ESTRUTURA DE DADOS

ORDENAÇÃO POR TROCA E PARTIÇÃO

ORDENAÇÃO - QUICKSORT

O algoritmo Quicksort adota a estratégia de divisão e conquista. Os passos são:

- Escolha um elemento da lista, denominado pivô;
- Rearranje a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele.

Ao fim do processo o pivô estará em sua posição final e haverá duas sublistas não ordenadas. Essa operação é denominada partição;

ORDENAÇÃO - QUICKSORT

- Recursivamente ordene a sublista dos elementos menores e a sublista dos elementos maiores;
- A base da recursão são as listas de tamanho zero ou um, que estão sempre ordenadas. O processo é finito, pois a cada iteração pelo menos um elemento é posto em sua posição final e não será mais manipulado na iteração seguinte.

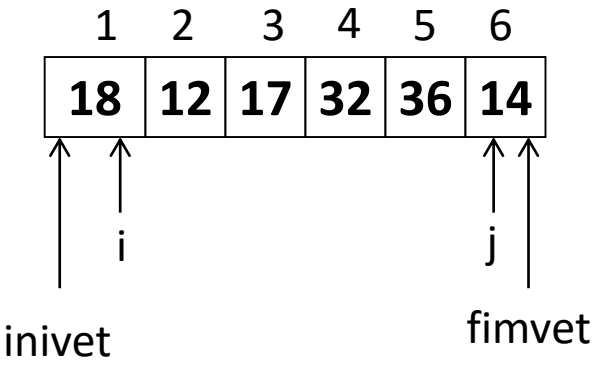
ORDENAÇÃO - QUICKSORT

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)
  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

```
Algoritmo
  LERVETOR(A)
  QUICKSORT(A, 1, n)
  ESCREVERVETOR(A)
fim_algoritmo
```

ORDENAÇÃO - QUICK SORT

Vetor desorganizado



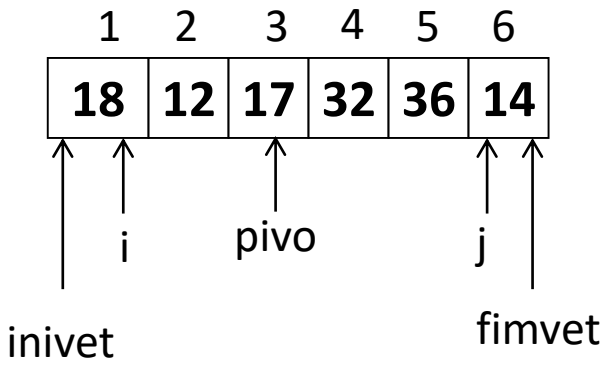
Pivo = a[(inivet+fimvet)/2]

Pivo = a[(1+6)/2]=a[3]=17

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT



Enquanto i <= j faça

Enquanto 1 <= 6 faça

- Procedimento QUICKSORT(var A, INIVET, FIMVET)

```
I <- INIVET
J <- FIMVET
PIVO <- A[(INIVET + FIMVET) div 2]
enquanto I <= J faça
  enquanto A[I] < PIVO faça
    I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
    J <- J - 1
  fim_enquanto
  se (I <= j) então
    AUX  <- A[I]
    A[I] <- A[j]
    A[j] <- AUX
    I <- I + 1
    J <- J - 1
  fim_se
fim_enquanto
se J > INIVET então
  QUICKSORT(A, INIVET, J)
fim_se
se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
fim_se
fim_procedimento
```

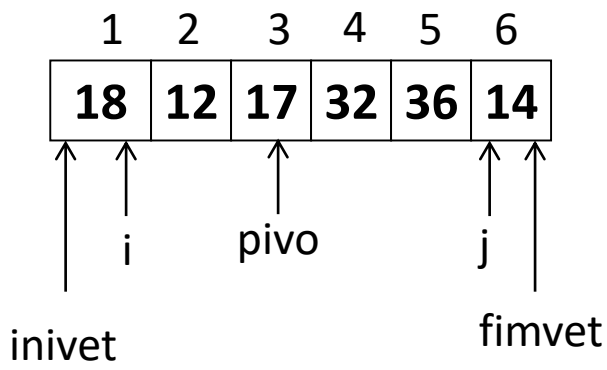
ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

A[1] < pivo
18 < 17

A condição e falsa o ponteiro(i) para e a busca começa do lado direito




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

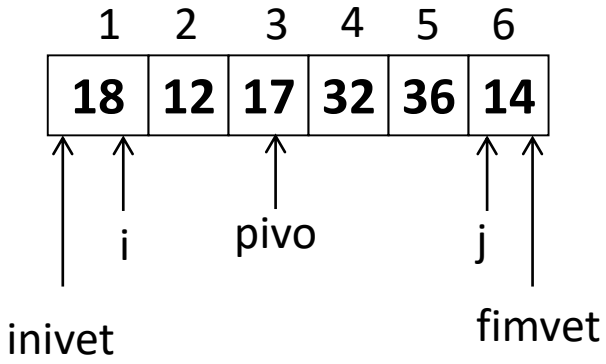
ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[6] > pivo
14 > 17

A condição e falsa, o ponteiro(j) para
E sai do enquanto passa a fazer o **se**




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Se 1 <= 6 então

  AUX <- A[1]

  A[1] <- A[6] = 14

  A[j] <- AUX = 18

  I <- I + 1 = 2

  J <- J – 1 = 5

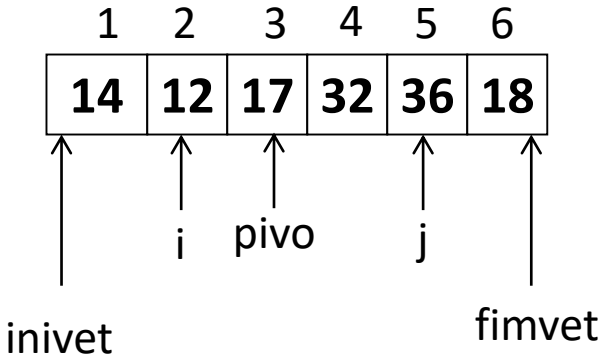
- fim_se

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto 2 <= 5 faça




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

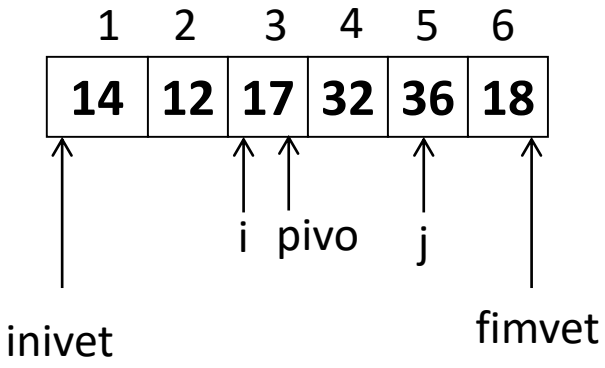
ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

A[2] < pivo
12 < 17

A condição é verdadeira e incrementa (i), i = 3




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

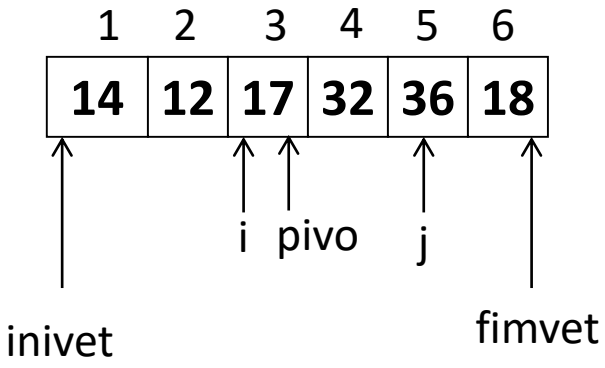
ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

A[3] < pivo
17 < 17

A condição é falsa, o ponteiro (i) para e a busca começa do lado direito




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

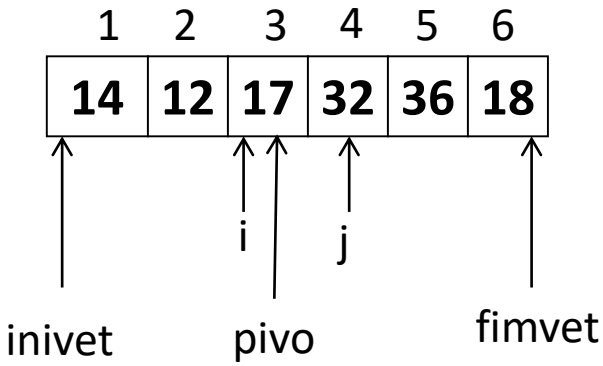
ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[5] > pivo
36 > 17

A condição é verdadeira e decrementa
J, j = 5 – 1 = 4




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

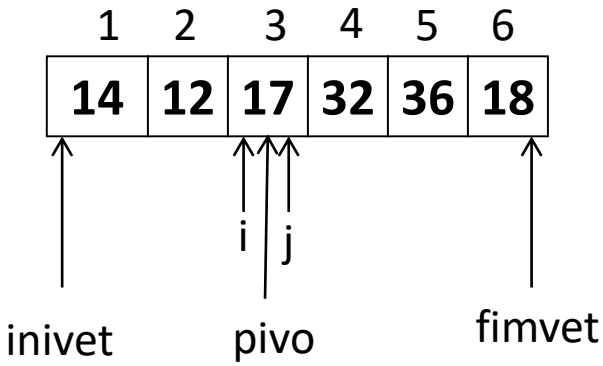
ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[4] > pivo
32 > 17

A condição é verdadeira e decrementa
j, j = 4 – 1 = 3




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[3] > pivo
17 > 17

A condição é falsa, o ponteiro(j) para
E sai do enquanto passa a fazer o **se**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Se 3 <= 3 então

  AUX <- A[3]= 17

  A[3] <- A[3] = 17

  A[j] <- AUX = 17

  I <- I + 1 = 4

  J <- J – 1 = 2

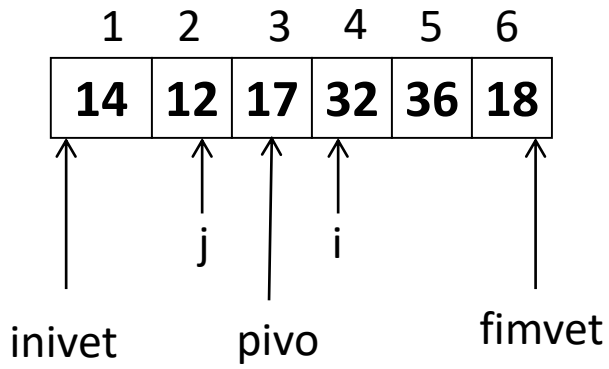
- fim_se

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- i = 4
- j = 2




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto i <= j faça
- Enquanto 4<= 2 faça

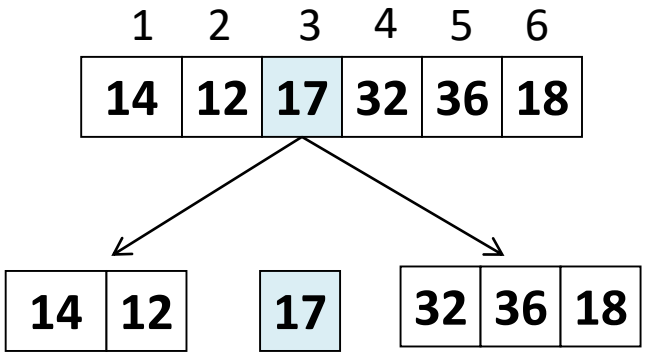
- A condição é falsa e sai do enquanto e passa executar as chamadas recursivas: primeiro executa Se J > INIVET então QUICKSORT(A, INIVET, J), e depois Se I < FIMVET então QUICKSORT(A, I, FIMVET)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- À esquerda os números menores que o pivô e à direita os números maiores que o pivô




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se J > INIVET então

  QUICKSORT(A, INIVET, J)

  fim_se
- J = 2, INIVET = 1
- se 2 > 1 então

  QUICKSORT(A, 1, 2)

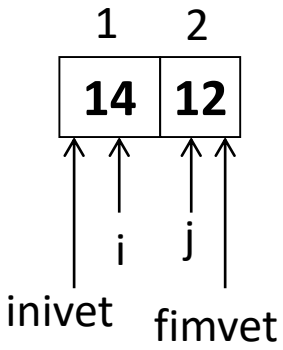
  fim_se

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- INIVET = 1, FIMVET= 2, I = 1, J =2



Pivo = a[(inivet+fimvet)/2]

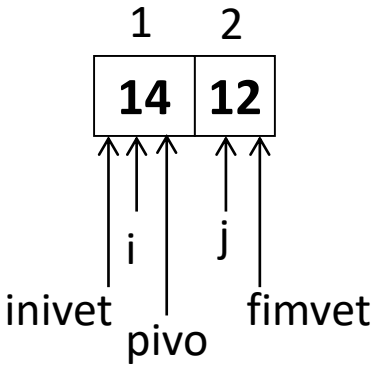
Pivo = a[(1+2)/2]=a[1]=14

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto I <= J faça
- Enquanto 1 <=2 faça




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

  A[1] < pivo
  14 < 14

  A condição e falsa o ponteiro(i) para e a busca começa do lado direito

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[2] > pivo
12 > 14

A condição e falsa, o ponteiro(j) para
E sai do enquanto passa a fazer o **se**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Se 1 <= 2 então

  AUX <- A[1] =14

  A[1] <- A[2] = 12

  A[2] <- AUX = 14

  I <- I + 1 = 2

  J <- J – 1 = 1

- fim_se

1	2
12	**14**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

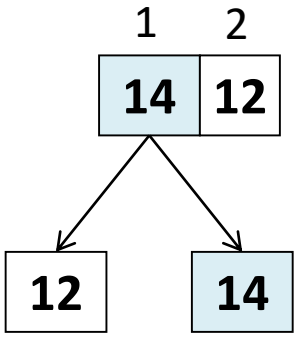
- Enquanto i <= j faça
- Enquanto 2 <= 1 faça

- A condição é falsa e sai do enquanto e passa executar as chamadas recursivas: primeiro executa Se J > INIVET então QUICKSORT(A, INIVET, J), e depois Se I < FIMVET então QUICKSORT(A, I, FIMVET)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se J > INIVET então

  QUICKSORT(A, INIVET, J)

  fim_se
- J = 1, INIVET = 1
- se 1 > 1 então

  QUICKSORT(A, 1, 1)

  fim_se

A condição é falsa e não executa o então.

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se I < FIMVET então

  QUICKSORT(A, I, FIMVET)
- fim_se

I = 2, FIMVET = 2
se 2 > 2 então
QUICKSORT(A, 1, 1)
fim_se

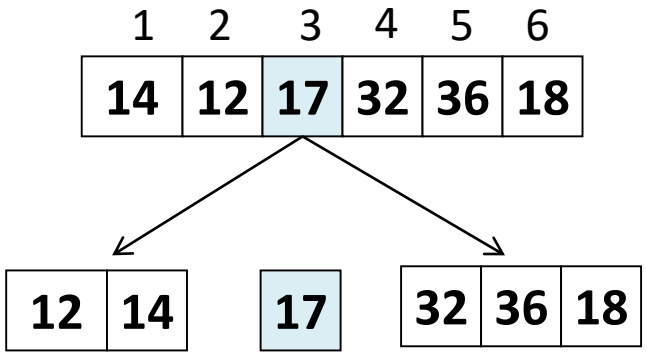
- A condição é falsa e não executa o então.

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Agora que terminou de ordenar o lado esquerdo do pivo = 17 e passa a ordenar o lado direito




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- I = 4
- FIMVET = 6
- Se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
- fim_se

- Se 4 < 6 então
  QUICKSORT(A, 4, 6)
- fim_se

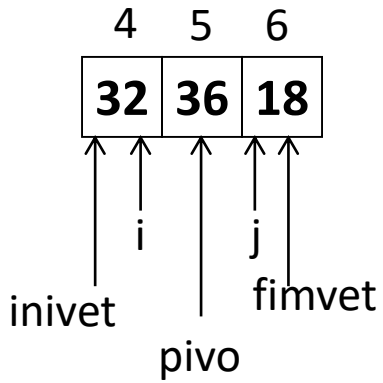
- A condição é verdadeira então chama QUICKSORT(A, 4, 6)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- INIVET = 4, FIMVET= 6, I = 4, J =6



Pivo = a[(inivet+fimvet)/2]

Pivo = a[(4+6)/2]=a[5]=36

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto I <= J faça
- Enquanto 4 <=6 faça

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

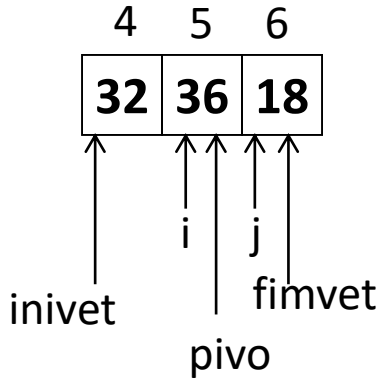
ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

A[4] < pivo
32< 36

A condição é verdadeira e incrementa (i)
I = I+ 1 = 5




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

  A[5] < pivo
  36< 36

  A condição é falsa , e para o ponteiro (i)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[6] > pivo
18 > 36

A condição e falsa, o ponteiro(j) para
E sai do enquanto passa a fazer o **se**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se (5 <= 6) então

  AUX <- A[5] = 36

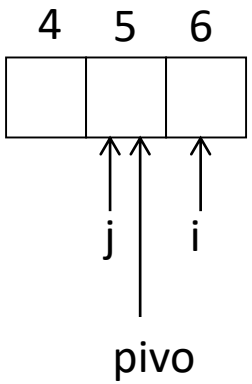
  A[5] <- A[6] = 18

  A[6 <- AUX = 36

  I <- I + 1 = 6

  J <- J – 1 = 5

  fim_se

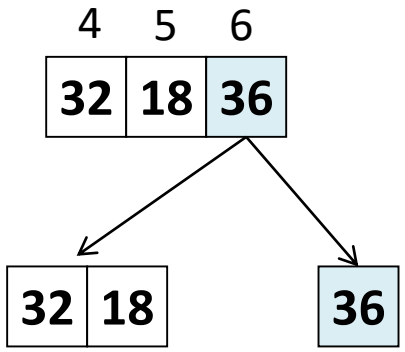



```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- À esquerda os números menores que o pivô.




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- I = 6 e j = 5
- Enquanto I <= j faça
- Enquanto 6 <= 5 faça
- A condição é falsa e sai do enquanto e passa executar as chamadas recursivas: primeiro executa Se J > INIVET então QUICKSORT(A, INIVET, J), e depois Se I < FIMVET então QUICKSORT(A, I, FIMVET)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se J > INIVET então

  QUICKSORT(A, INIVET, J)

  fim_se
- J = 5, INIVET = 4
- Se 5 > 4 então

  QUICKSORT(A, 4, 5)

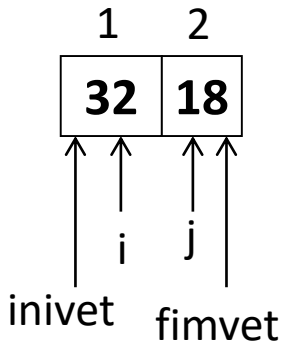
  fim_se

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- INIVET = 4, FIMVET= 5, I = 4, J = 5



Pivo = a[(inivet+fimvet)/2]

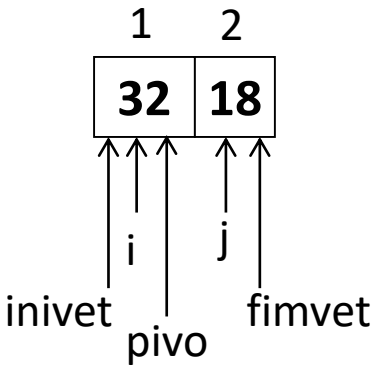
Pivo = a[(4+5)/2]=a[4]=32

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto i <= j faça
- Enquanto 4 < = 5 faça




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[i] < pivo faça

  i ←i+1

A[4] < pivo
32 < 32

A condição e falsa o ponteiro(i) para e a busca começa do lado direito

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto a[j] > pivo faça

  j ←j-1

A[5] > pivo
18 > 32

A condição e falsa, o ponteiro(j) para
E sai do enquanto passa a fazer o **se**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Se 4 <= 5 então

  AUX <- A[4] =32

  A[4] <- A[5] = 18

  A[5] <- AUX =32

  I <- I + 1 = 5

  J <- J – 1 = 4

- fim_se

4	5
18	**32**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- Enquanto i <= j faça
- Enquanto 5 <= 4 faça

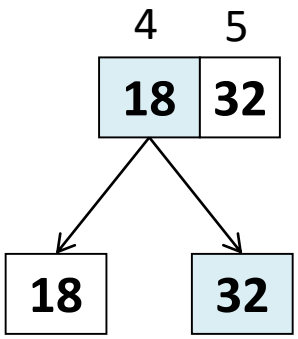
- A condição é falsa e sai do enquanto e passa executar as chamadas recursivas: primeiro executa Se J > INIVET então QUICKSORT(A, INIVET, J), e depois Se I < FIMVET então QUICKSORT(A, I, FIMVET)

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- J = 4 e i = 5




```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se J > INIVET então

  QUICKSORT(A, INIVET, J)

  fim_se
- J = 4, INIVET = 4
- Se 4 > 4 então

  QUICKSORT(A, 4, 4)

  fim_se

A condição é falsa e não executa o então.

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- se I < FIMVET então

  QUICKSORT(A, I, FIMVET)
- fim_se

I = 5, FIMVET = 5
Se 5 > 5 então
QUICKSORT(A, 5, 5)
fim_se

- A condição é falsa e não executa o então.

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

 I <- INIVET
 J <- FIMVET
 PIVO <- A[(INIVET + FIMVET) div 2]
 enquanto I <= J faça
  enquanto A[I] < PIVO faça
   I <- I + 1
  fim_enquanto
  enquanto A[j] > PIVO faça
   J <- J - 1
  fim_enquanto
  se (I <= j) então
   AUX  <- A[I]
   A[I] <- A[j]
   A[j] <- AUX
   I <- I + 1
   J <- J - 1
  fim_se
 fim_enquanto
 se J > INIVET então
  QUICKSORT(A, INIVET, J)
 fim_se
 se I < FIMVET então
  QUICKSORT(A, I, FIMVET)
 fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- O vetor ordenado

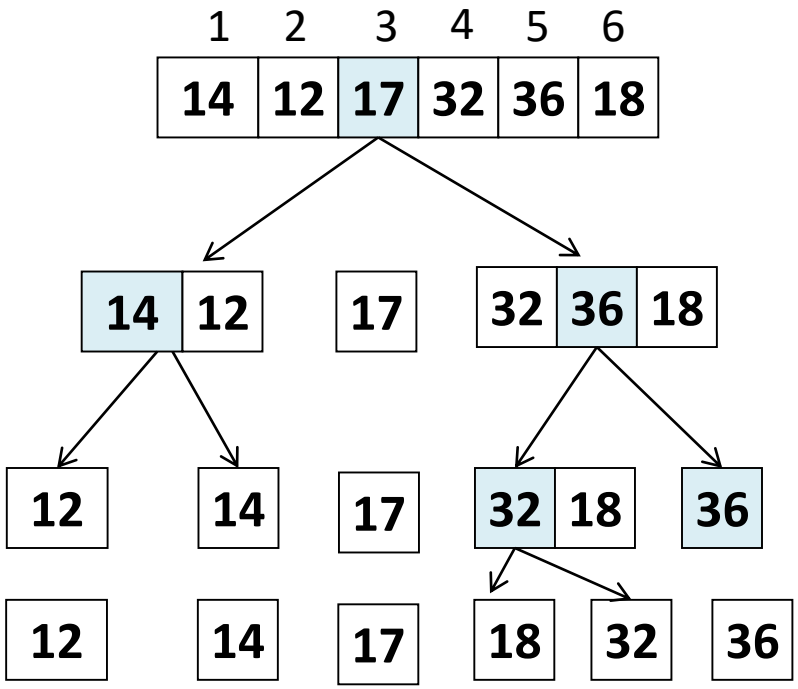
1	2	3	4	5	6
14	**12**	**17**	**18**	**32**	**36**

```
Procedimento QUICKSORT(var A, INIVET, FIMVET)

  I <- INIVET
  J <- FIMVET
  PIVO <- A[(INIVET + FIMVET) div 2]
  enquanto I <= J faça
    enquanto A[I] < PIVO faça
      I <- I + 1
    fim_enquanto
    enquanto A[j] > PIVO faça
      J <- J - 1
    fim_enquanto
    se (I <= j) então
      AUX  <- A[I]
      A[I] <- A[j]
      A[j] <- AUX
      I <- I + 1
      J <- J - 1
    fim_se
  fim_enquanto
  se J > INIVET então
    QUICKSORT(A, INIVET, J)
  fim_se
  se I < FIMVET então
    QUICKSORT(A, I, FIMVET)
  fim_se
fim_procedimento
```

ORDENAÇÃO - QUICKSORT

- O vetor ordenado



ORDENAÇÃO - QUICKSORT

- Comparações
  - Pior caso: O (n^2)
  - Melhor caso: O (nlogn)
  - Caso médio: O (nlogn)