Aula 1

1) Calcular os nºs de Fibonacci
2) Notação Assimptótica
3) Dividir - para - conquistar - Merge Sort
4) Teorema Mestre

Exemplo 1 [Calcular o $n^{\text{ésimo}}$ de Fibonacci]

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{if } n > 1 \\ 1 & \text{if } n = 0 \text{ ou } n = 1 \end{cases}$$

- Points to discuss:
  - Is this a tail-recursion?
  - How do we count the elementary steps?

- **Solução 1**

```
Fib (n)
  if n==0 or n==1
    return 1
  else return Fib(n-1) + Fib(n-2)
```

$$T(n) = \begin{cases} T(n-1) + T(n-2) + 4 & \text{if } n > 1 \\ 3 & \text{if } n \leq 1 \end{cases}$$

(4 → pergunta)

* $T(n) > Fib(n)$

- **Solução 2**

```
Fib (n)
  let C[0..n] be a new array
  C[0] = 1
  C[1] = 1
  for i=2 to n
    C[i] = C[i-2] + C[i-1]
  return C[n]
```

$$T(n) = \begin{cases} 4 & \text{if } n \leq 1 \\ n+3 & \text{if } n > 1 \\ 4 + (n-2+1) & \end{cases}$$

- Comments
- Apparently linear time.

- **Solução 3**

```
Fib (n)
  if (n==0) or (n==1)
    return 1
  let c1 = 1
  let c2 = 1
  for i=2 to n
    let aux = c1
    c1 = c2
    c2 = c2 + aux
  return c2
```

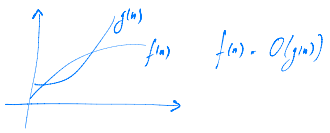
Definição 1 [Notação Assimptótica]

1) $O(g) = \{ f \mid \exists c, n_0.\ \forall n \geq n_0.\ 0 \leq f(n) \leq c \cdot g(n) \}$

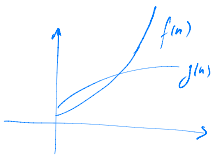
2) $\Theta(g) = \{ f \mid \exists c_1, c_2, n_0.\ \forall n \geq n_0.\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \}$

3) $\Omega(g) = \{ f \mid \exists c, n_0.\ \forall n \geq n_0.\ c \cdot g(n) \leq f(n) \}$

Mnemónica: $f = O(g) \sim f \leq g$
$f = \Theta(g) \sim f = g$
$f = \Omega(g) \sim f \geq g$



$f(n) = O(g(n))$



$f(n) = \Omega(g(n))$

Lema 1. $f = \Theta(g) \Leftrightarrow f = O(g) \wedge f = \Omega(g)$

Prova:

($\Rightarrow$)

Supomos que $f(n) = \Theta(g(n))$, concluímos da definição de Θ que existem constantes c_1, c_2 e n_0 tais que:

- $\forall n \geq n_0.\ c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

Donde se conclui que:

$\exists c, n_0.\ \forall n \geq n_0.\ f(n) \leq c \cdot g(n) \Rightarrow f(n) = O(g(n))$

$\exists c, n_0.\ \forall n \geq n_0.\ c \cdot g(n) \leq f(n) \Leftrightarrow f(n) = \Omega(g(n))$

($\Leftarrow$) Supomos que $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

Concluímos que existe c_1, c_2, n_0, n_0', tais que:

- $\forall n \geq n_0.\ f(n) \leq c_1 g(n)$
- $\forall n \geq n_0'.\ c_2 g(n) \leq f(n)$

Seja $n_0'' = \max(n_0, n_0')$, concluímos que:

$\forall n \geq n_0''.\ c_2 \cdot g(n) \leq f(n) \leq c_1 \cdot g(n)$

Do que se conclui que $f(n) = \Theta(g(n))$

Lema 2 [Transitividade]

a) $f = O(g) \wedge g = O(h) \Rightarrow f = O(h)$

b) $f = \Theta(g) \wedge g = \Theta(h) \Rightarrow f = \Theta(h)$

c) $f = \Omega(g) \wedge g = \Omega(h) \Rightarrow f = \Omega(h)$

Prof:

a) Supomos q $f(n) = O(g(n))$ e $g(n) = O(h(n))$.

Concluímos a partir da definição de O, que existem c_1, c_2, n_0 e n_0', tais que:

$\forall n \geq n_0.\ f(n) \leq c_1 \cdot g(n)$

$\forall n \geq n_0'.\ g(n) \leq c_2 \cdot h(n)$

Donde segue q:

$\forall n \geq \max(n_0, n_0').\ f(n) \leq c_1 c_2 \cdot h(n)$

Donde segue q $f(n) = O(h(n))$

Lema 3. $f = O(g) \Leftrightarrow g = \Omega(f)$

Prova.

$(\Rightarrow)$

Sabemos que $f = O(g)$, isto é, existem constantes $c \in n_0$ tais que:

$\forall n \geq n_0 \quad f(n) \leq c\, g(n)$

De onde se conclui que:

$\forall n \geq n_0 \quad \frac{1}{c} f(n) \leq g(n)$

De onde segue que $g = \Omega(f)$

Comentário: Dadas duas funções f e g uma forma simples de percebermos a sua ordem de magnitude relativa é observarmos para o limite do rácio:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} \begin{cases} +\infty, & f = \Omega(g) \\ c, & f = \Theta(g) \\ 0, & f = O(g) \end{cases}$$

Exemplo 2:

a) $f(n) = 100n + \log n$, $g(n) = n + (\log n)^2$

b) $f(n) = n$, $g(n) = (\log n)^3$

a) $$\lim_{n \to \infty} \frac{100n + \log n}{n + (\log n)^2} = \lim_{n \to \infty} \frac{100 + \frac{1}{n}}{1 + 2\log n \cdot \frac{1}{n}} = \lim_{n \to \infty} \frac{\frac{100n+1}{n}}{\frac{n + 2\log n}{n}} = \lim_{n \to \infty} \frac{(100n+1)\,n}{(n + 2\log n)\,n}$$

$$= \lim_{n \to \infty} \frac{100n + 1}{n + 2\log n} = \lim_{n \to \infty} \frac{100}{1 + \frac{2}{n}} = \lim_{n \to \infty} \frac{100}{\frac{n+2}{n}} = \lim_{n \to \infty} \frac{100n}{n+2} = \lim_{n \to \infty} \frac{100}{1} = 100$$

$f = \Theta(g)$

b) $$\lim_{n \to \infty} \frac{n}{(\log n)^3} = \lim_{n \to \infty} \frac{1}{3(\log n)^2 \frac{1}{n}} = \lim_{n \to \infty} \frac{n}{3(\log n)^2} = \lim_{n \to \infty} \frac{1}{6 \log n \frac{1}{n}} = \lim_{n \to \infty} \frac{n}{6 \log n} = \lim_{n \to \infty} \frac{1}{6 \cdot \frac{1}{n}} = \lim_{n \to \infty} \frac{n}{6} = \infty$$

$f = \Omega(g)$

Método de Divide & Conquista

1. Divida o problema num conjunto de subproblemas do mesmo tipo
2. Resolva cada um dos subproblemas
3. Combinar as soluções dos subproblemas p/ obter a solução do problema original

Algoritmo 1 [MergeSort]

```
MergeSort (A, p, r)
  if p < r
    let q = ⌊(p+r)/2⌋
    MergeSort (A, p, q)
    MergeSort (A, q+1, r)
    Merge (A, p, q, r)
```

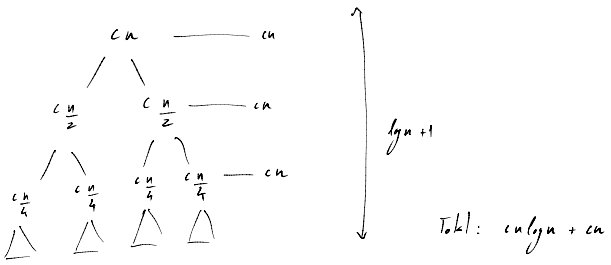
Complexity Analysis

$D(n) = \Theta(1)$

$S(n) = 2\,T(n/2)$

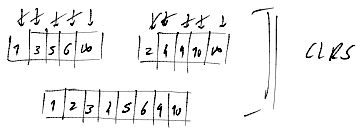
$C(n) = \Theta(n)$

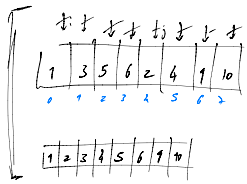
$$T(n) = \begin{cases} 2\,T(n/2) + cn & \text{if } n > 1 \\ c & \text{if } n = 1 \end{cases}$$



Total: $cn \lg n + cn$

```
Merge (A, p, q, r)
  let L[1..((q-p)+2)] be a new array
  let R[1..((r-(q+1))+2)] be a new array
  for i = 1 to (q-p)                    ↳ r-q-1+1 = r-q
    L[i] = A[p + i]
  for i = 1 to (r-(q+1))
    R[i] = A[q+1+i]
  L[(q-p)+2] = ∞
  R[(r-(q+1))+2] = ∞
  i = 0, j = 0, k = p
  while k ≤ r
    if L[i] ≤ R[j]
      A[k] = L[i]; k++, i++
    else A[k] = R[j]; k++; j++
```





Reallocate the entire memory and iterate over what is there

1 2 3 4 5

1

3, 3++ - - -

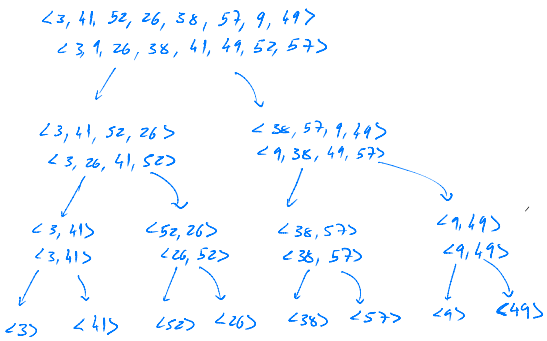
3 - 1 = 2

4 - 2 = 2

1 2

Example 3 [Execução do MergeSort]

$A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$



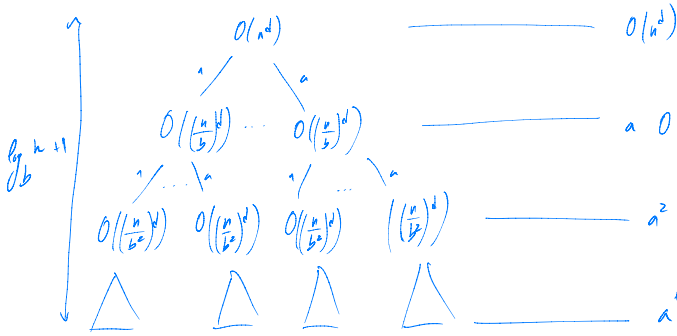
Teorema 1 [Teorema Mestre - Simplificado]

Se $T(n) = a\,T(\lceil n/b \rceil) + O(n^d)$ para 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. Simplificamos, sem perda de generalidade, que n é uma potência de b



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

- A complexidade do problema depende do rácio $\frac{a}{b^d}$.

Consideramos os seguintes 3 casos.

$a/b^d < 1$

$$\sum_{k=0}^{n} \left(\frac{a}{b^d}\right)^k < \frac{1}{1 - \frac{a}{b^d}}$$ A série geométrica é convergente.

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

$d > \log_b a \Rightarrow b^d > b^{\log_b a}$
$\Leftrightarrow b^d > a$
$\Leftrightarrow \frac{a}{b^d} < 1$

$a/b^d = 1$

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

$d = \log_b a \Leftrightarrow b^d = a$
$\Leftrightarrow \frac{a}{b^d} = 1$

$a/b^d > 1$

O custo da série de potências é dominado pelo custo último termo: $\left(\frac{a}{b^d}\right)^{\log_b n}$

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

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

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

$d < \log_b a \Rightarrow b^d < b^{\log_b a}$
$\Leftrightarrow b^d < a \Leftrightarrow \frac{a}{b^d} > 1$

∎

Exemplo 4 [Aplicação do Teorema Mestre]

Merge Sort: $T(n) = 2T(n/2) + O(n)$

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

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

- $T(n) \leq \begin{cases} c & , n \leq n_0 \\ 1T(n/3) + n^2 & , n > n_0 \end{cases}$ Ficha 1 (2008/2009)

$a = 1 \quad b = 3 \quad d = 2$

$\log_b a = 2 \quad d = 2 \quad O(n^2 \log n)$

- $T(n) \leq \begin{cases} c & , n \leq n_0 \\ 8T(n/2) + n^4 & , n > n_0 \end{cases}$ Ficha 1 (2008/2009)

$a = 8 \quad b = 2 \quad d = 4$

$\log_b a = 3 \quad 3 < 4 \quad O(n^4)$

Food for thought

1) Algoritmo p/ remover elementos duplicados
2) Binary search when we do not know the length
3) Elemento diagonal

4^{11}

2 7 3 8 3 (8)

5 6 2