Aula 3
Programação Dinâmica

Introdução

- Programação Dinâmica
  - estratégia de projeto de algoritmos
  - espécie de tradução iterativa inteligente da recursão
  - “recursão com o apoio de uma tabela”

Introdução

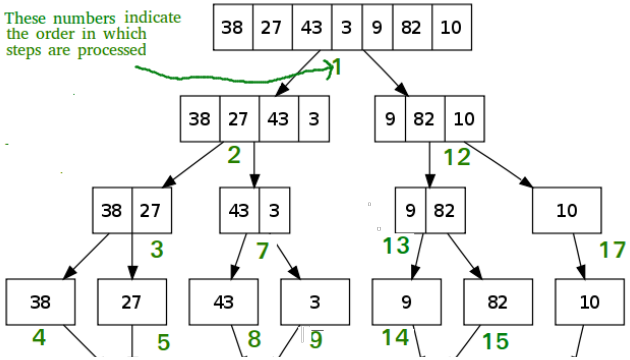
- Programação Dinâmica
  - precisa que o problema tenha estrutura recursiva
    - cada instância do problema é resolvida a partir da solução de instâncias menores
    - tabela que armazena as soluções das várias subinstâncias

Introdução

- Divide and Conquer
  - Recursivo
  - Sem overlapping
- Program. Dinâmica
  - Recursivo
  - Com overlapping

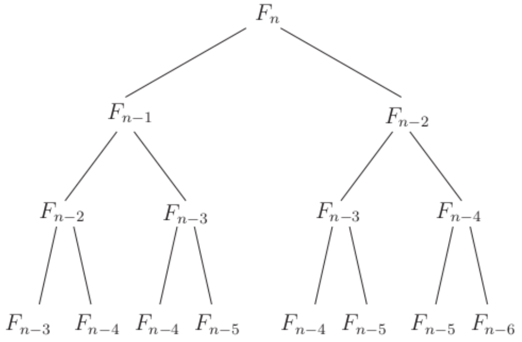
Busca binária

Mergesort



Fibonacci

```
function fib1(n)
if n = 0:  return 0
if n = 1:  return 1
return fib1(n − 1) + fib1(n − 2)
```



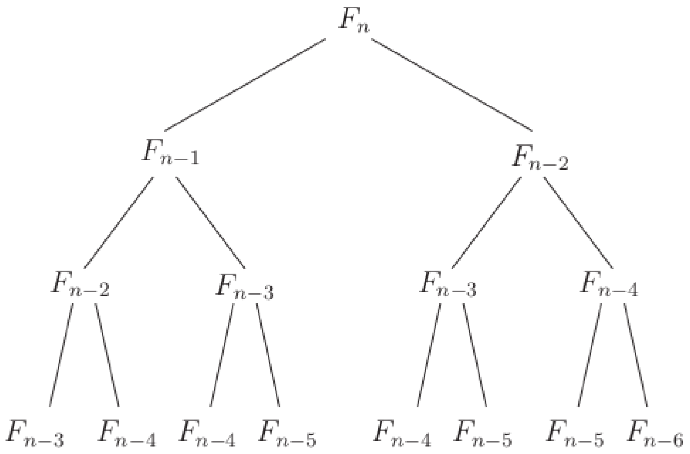
Programação Dinâmica

- Recursão
  - refaz a solução de cada subinstância muitas vezes
- Solução
  - armazenar as solução das subinstâncias numa tabela e assim evitar que elas sejam recalculadas

Exemplo Básico

- Fibonacci

```
function fib1(n)
if n = 0: return 0
if n = 1: return 1
return fib1(n − 1) + fib1(n − 2)
```

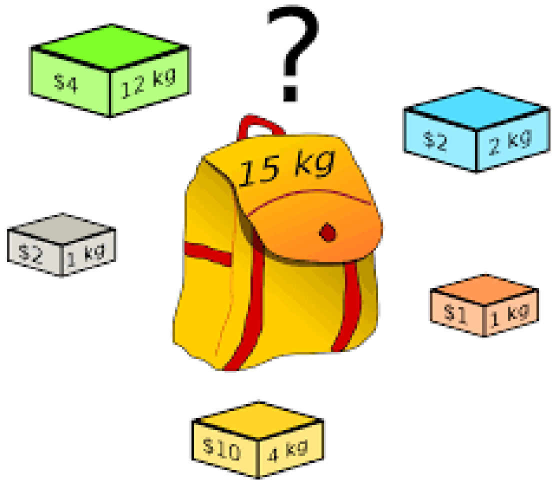



```
function fib2(n)
if n = 0: return 0
create an array f[0...n]
f[0] = 0, f[1] = 1
for i = 2...n:
   f[i] = f[i − 1] + f[i − 2]
return f[n]
```

Armazena soluções das subinstâncias

Mochila Booleana

- Exemplo:
  - Mochila – capacidade 15kg
  - conjunto de objetos, cada um com um certo peso e um certo valor



Quais dos objetos devo colocar na minha mochila para que o valor total seja o maior possível?

Mochila Booleana

- Definição formal

PROBLEMA DA MOCHILA BOOLEANA: Dados vetores $p[1..n]$ e $v[1..n]$ de números naturais e um número natural c, encontrar um subconjunto X do intervalo $1..n$ que maximize $v(X)$ sob a restrição $p(X) \leq c$.

EXEMPLO: Considere a instância do problema da mochila que tem $c = 50$, $n = 5$, e p e v dados na tabela abaixo. O conjunto $\{2, 3\}$ é uma mochila de valor máximo. O valor desta mochila é 1000. Os objetos 2 e 3 esgotam a capacidade da instância, mas isso é um acidente, não uma exigência do problema.

p	40	30	20	10	20
v	840	600	400	100	300
		☑	☑		

Mochila Booleana

- Estrutura recursiva
  - Suponha X é solução da instância (p, v, n, c)
    - Se $n \notin X \rightarrow X$ é solução de $(p, v, n-1, c)$
    - Se $n \in X \rightarrow X - \{n\}$ é solução de $(p, v, n-1, c-p[n])$

Mochila Booleana

- Programação dinâmica
  - guardar em uma tabela as soluções das subinstâncias

t[i, j] é o valor da instância (p, v, i, j)

t[0, j] = 0 para todo j e t[i, 0] = 0 para todo i

Mochila Booleana

- Recorrência

$$t[i,j] = \begin{cases} t[i-1, j] & \text{se } p[i] > j \\ \max\left(t[i-1, j],\, v[i] + t[i-1, j-p[i]]\right) & \text{se } p[i] \leq j \end{cases}$$

- Exemplo:
  - mochila com n = 4 objetos e capacidade c = 5

	1	2	3	4
p	4	2	1	3
v	500	400	300	450

Mochila Booleana

- Recorrência

$$t[i,j] = \begin{cases} t[i-1, j] & \text{se } p[i] > j \\ \max\left(t[i-1, j],\ v[i] + t[i-1, j-p[i]]\right) & \text{se } p[i] \leq j \end{cases}$$

- Exemplo:
  - mochila com n = 4 objetos e capacidade c = 5

	1	2	3	4
p	4	2	1	3
v	500	400	300	450

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	500	500
2	0	0	400	400	500	500
3	0	300	400	400	700	800
4	0	300	400	450	750	850

Mochila Booleana

```
MOCHILA-PROG-DIN (p, v, n, c)
para j := 0 até c
    t[0, j] := 0
    para i := 1 até n
        t[i, j] := t[i−1, j]
        se p_i ≤ j
            t[i, j] := max (t[i, j], v_i + t[i−1, j−p_i])
j := c
para i := n decrescendo até 1
    se t[i, j] = t[i−1, j]
        x[i] := 0
    senão x[i] := 1
            j := j − p_i
devolva x
```

Mochila Booleana

- O consumo de tempo do algoritmo é proporcional ao tamanho da tabela t

$$\Theta(nc)$$