Aula 2

- Teorema Mestre
- Divida - para - Conquistar
  - Procura Binária
  - Elemento Majoritário
  - Mediana
- Quicksort

Teorema Mestre (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a \\ O(n^d \log n) & \text{se } d = \log_b a \\ O(n^{\log_b a}) & \text{se } d < \log_b a \end{cases}$$

Teorema Mestre (Simplificado)

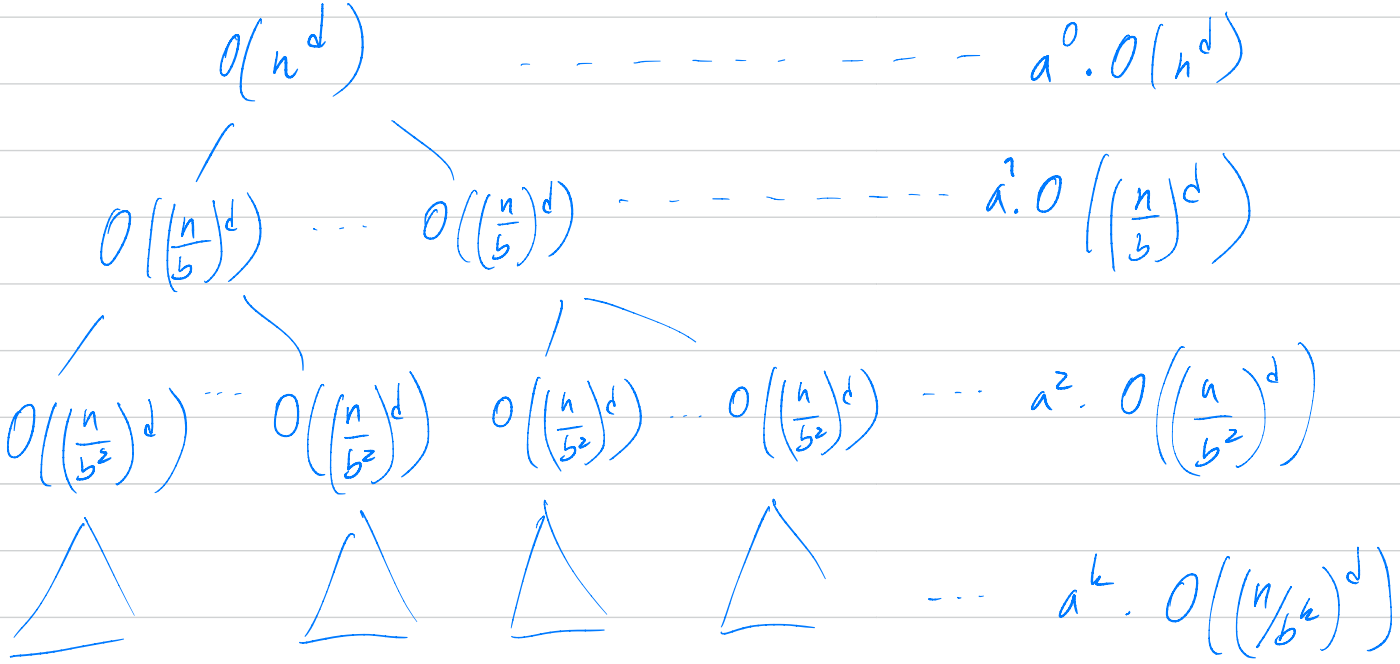
Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a \\ O(n^d \log n) & \text{se } d = \log_b a \\ O(n^{\log_b a}) & \text{se } d < \log_b a \end{cases}$$

Prova [Sketch]:

- Suponhamos q n é uma potência de b ($n = b^k$, para algum k)

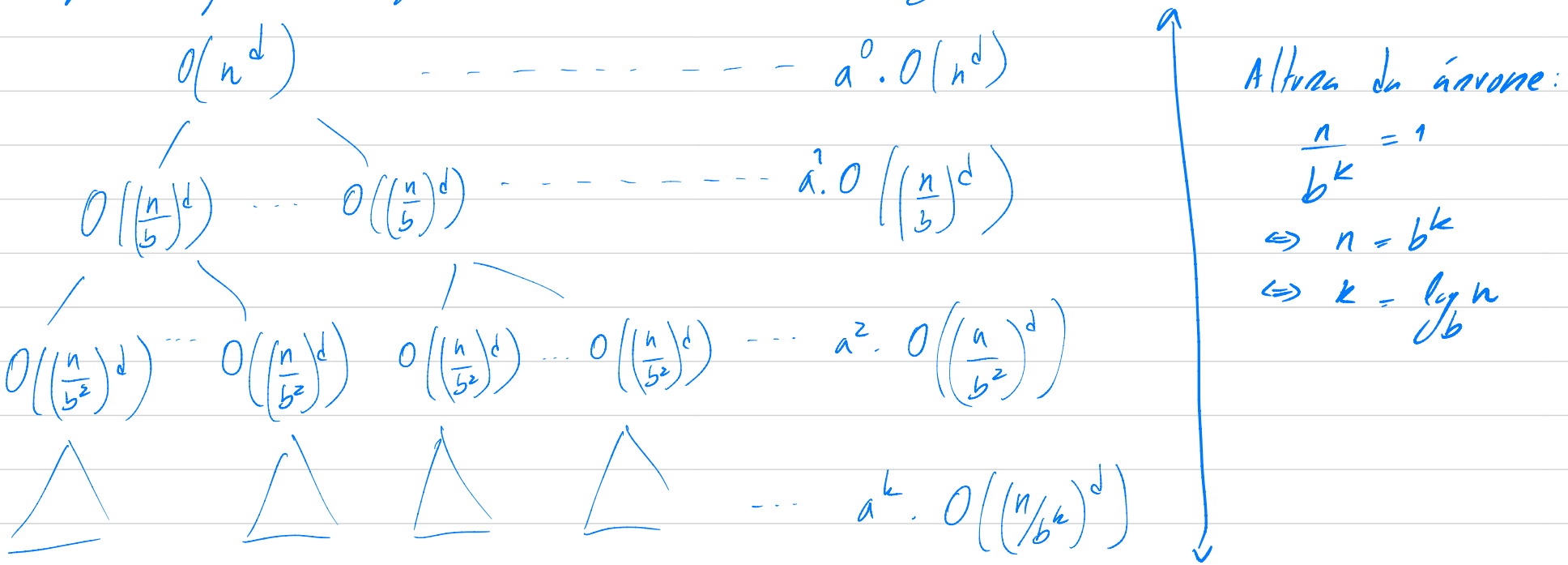


Prova [Sketch]:

- Suponhamos já n é uma potência de b ($n = b^k$, para algum k)

Prova [Sketch]:

- Suponhamos já n é uma potência de b ($n = b^k$, para algum k)



Altura da árvore:

$$\frac{n}{b^k} = 1$$
$$\Leftrightarrow n = b^k$$
$$\Leftrightarrow k = \log_b n$$

Custo da Árvore:

$$T(n) = \sum_{k=0}^{?} a^k \cdot O\left(\left(n/b^k\right)^d\right)$$

? → altura da árvore

Custo da Árvore:

$$T(n) = \sum_{k=0}^{\log_b n} a^k \cdot O\left(\left(n/b^k\right)^d\right)$$

Custo da Árvore:

$$T(n) = \sum_{k=0}^{\log_b n} a^k \cdot O\left(\left(n/b^k\right)^d\right)$$

$$= \sum_{k=0}^{\log_b n} O\left(a^k \cdot \left(\frac{n}{b^k}\right)^d\right)$$

$$= n^d \sum_{k=0}^{\log_b n} O\left(\left(\frac{a}{b^d}\right)^k\right)$$

$$= O\left(n^d \sum_{k=0}^{\log_b n} \left(\left(a/b^d\right)\right)^k\right)$$

3 casos

I - $(a/b^d) < 1$

II - $(a/b^d) = 1$

III - $(a/b^d) > 1$

Custo da Árvore:

$$T(n) = O\left(n^d \sum_{k=0}^{\log_b n} \left((a/b^d) \right)^k \right)$$

(III) $a/b^d > 1$

3 casos

I - $(a/b^d) < 1$

II - $(a/b^d) = 1$

III - $(a/b^d) > 1$

(I) $a/b^d < 1$

(II) $a/b^d = 1$

Custo da Árvore:

$$T(n) = O\left(n^d \sum_{k=0}^{\log_b n} \left((a/b^d) \right)^k \right)$$

3 casos

I - $(a/b^d) < 1$

II - $(a/b^d) = 1$

III - $(a/b^d) > 1$

(III) $a/b^d > 1 \rightsquigarrow$ o custo da série é dominado pelo custo do último termo.

$$\left(a/b^d\right)^{\log_b n} = a^{\log_b n} \cdot \frac{1}{\left(b^{\log_b n}\right)^d}$$

$$= \frac{a^{\log_b n}}{n^d}$$

$$T(n) = O\left(n^d \cdot \frac{a^{\log_b n}}{n^d} \right)$$

$$= O\left(a^{\log_b n} \right)$$

$$= O\left(n^{\log_b a} \right)$$

(I) $a/b^d < 1 \rightsquigarrow$ A série converge!

$$T(n) = O(n^d)$$

(II) $a/b^d = 1$

$$T(n) = O(n^d \cdot \log_b n)$$

Notas

- $a^{\log_b n} = a^{\log_a n / \log_a b}$

$= \left(a^{\log_a n}\right)^{1/\log_a b}$

$= n^{1/\log_a b}$

$= n^{\log_b a}$

$\log_a b = \log_b b / \log_b a$

$= 1 / \log_b a$

Teorema Mestre (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

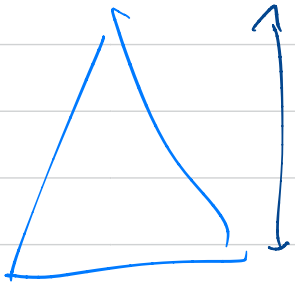
então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a \quad \text{(I)} \\ O(n^d \log n) & \text{se } d = \log_b a \quad \text{(II)} \\ O(n^{\log_b a}) & \text{se } d < \log_b a \quad \text{(III)} \end{cases}$$

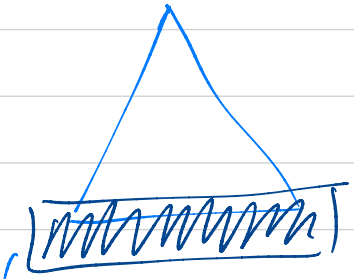
(I)

Custo da raiz domina o custo do problema

(II)



(III)



Custo é dominado pelo nº de folhas

Teorema Mestrae (Simplificado)

Se $T(n) = a \cdot T(\lceil n/b \rceil) + O(n^d)$ p/ constantes $a > 0$, $b > 1$ e $d \geq 0$

então:

$$T(n) = \begin{cases} O(n^d) & \text{se } d > \log_b a \quad \text{(I)} \\ O(n^d \log n) & \text{se } d = \log_b a \quad \text{(II)} \\ O(n^{\log_b a}) & \text{se } d < \log_b a \quad \text{(III)} \end{cases}$$

Merge Sort

$$T(n) = 2T(n/2) + O(n)$$

$a: 2$
$b: 2$
$d: 2$

$\log_b a = \log_2 2 = 1 = d$

Caso II $\rightarrow$ $T \in O(n \log n)$

Teorema Mestre Generalizado

Se $T(n) = a\,T(n/b) + f(n)$ p/ constantes $a \geq 1$ e $b > 1$
então:

Condição de Regularidade

(I) Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ p/ algum $\varepsilon > 0$, e se $a\,f(n/b) \leq c \cdot f(n)$ p/ algum $c < 1$ e n suficientemente grande,
então: $T(n) = \Theta(f(n))$

(II) Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \cdot \log n)$

(III) Se $f(n) = O(n^{\log_b a - \varepsilon})$ p/ algum $\varepsilon > 0$, então: $T(n) = \Theta(n^{\log_b a})$

Teorema Mestre - Exemplos

1. $T(n) = 1\,T(n/3) + n$

   - Simplificado:

   - Generalizado:

Teorema Mestre - Exemplos

1. $T(n) = 9\,T(n/3) + n$

- Simplificado:

$a: 9$
$b: 3$
$d: 1$

$\rightarrow \log_b a = 2 > 1 \Rightarrow T(n) = O(n^2)$

- Generalizado: $f(n) = n \rightarrow$ Relação entre $f(n)$ e $n^{\log_b a} = n^2$

$f(n) \in O(n^{2-\varepsilon})$

$\Downarrow$

$T(n) = \Theta(n^2)$

Teorema Mestre - Exemplos

1. $T(n) = T(2n/3) + 1$

   - Simplificado:

   - Generalizado:

Teorema Mestre - Exemplos

1. $T(n) = T(2n/3) + 1$

- Simplificado: $a = 1$, $b = 3/2$, $d = 0$

$$\log_b a = 0 = d$$

$$T(n) = O(n^d \cdot \log n) = O(\log n)$$

- Generalizado:

$$f(n) = 1 \qquad n^{\log_b a} = 1$$

$$f(n) = \Theta(n^{\log_b a}) \rightarrow T(n) = \Theta(\log n)$$

Teorema Mestre - Exemplos

1. $T(n) = 3T(n/4) + n \log n$

- Simplificado:

- Generalizado:

$f(n) = n \log n \qquad n^{\log_b a} = n^{\log_4 3} \qquad \log_4 3 < 1$

$n \log n > n^{\log_4 3}$

$T(n) = \Theta(n \log n)$

Condição de regularidade:

$a f(n/b) \leq c \cdot f(n)$

$3 \frac{n}{4} \log(n/4) \leq c \cdot n \log n$

Seja $c = 3/4$

$3/4\, n \log(n/4) \leq 3/4\, n \log n$

$\log(n/4) \leq \log(n)$

$\log n - \log(n/4) \geq 0$

$\log\left(\frac{n}{n/4}\right) \geq 0$

$\log 4 \geq 0$ (T)

Procura Binária

```
BinSearch(A, l, r, v)
    m = ⌊(l + r)/2⌋
    vm = A[m]
    if vm == v

    if r ≤ l

    if vm > v

    else
```

Procura Binária

```
BinSearch (A, l, r, v)
    m = ⌊(l+r)/2⌋
    vm = A[m]
    if vm == v
        return m
    if r ≤ l
        return NULL
    if vm > v
        return BinSearch (A, l, m-1, v)
    else
        return BinSearch (A, m+1, r, v)
```

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

$a - 1$

$b - 2$

$d - 0$

$\log_b a = \log_2 1 = 0 = d$

$T(n) = O(n \log n)$

Procura Binária

```
BinSearch (A, l, r, v)
    m = ⌊ l+r/2 ⌋
    vm = A [m]
    if vm == v
        return m
    if r ≤ l
        return NULL
    if vm > v
        return BinSearch (A, l, m-1, v)
    else
        return BinSearch (A, m+1, r, v)
```

- Exemplo:

1	3	5	6	7	8	14	18
1	2	3	4	5	6	7	8

$v = 8$

① $l_1 = 1,\ r_1 = 8$
$m_1 = \lfloor (1+8)/2 \rfloor = 4$
$A[m_1] = 6 < 8$

② $l_2 = 5,\ r_2 = 8$
$m_2 = \lfloor 5+8/2 \rfloor = 7$
$A[m_2] = 14 > 8$

③ $l_3 = 5,\ r_3 = 6$
$m_3 = \lfloor 6+5/2 \rfloor = 5$
$A[m_3] = 7 < 8$

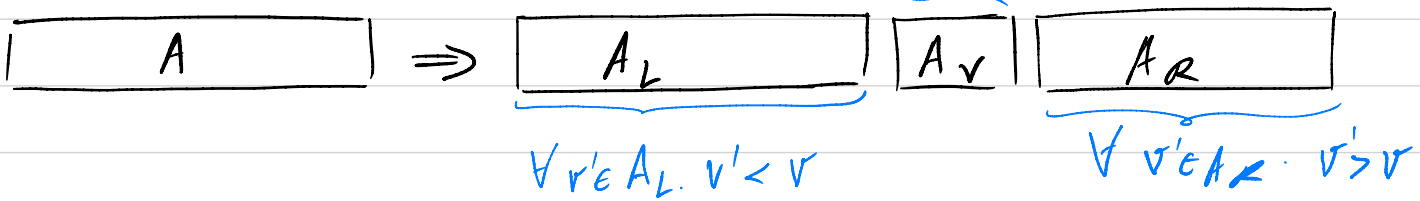
④ $l_4 = 6,\ r_4 = 6$
$m_4 = 6$
$A[m_4] = 8$ ✓

Elemento de Ordem k

- Selection (A, k) – k-ésimo elemento mais pequeno do array

$$Median(A) = Selection(A, \lfloor |A|/2 \rfloor)$$

- Estratégia:
  - Escolha aleatoreamente um valor v e re-arranje os elementos do array original como se segue:



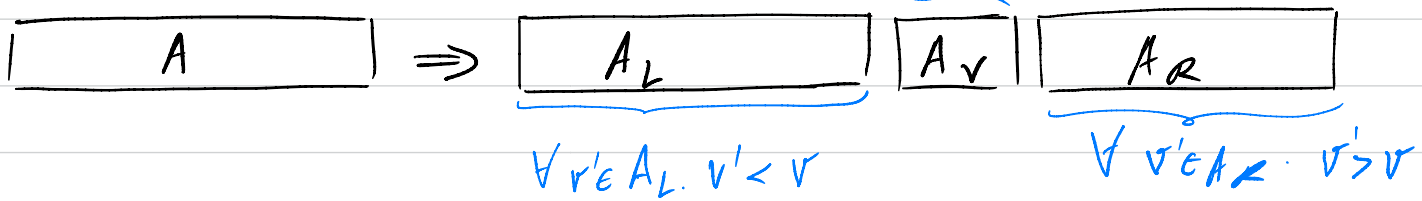
$$Selection(A, k) = \begin{cases}$$

Elemento de Ordem k

- Selection (A, k) – k-ésimo elemento mais pequeno do array

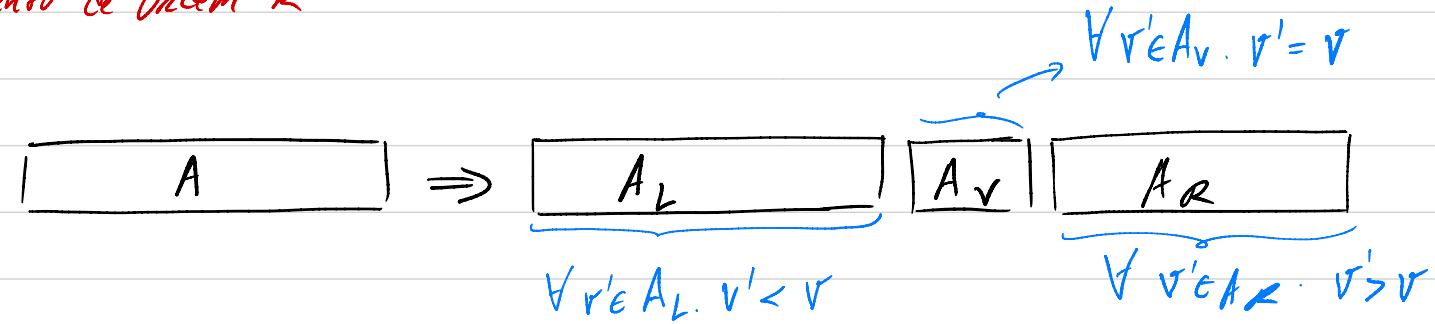
$$Median(A) = Selection(A, \lfloor |A|/2 \rfloor)$$

- Estratégia:
  - Escolha aleatoreamente um valor v e re-arranje os elementos do array original como se segue:



$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_v| \\ Select(A_R, k - |A_L| - |A_v|) & c.c. \end{cases}$$

Elemento de Ordem k

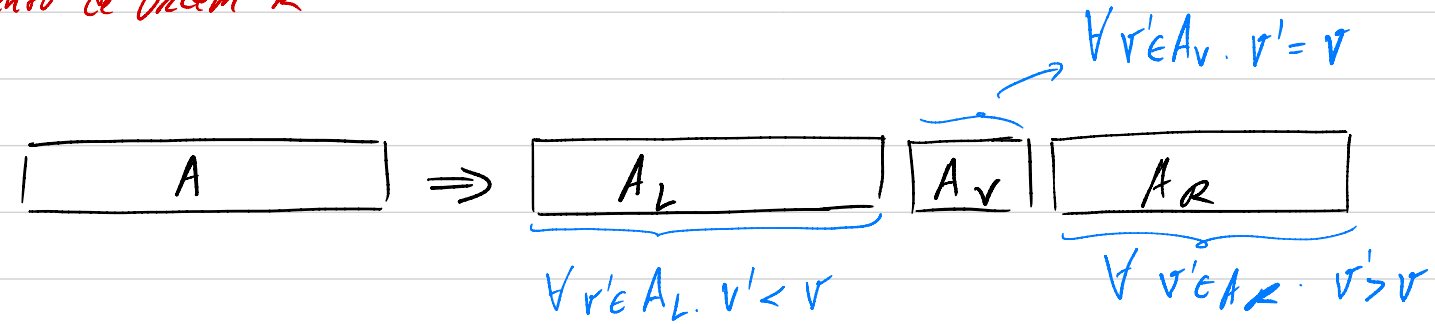


$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_V| \\ Select(A_R, k - |A_L| - |A_V|) & c.c. \end{cases}$$

Complexidade

- Melhor caso: Conseguimos sempre dividir o array em metade:

Elemento de Ordem k



$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_V| \\ Select(A_R, k - |A_L| - |A_V|) & c.c. \end{cases}$$

Complexidade

- Melhor caso: Conseguimos sempre dividir o array em metade:

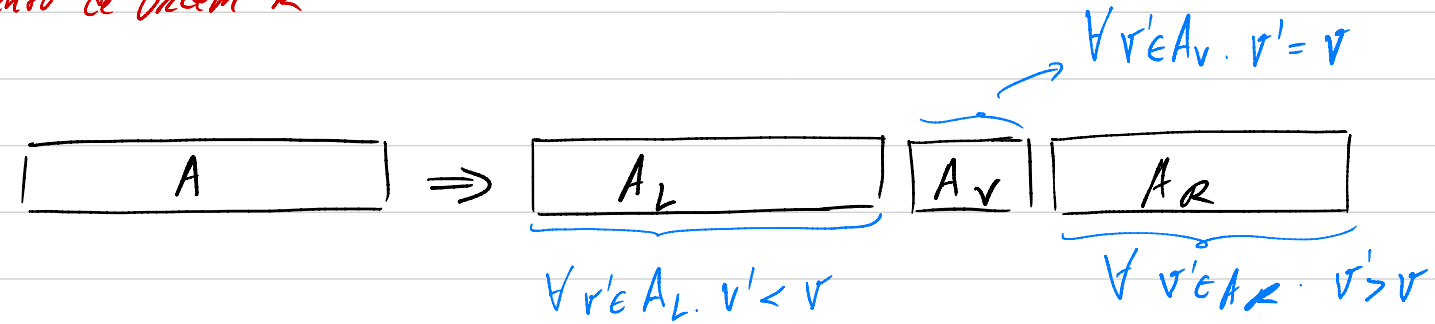
$$T(n) = T(n/2) + \Theta(n)$$

$a = 1 \quad \log_b a = 0 < 1 \Rightarrow T(n) = O(n)$

$b = 2$

$d = 1$

Elemento de Ordem k

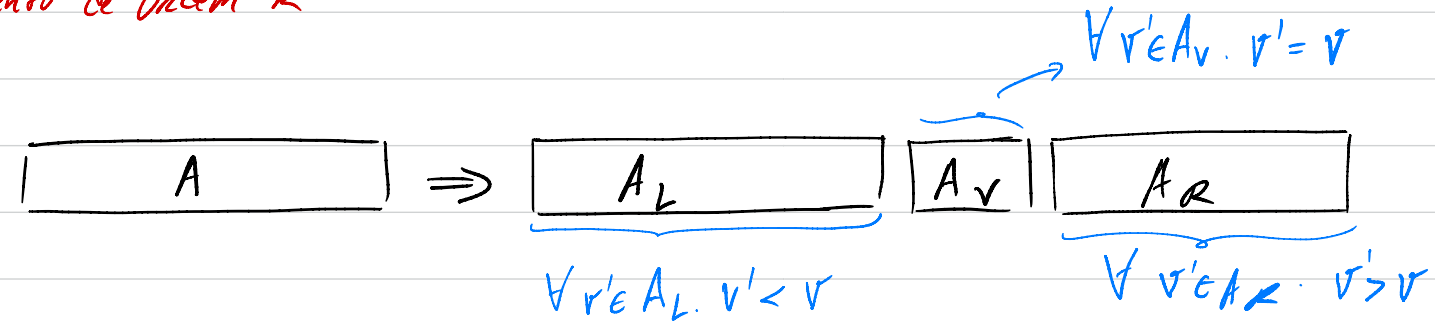


$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_v| \\ Select(A_R, k - |A_L| - |A_v|) & c.c. \end{cases}$$

Complexidade

- Pior caso: Escolhemos sempre o último:

Elemento de Ordem k



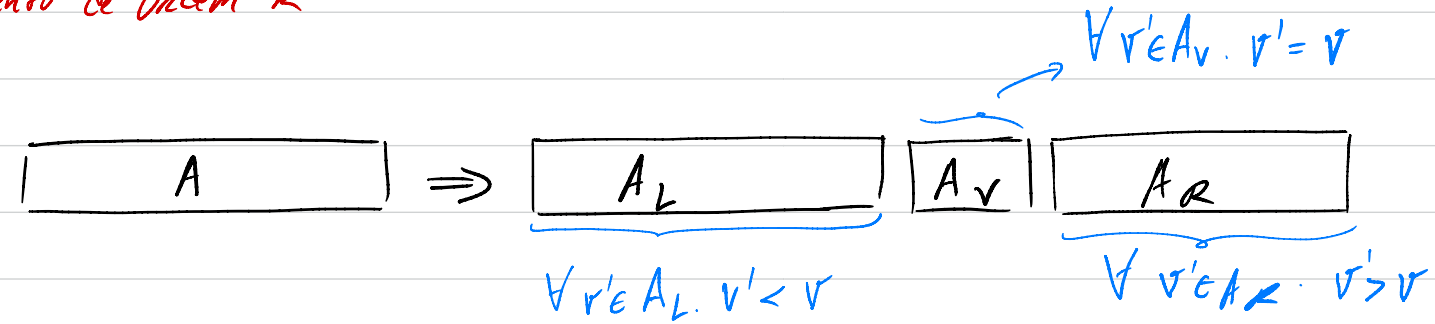
$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_V| \\ Select(A_R, k - |A_L| - |A_V|) & c.c. \end{cases}$$

Complexidade

- Pior caso: Escolhemos sempre o último:

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

Elemento de Ordem k

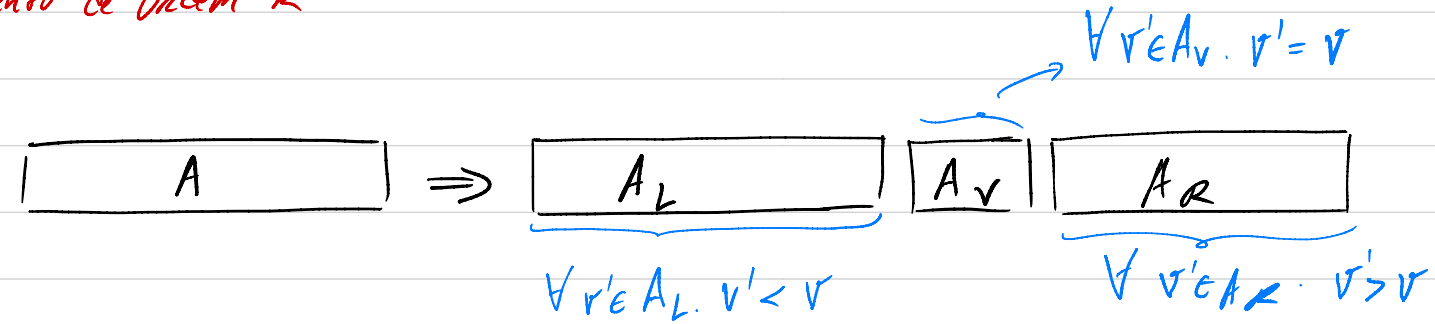


$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & Se \quad |A_L| < k \leq |A_L| + |A_V| \\ Select(A_R, k - |A_L| - |A_V|) & c.c. \end{cases}$$

Complexidade

- Caso Médio: Escolhemos sempre o último:

Elemento de Ordem k



$$Selection(A, k) = \begin{cases} Select(A_L, k) & se \quad k \leq |A_L| \\ v & se \quad |A_L| < k \leq |A_L| + |A_V| \\ Select(A_R, k - |A_L| - |A_V|) & c.c. \end{cases}$$

Complexidade

- Caso Médio: Escolhemos sempre o último:

$$T(n) = T\left(\tfrac{2}{3} \cdot n\right) + O(n)$$

$a = 1$

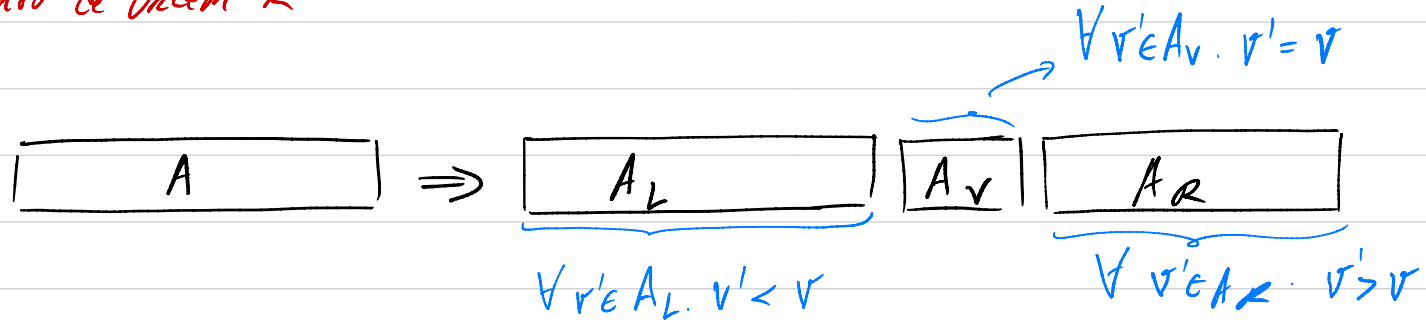
$b = 3/2$

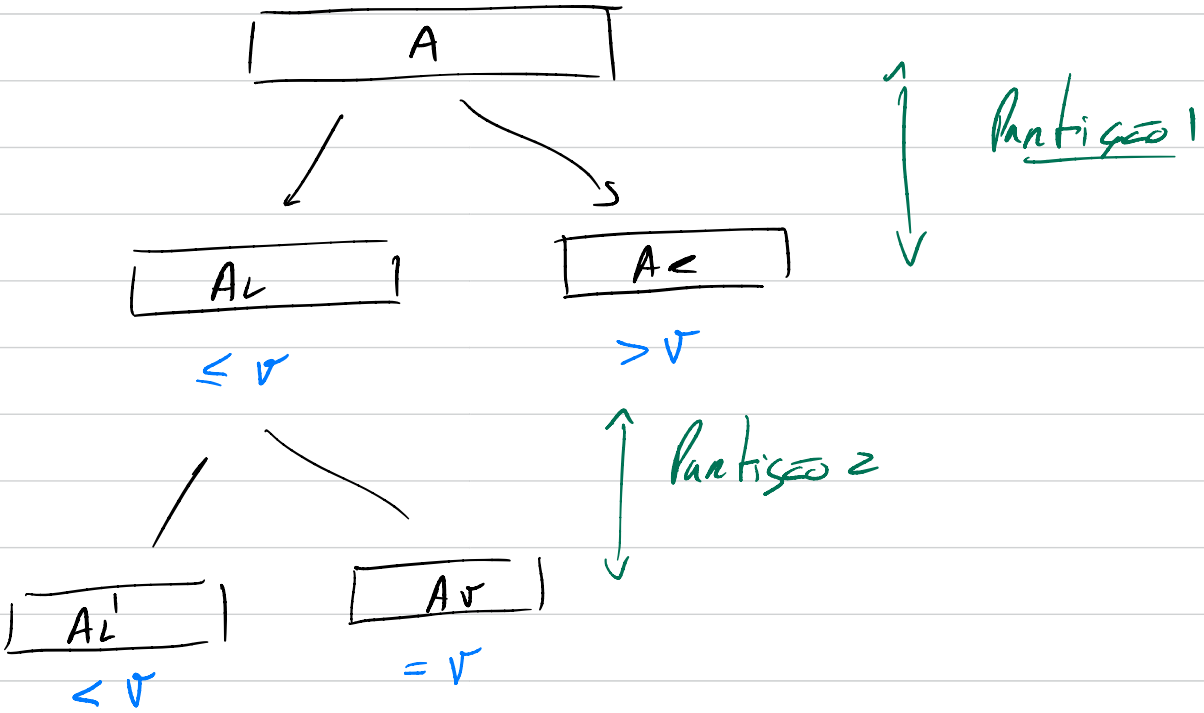
$d = 1$

$\log_b a = 0 < 1$

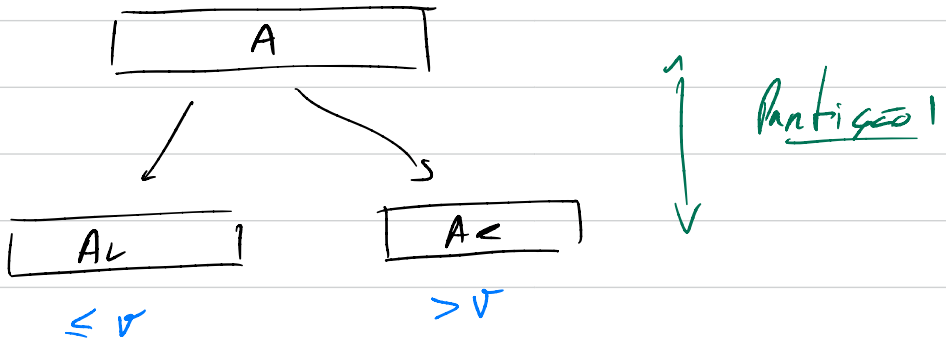
$\Rightarrow T(n) = O(n)$

Elemento de Ordem k





Elemento de Ordem k



Operação de Partição 1

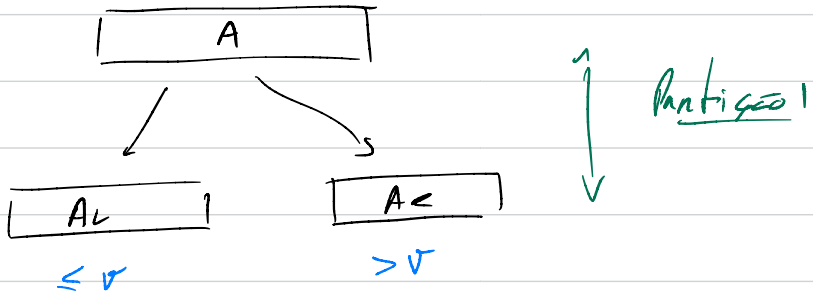
j → próxima posição do array a ser analisada

41	10	3	1	30	25	8	44	16

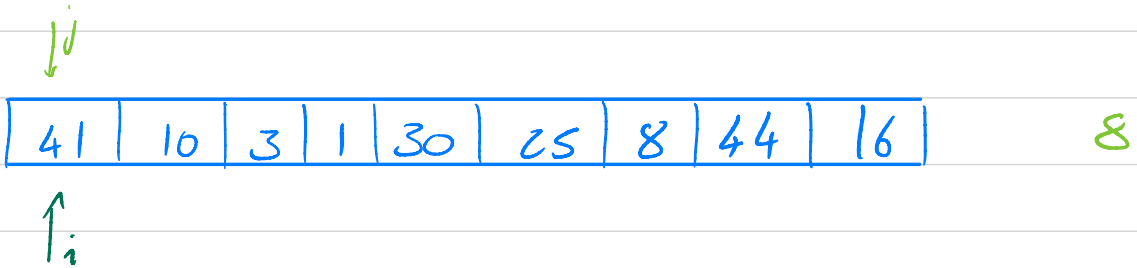
8

i → próxima posição livre para valores ≤ k

Elemento de Ordem k

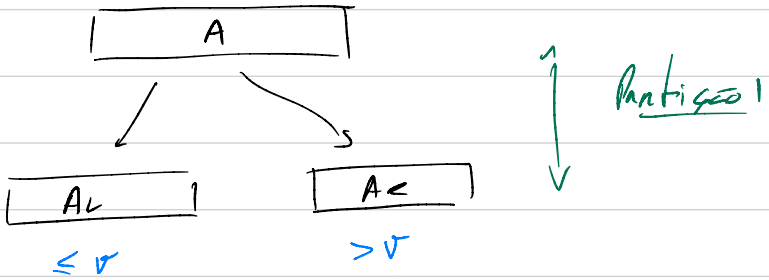


Operação de Partição

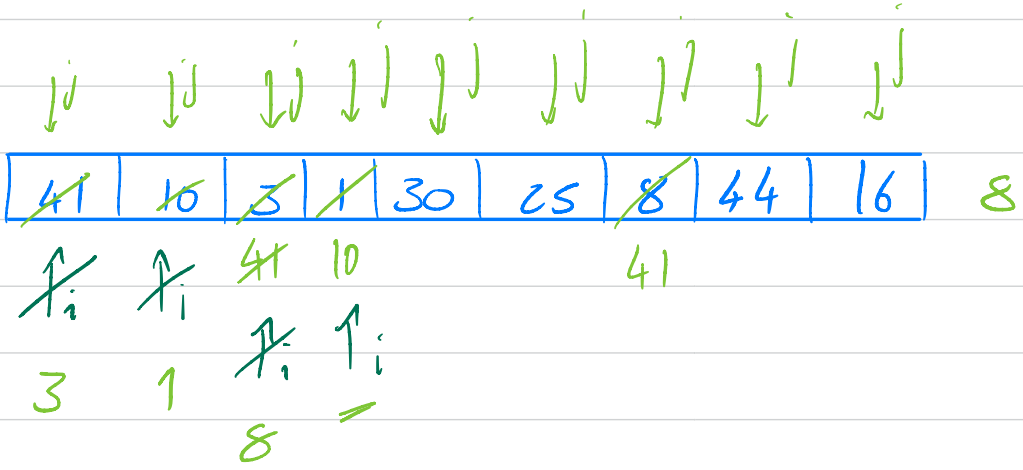


8

Elemento de Ordem k

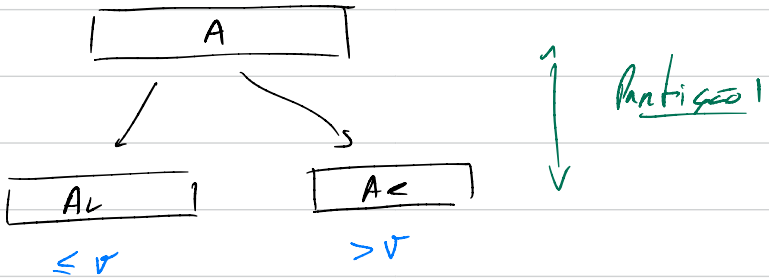


Operação de Partição 1

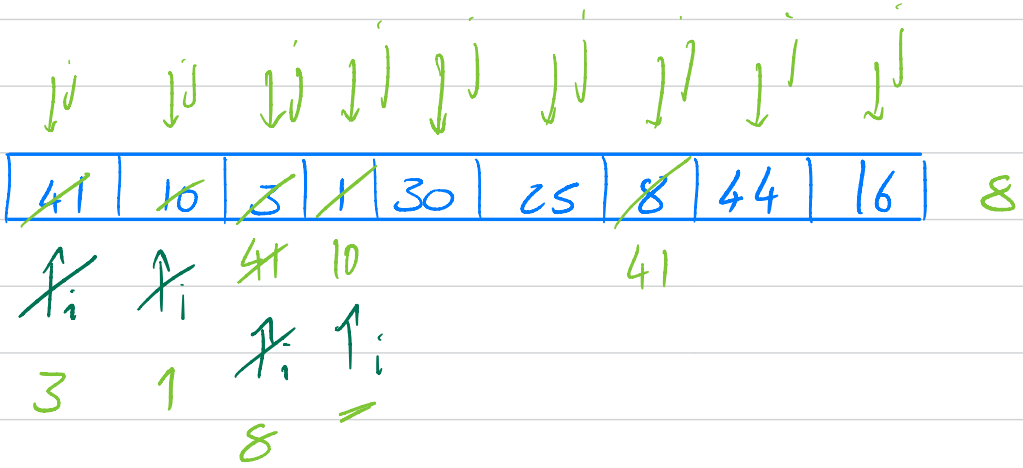


3	1	8	10	30	25	41	44	16

Elemento de Ordem k



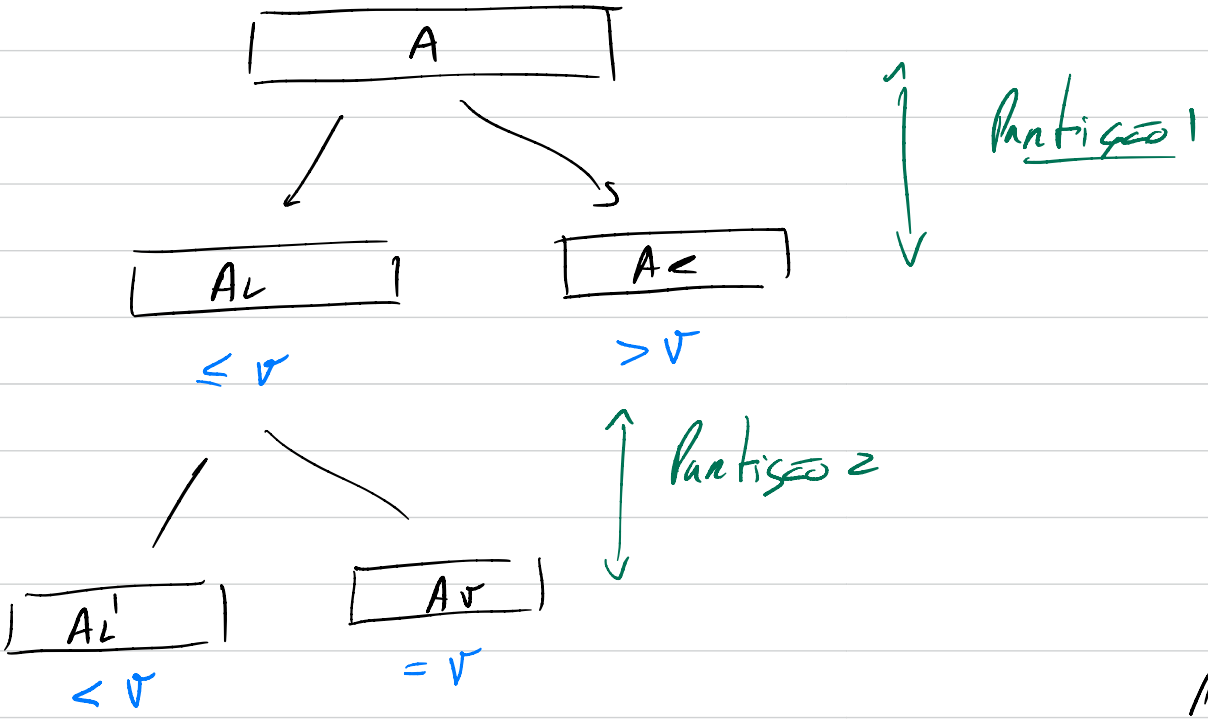
Operação de Partição 1




```
Partition1 (A, l, r, v)
    i := l
    for j = l to r
        if A[j] ≤ v
            swap(A, i, j)
            i := i+1
    return i
```

$T(n) = O(n)$

Elemento de Ordem k




```
Partition1 (A, l, r, v)
    i := l
    for j = l to r
        if A[j] ≤ v
            swap(A, i, j)
            i := i+1
    return i
```

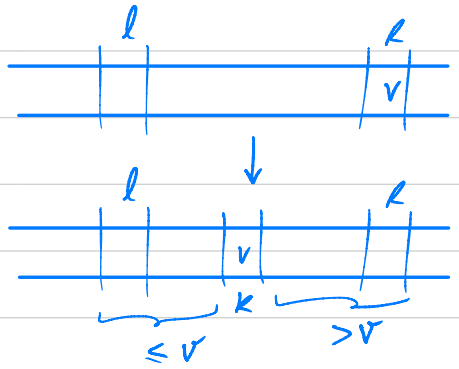
$T(n) = O(n)$

```
Partition2 (A, l, r, v)
    i := l
    for j = l to r
        if A[j] == v
            swap(A, i, j)
            i := i+1
    return i
```

QuickSort

```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

A:



} Partition(A, l, r)

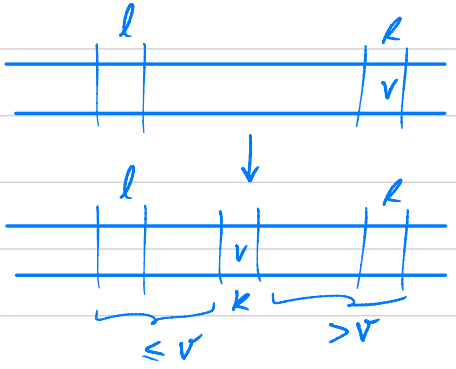
Exemplo:

2	8	7	1	3	5	6	4

QuickSort

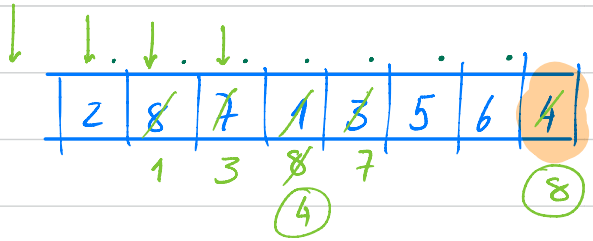
```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

A:



} Partition(A, l, r)

Exemplo:

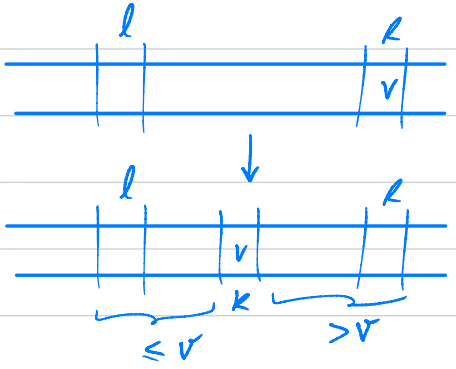



```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

QuickSort

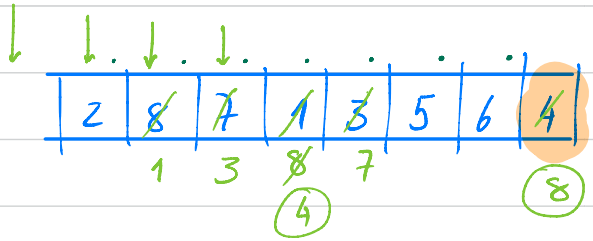
```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

A:



} Partition(A, l, r)

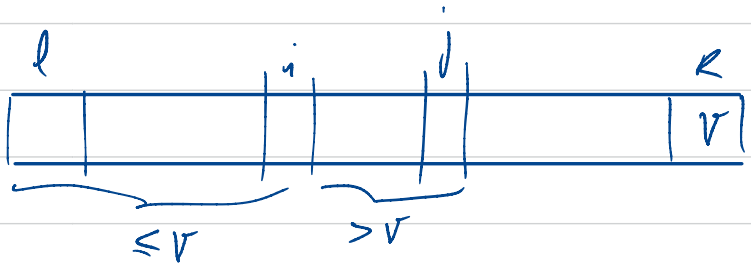
Exemplo:




```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

Partition - Invariante

```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

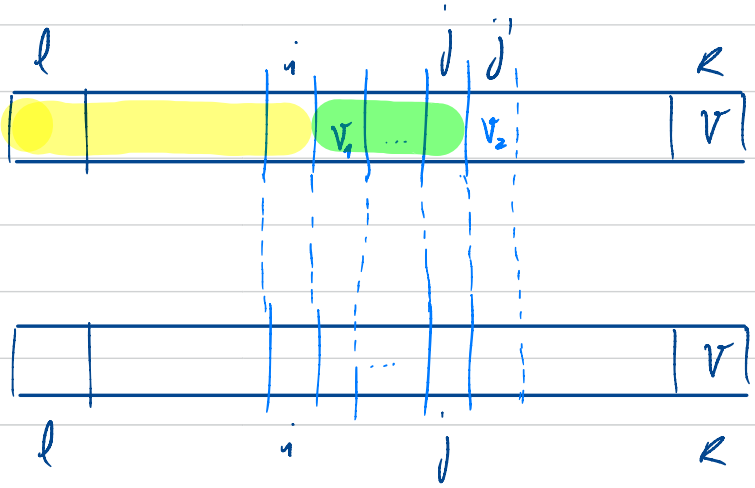


O invariante é mantido em cada iteração?

- $j' = j+1$
- 2 casos

Invariante?

- Caso I

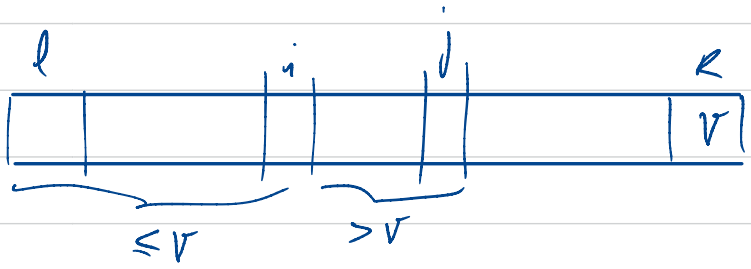


≤ v

> v

Partition - Invariante

```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

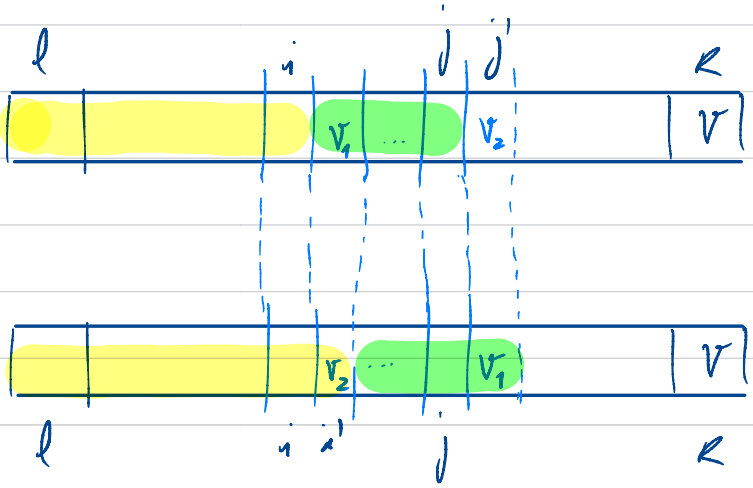


O invariante é mantido em cada iteração?

- $j' = j+1$
- 2 casos
  - $A[j'] < v$
  - $A[j'] > v$

Invariante?

- Caso I ($A[j'] \leq v$)

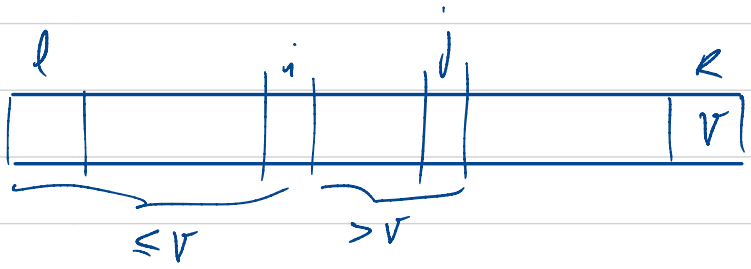


≤ v

> v

Partition - Invariante

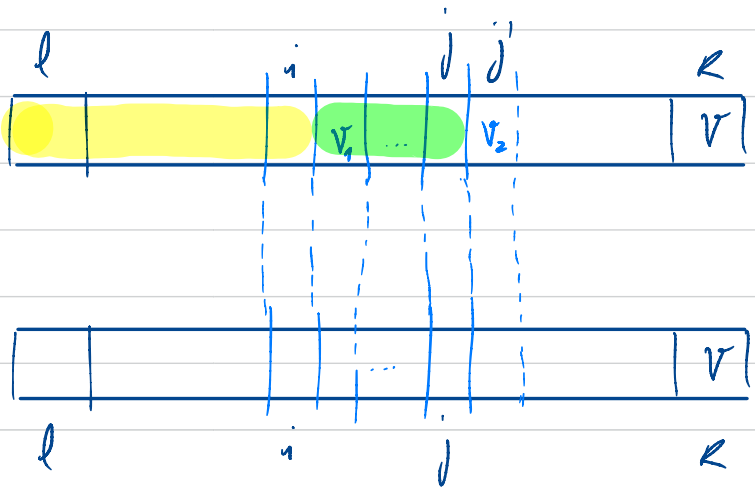
```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```



O invariante é mantido em cada iteração?

- $j' = j+1$
- 2 casos
  - $A[j'] < v$
  - $A[j'] > v$

Invariante?



- Caso II ($A[j'] > v$)

v_2

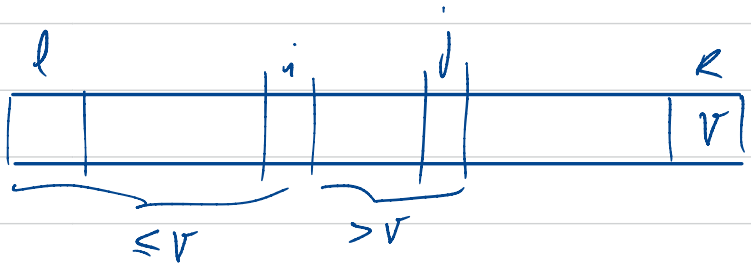
≤ v

> v

Partition - Invariante

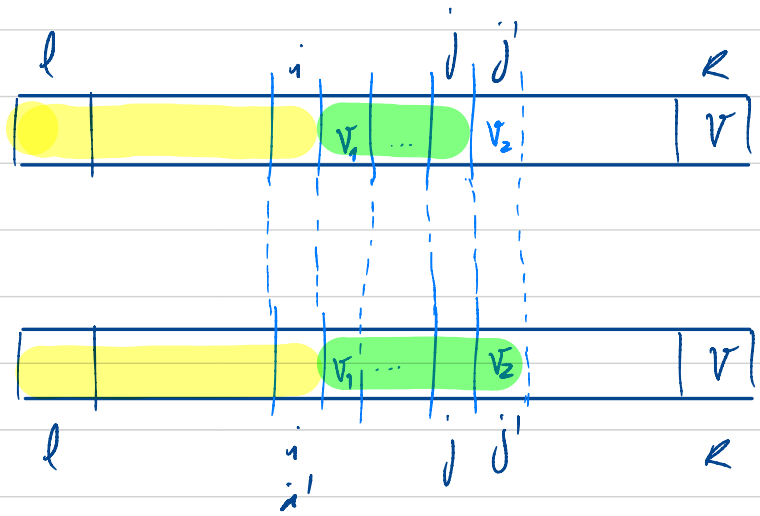
```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

Invariante?



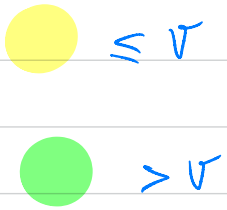
O invariante é mantido em cada iteração?

- $j' = j+1$
- 2 casos
  - $A[j'] < v$
  - $A[j'] > v$



- Caso II ($A[j'] > v$)

v_2



QuickSort - Complexidade

```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

Complexidade

QuickSort - Complexidade

```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

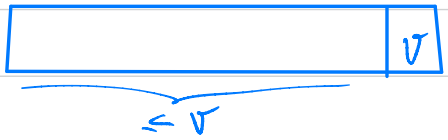
```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

Complexidade

- Nº de iterações do loop
  $r - l + 1 = r - l$

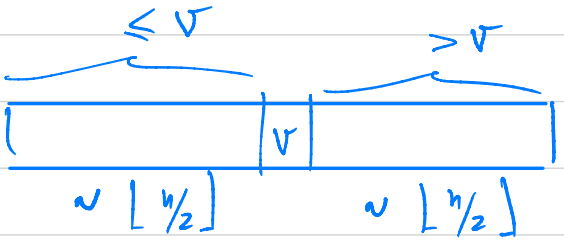
$O(r-l) = O(n)$

- Pior Caso



$T(n) = O(n) + T(n-1)$
$= O(\sum_{i=1}^{n} O(i))$
$= O(n^2)$

- Melhor Caso



$T(n) = 2T(n/2) + O(n)$

$T(n) = O(n \log n)$

QuickSort - Complexidade

```
QuickSort(A, i, j)
    if i < j
        k = Partition(A, i, j)
        QuickSort(A, i, k-1)
        QuickSort(A, k+1, j)
```

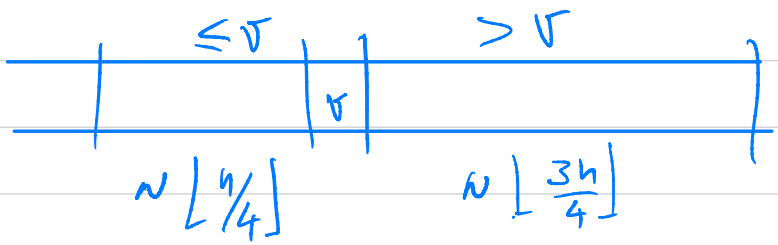
```
Partition(A, l, r)
    i := l-1 ; j := l
    v := A[r]
    for j = l to r-1
        if A[j] < v
            i := i+1
            swap(A, i, j)
    swap(A, i+1, r)
    return i+1
```

Complexidade

- Nº de iterações do loop
  $r - l + 1 = r - l$

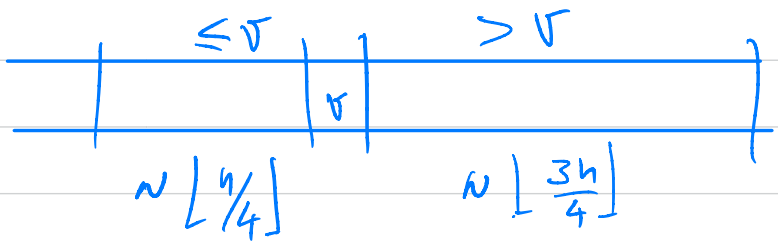
$O(r-l) = O(n)$

- Caso Médio (Intuição)

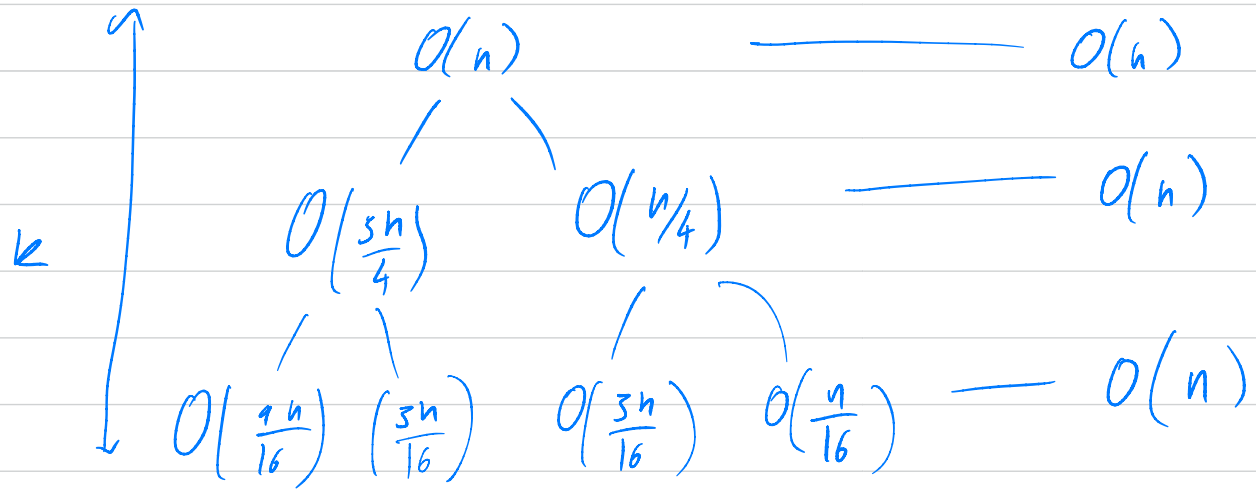


$$T(n) = T(n/4) + T\left(\frac{3n}{4}\right) + O(n)$$

- Caso Médio (Intuição)



$$T(n) = T(n/4) + T\left(\frac{3n}{4}\right) + O(n)$$



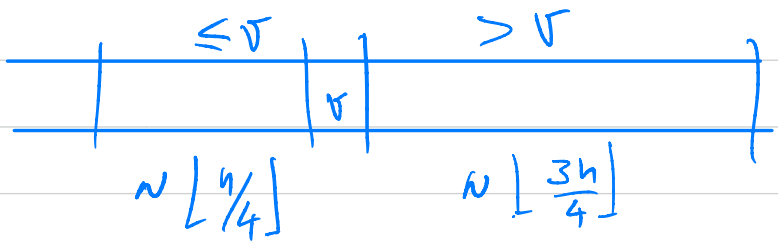
$$T(n) = O(n \lg n)$$

$k = ?$

$$\frac{n}{(4/3)^k} = 1 \iff n = (4/3)^k$$

$$\iff k = \log_{4/3} n$$

- Caso Médio (Intuição)



- $T(n) = T(n/4) + T(\frac{3n}{4}) + O(n)$

- $T(n) = O(n \lg n)$

$T(n+1) =$