

* Programação Dinâmica

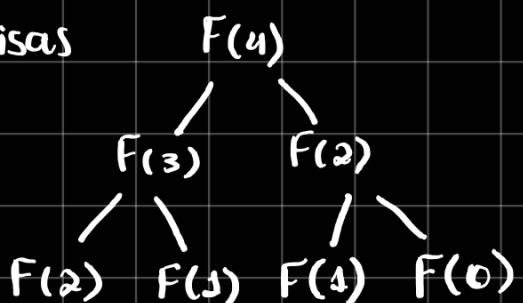
* Fibonacci

$$F_n = \begin{cases} n, & n \leq 1 \\ F_{n-1} + F_{n-2}, & \text{else} \end{cases}$$

```
F(n)
if n ≤ 1
    return n
return F(n-1) + F(n-2)
```

Recálculo coisas

=>

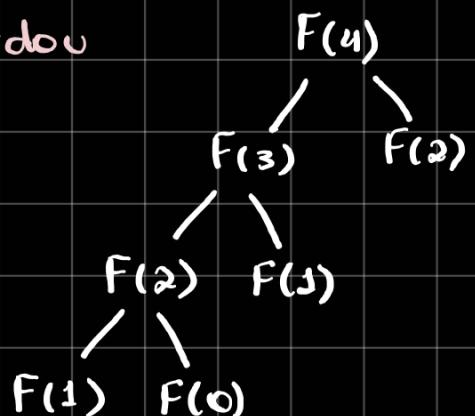


* Top-down / Memoization

```
F(n)
if n ≤ 1
    return n
if f[n] ≠ -1 → verifica se já guardou
    return f[n]
guarda f[n] = F(n-1) + F(n-2)
    return f[n]
```

$f[-1]$	-1	-1	-1	-1	3	5
0	1	2	3	4	5	

Complexidade: $O(n)$



* Bottom-up

$F(n)$

$$f[0] = 0$$

$$f[1] = 1$$

for $i=2$ até n

$$f[i] = f[i-1] + f[i-2]$$

return $f[n]$

Abrindo mão um pouco da complexidade de espaço diminuindo no tempo

Complexidade: $O(n)$

* Problema da Mochila

$$M(n, c) \begin{cases} 0 & n=0 \text{ ou } c=0 \\ M(n-1, c) & \text{se } p[n] > c \\ \max(M(n-1, c-p[n])+v[n], M(n-1, c)) & \text{caso geral} \end{cases}$$



Adiciono na
mochila

Não adiciono na
mochila

* Top down

$M(n, c)$

if $n == 0$ ou $c == 0$
return 0

if $m[n][c] \neq -1$
return $m[n][c]$

if $p[n] > c$
 $m[n][c] = M(n-1, c)$

else

$$m[n][c] = \max(M(n-1, c), M(n-1, c-p[n])+v[n])$$

```
return m[n][c]
```

Complexidade: $O(n \cdot C)$

* Bottom-up

M(n, C)

for i = 0 até C

m[0][i] = 0

for i = 0 até n

m[i][0] = 0

for i = 1 até n

for j = 1 até C

if p[i] > C

m[i][j] = m[i-1][j]

else ↳ não estou usando item

m[i][j] = max(m[i-1][j],

$m[i-1][j - p[i]] + v[i]$)

return m[n][C]

Complexidade: $O(n \cdot C)$

* Seleção de Atividades com peso

Melhor atividade é a com maior peso

SA(i) = maior peso que uma solução que pode utilizar
as atividades $\{1, \dots, i\}$

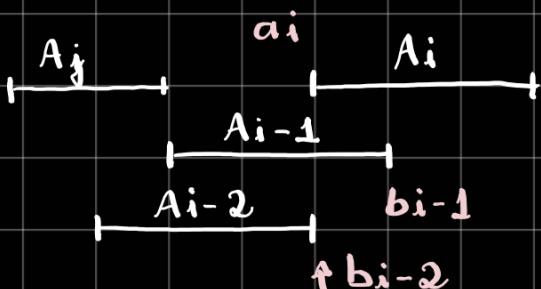
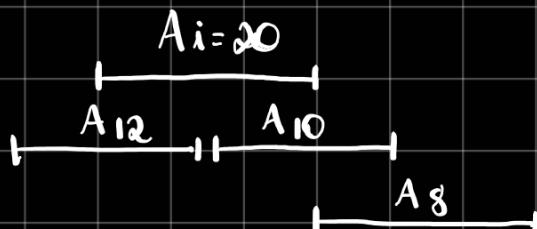
$$SA \leftarrow SA(j) + p_i \text{ (usar } i\text{)}$$

\hookrightarrow maior índice $< i$, que não intersecta

$$SA(i-1) \text{ (não usar } i\text{)}$$

$$SA(i) = \begin{cases} 0, & i=0 \\ \max \left\{ SA(i), p_i + \max_{0 \leq j < i} SA(j) \right\} & t.q. I_j \cap I_i \neq \emptyset \end{cases}$$

maior j t.q. A_j está à esq de A_i



Não pode acontecer $A_{i-2} \cap A_i \neq \emptyset$ e $A_{i-1} \cap A_i = \emptyset$

$i \in S$

$i \notin S$

$$p_i + SA(j)$$

$$SA(i-1)$$

$\hookrightarrow \max j$ t.q. $b_j < a_i$

$SA(sa[], n)$

$sa[0] = 0$

for $i = 1$ até n

$sa[i] = sa[i-1]$

for $k=1$ até $i-1$

if $b_k < a_i$

$j = k$

$O(n)$

$\downarrow O(\lg n)$

com busca

binária

$sa[i] = \max(sa[i], sa[j] + p[i])$

return $sa[n]$

Complexidade: $O(n^2)$

1 2 3 4 .. j j+1 ... i

1 1 1 1 1 0 0 0 0

↑

Como achar o 1 mais a
direita? Busca Binária
(vai eliminando sempre
metade)

Vetor vai informar se está
totalmente a esq do i ou
não. Como está ordenado
por termo no todos a esq
do j é 1.

$SA(sa[], n)$

$sa[0] = 0$

for $i = 1$ até n

$j = i-1$

while $b_j > a$

$j = j-1$

$sa[i] = \max(sa[i-1], p[i] + sa[j])$

return $sa[n]$

Complexidade: $O(n \log n)$