Aula 2

- Teorema Mestre (Revisão)
- Divisão & Conquista I Exemplos)
  - Produto Binário
  - Elemento Majoritário
  - Mediana
- Quicksort
  - Complexidade: Pior caso e Caso Médio

Teorema 1 [Teorema Mestre - Simplificado]

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

então:

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

Exemplo 1 [Procura Binária]

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

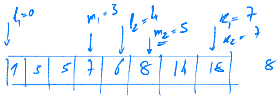
↳ porque não m-1?

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

$a = 1 \quad b = 2 \quad d = 0$

$\log_b a = \log_2 1 = 0$

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



$m_1 = \lfloor \frac{0+7}{2} \rfloor = 3$

$m_2 = \lfloor \frac{4+7}{2} \rfloor = 5$

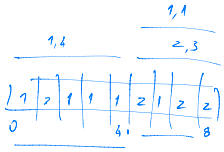
Exemplo 2 [Elemento Maioritário]

Diremos que um array $A[1 \ldots n]$ tem um elemento maioritário se mais de metade dos entradas do array estão associadas ao mesmo valor, o elemento maioritário do array.

Os elementos do array não são ordenáveis, conseguimos apenas testar a igualdade entre os elementos do array: $A[i] == A[j]$.

Solução

```
Find Maj (A, i, j)
    if (i == j) return (A[i], 1)
    m = ⌊(i+j)/2⌋
    (v1, k1) = Find Maj (A, i, m)
    (v2, k2) = Find Maj (A, m+1, j)
    if (v1 == v2) return (v1, k1+k2)
    if (v1 > v2)
        k1' = Count (A, v1, i, m)
        if (k1' + k1 ≥ ⌈((j-i)+1)/2⌉)
            return (v1, k1' + k1)
    else
        k2' = Count (A, v2, m+1, j)
        if (k2 + k2' ≥ ⌈(j-i)+1/2⌉)
            return (v2, k2' + k2)
```



$\frac{0+8}{2} = 4 \qquad \lceil \frac{1}{2} \rceil$

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

$a = 2 \quad b = 2 \quad d = 1$

$\log_b a = 1 = d$ → $O(n \log n)$

Exemplo 3 [Mediana]

Dado um array $A = [a_1, \dots, a_n]$, o valor mediano corresponde ao elemento $\lfloor \frac{n}{2} \rfloor$ do array A', correspondente ao array A ordenado.

Exemplo: $[41, 1, 10, 30, 25] \Rightarrow \underline{25}$

Solução 1: Ordenar o array A para obter A'
Calcular $\lfloor n/2 \rfloor = k$
Retornar $A'[k]$
} complexidade: $O(n \lg n)$

Solução 2: Uma solução $\bar{q}$ é em média melhor

Generalização do problema de encontrar o valor mediano.

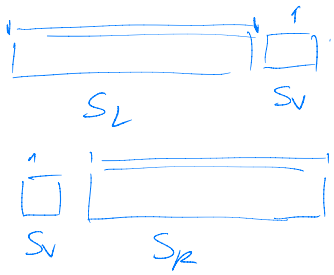
- Selection (S, k) ⇒ returns the k^{th} smallest element of S
  - S: a list of numbers
  - k: an integer

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

* Escolhe aleatoriamente um valor $v \in S$ e reorganiza os elementos do array original como se segue:

↳ $\forall v' \in S_L . \ v' < v$
$\forall v' \in S_V . \ v' = v$
$\forall v' \in S_R . \ v' > v$

- $$\text{Selection}(S,k) = \begin{cases} \text{Selection}(S_L, k) & \text{if } k \le |S_L| \\ v & \text{if } |S_L| < k \le |S_L| + |S_V| \\ \text{Selection}(S_R, k - |S_L| - |S_V|) & \text{if } k > |S_L| + |S_V| \end{cases}$$



* Complexidade

Melhor caso:

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

$a = 1 \quad b = 2 \quad d = 1$

$\log_b a = 0 < d \Rightarrow \Theta(n)$

Pior Caso:

$$\begin{aligned} T(n) &= T(n-1) + \Theta(n) \\ &= T(n-2) + \Theta(n-1) + \Theta(n) \\ &= \Theta(1) + \dots + \Theta(n-1) + \Theta(n) \\ &= \sum_{j=1}^{n} \Theta(j) \\ &= \Theta\left(\sum_{j=1}^{n} j\right) \\ &= \Theta\left(\frac{n(n-1)}{2}\right) = \Theta(n^2) \end{aligned}$$

* Caso Médio: Assumimos q no caso médio
  o array diminui de 25% a cada
  iteração do algoritmo. Porquê 25%?
  - Qual é a probabilidade de escolhermos um valor entre o 1º quartil e o 3º quartil? 1/2
  - A cada 2 iterações o array onde estamos a operar decresce 25%

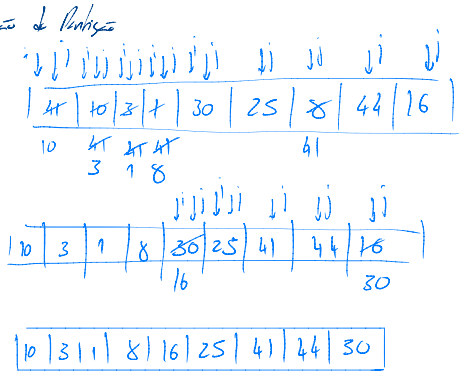
$$T(n) = T\left(\frac{3}{4}n\right) + O(n)$$

→ o custo da iteração inútil é dominado por $O(n)$

$a = 1 \quad b = \frac{4}{3} \quad d = 1 \quad \Rightarrow T(n) = O(n)$

$\log_{\frac{1}{3}} 1 = 0 < d$

- Operação de Partição



16

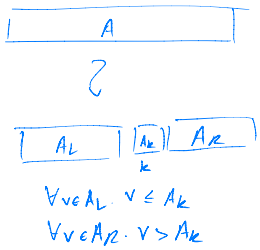
```
selectPartitionStep1(A, l, r, k)
    i = l
    for j = l to r
        if (A[j] < k)
            swap(A, i, j)
            i = i + 1
    return i
```

```
selectPartitionStep2(A, l, r, k)
    i = l
    for j = l to r
        if (A[j] == k)
            swap(A, i, j)
            i = i + 1
```

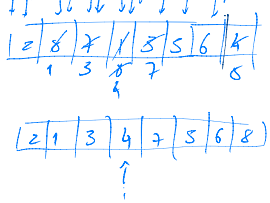
```
selectPartition(A, l, r, k)
    i = selectPartitionStep1(A, l, r, k)
    selectPartitionStep2(A, i, r, k)
```

Algoritmo 1 [Quick Sort]

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



- How does the partitioning work?



- Invariant of the partitioning

Quando o loop termina: $j = r - 1$

$\forall 1 \leq k < i \cdot A[k] \leq A_0[r]$

$\forall i \leq k < r \cdot A[k] > A_0[r]$

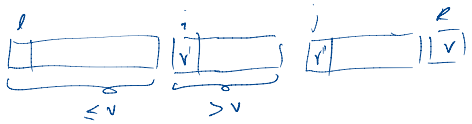
Quando o algoritmo termina:

$\forall l \leq k < i \cdot A[k] \leq A_0[r]$

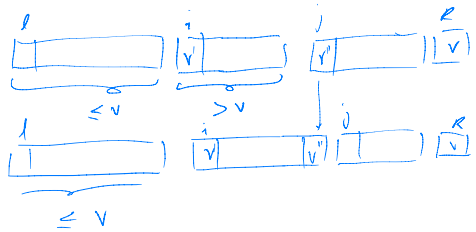
$A[i] = A_0[r]$

$\forall i < k \leq r \cdot A[k] > A_0[r]$

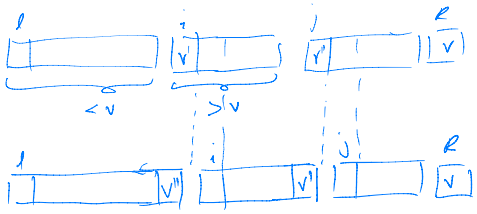
- Two cases



$v'' > v \Rightarrow$



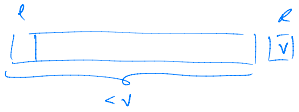
$v'' \leq v$



Esboço de prova de que o algoritmo mantém o invariante

```
Partition (A, l, r)
    Partition (A, l, r)
    i = l
    v_R = A[r]
    for j = l to r-1
        if A[j] ≤ v_R
            swap (A, i, j)
            i = i + 1
    swap (A, i, r)
    return i
```

- Complexidade



  - Melhor caso

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

$a = 2 \quad b = 2 \quad d = 1$

$\log_b a = 1 = d \longrightarrow T(n) = O(n \log n)$

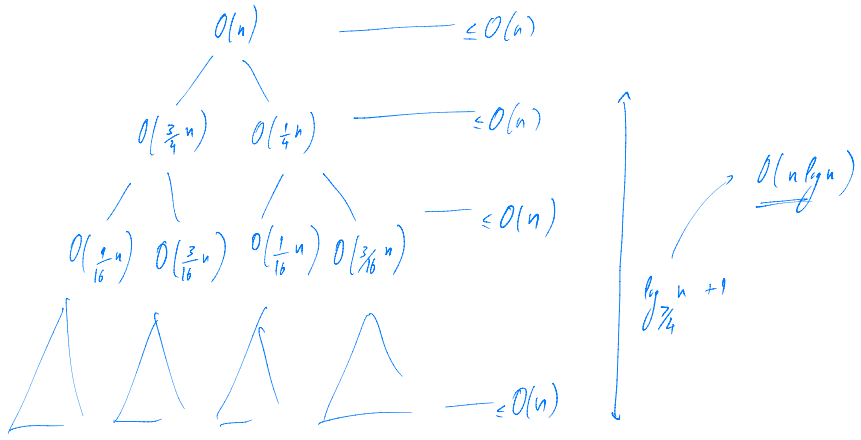
  - Pior caso

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

$$T(n) = \sum_{k=1}^{n} O(k) = O(n^2)$$

  - Caso Médio

$$T(n) = T\left(\tfrac{3}{4}n\right) + T\left(\tfrac{1}{4}n\right) + O(n)$$



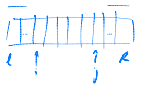
Exemplo 4 [Soma de nºs]

3	6	8	11	14	23	31	33

i i i i — 42 — j j

```
Find Sum (A, i, j, v)
    if (j < i) return None
    if (A[i] + A[j] == v) return Sum (i, j)
    if (A[i] + A[j] < v)
        return FindSum (A, i+1, j, v)
    else return Find Sum (A, i, j-1, v)
```

- Invariante



$\forall l \le k_1 < i \;.\; \forall j < k_2 \le r \;.\; A[k_1] + A[k_2] \neq v$

$\forall l \le k_1, k_2 \le r \;.\; k_1 \le k_2 \Rightarrow A[k_1] \le A[k_2]$

Quando o algoritmo termina: $j < i$

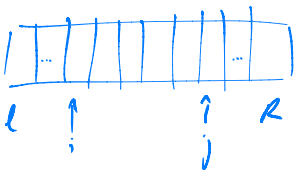
- $\forall\, l \le k_1 < i \;.\; \forall\, j < k_2 \le r \;.\; A[k_1] + A[k_2] \neq v$

  $\forall\, l \le k_1 < i \;.\; \forall\, j < k_2 \le r \;.\; A[k_2] + A[k_1] \neq v$

- $\forall\, j < k_1 \le r \;.\; \forall\, l \le k_2 < i \;.\; A[k_1] + A[k_2] \neq v$

  $\forall\, l \le k_1 \le r \;.\; \forall\, l < k_2 \le r \;.\; A[k_1] + A[k_2] \neq v$

Manutenção do invariante:



$A[i] + A[j] > v$

$i' = i + 1$

$j' = j$

Temos de provar $\bar{q}$:

$\forall l \le k_1 < i'.\ \forall j' < k_2 \le R.\ A[k_1] + A[k_2] \neq v$

$\Leftrightarrow \forall l \le k_1 < i+1.\ \forall j < k_2 \le R.\ A[k_1] + A[k_2] \neq v$

$\Leftarrow \forall l \le k_1 < i.\ \forall j < k_2 \le R.\ A[k_1] + A[k_2] \neq v$ (invariante)

$\wedge \underbrace{\forall j < k_2 \le R.\ A[i] + A[k_2] \neq v}_{\text{Temos de provar isto}}$

$\forall j < k_2 \le R.\ A[k_2] \geq A[j]$

$\forall j < k_2 \le R.\ A[i] + A[k_2] \geq A[i] + A[j] > v$

$\lightning$