE v[j], v[k],
            j--, k--
      SENÃO \\ v[k] == pivo:
            k--
      FIM_SE
FIM_ENQUANTO
```



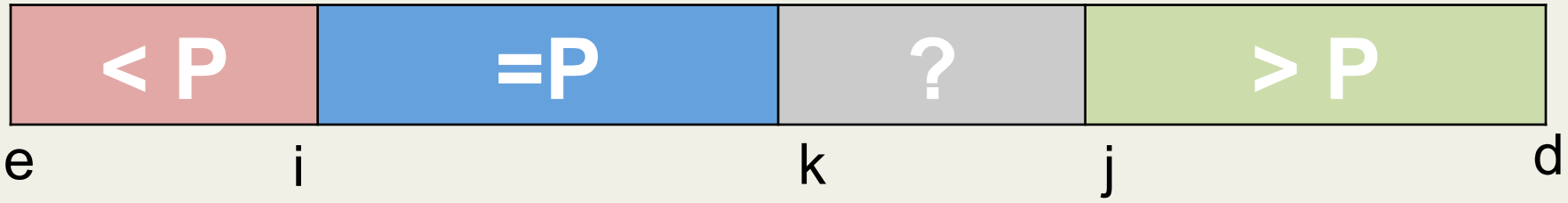
Elementos repetidos

- E se a configuração do vetor antes e durante a execução do laço tiver que ser como abaixo, como ficaria o algoritmo?

Durante:



Durante:



Prática

- Implementem a versão do algoritmo de partição que separa os elementos repetidos
  - Implemente a versão que define o pivô como sendo o elemento a esquerda: pivo = v[esquerda]
- Use a nova versão do algoritmo de partição para implementar o quick-sort
- Comparem o desempenho do quick-sort:
  - Versão que leva em conta elementos repetidos X Versão que não leva em conta elementos repetidos

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I.

Algoritmos de ordenação VI

Prof. Eiji Adachi M. Barbosa