Programação dinâmica

Marcos Castro

Programação dinâmica

- Programação dinâmica é um método para resolver problemas.
- Aplicável em problemas nos quais a solução ótima pode ser obtida a partir da solução ótima previamente calculada e memorizada.
- Essa memorização tem como objetivo evitar o recálculo de subproblemas sobrepostos que compõem o problema original.
- Um problema de otimização deve ter duas características para que a programação dinâmica seja aplicável:
  - Subestrutura ótima
  - Sobreposição de subproblemas

Programação dinâmica

- Subestrutura ótima
  - Isso ocorre quando podemos chegar à solução ótima de um problema através das soluções ótimas de seus subproblemas.

- Sobreposição de subproblemas
  - A sobreposição acontece quando o algoritmo reexamina o mesmo problema várias vezes.

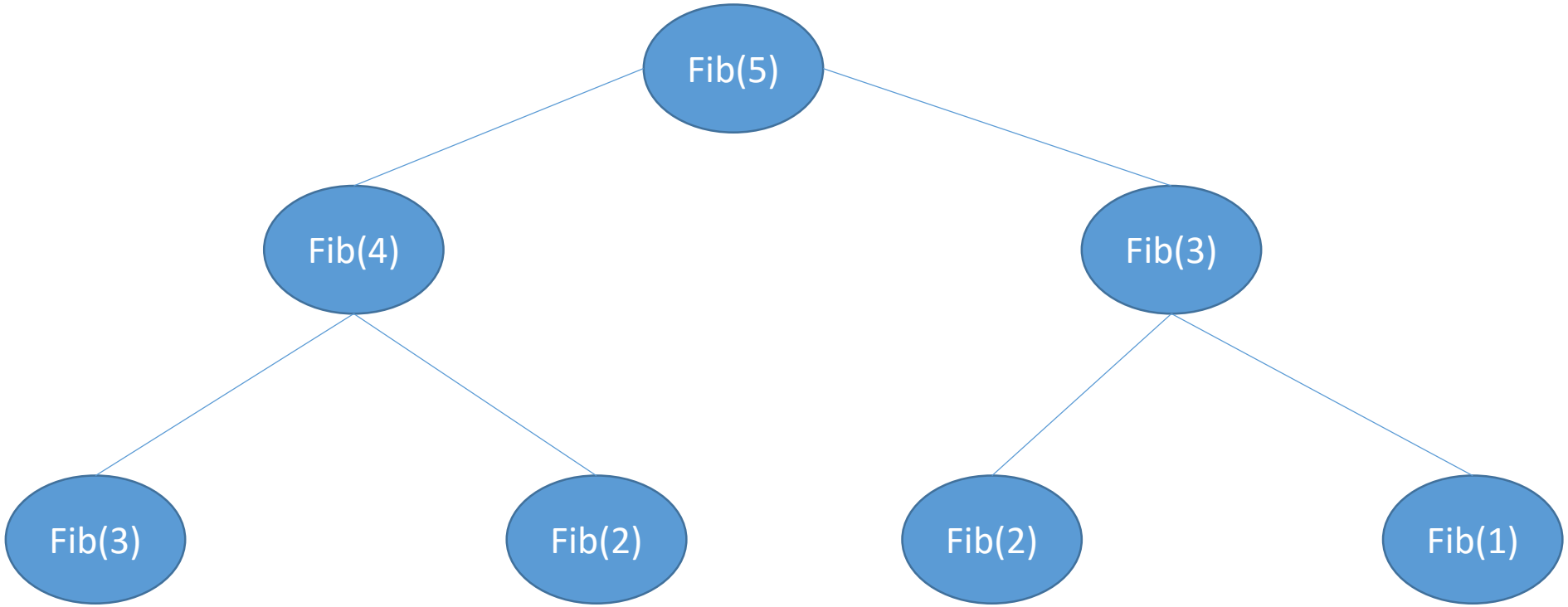
Programação dinâmica

- Um dos exemplos mais simples é a sequência de Fibonacci.
  - Relembrando: a sequência de Fibonacci é uma sequência na qual os dois primeiros termos são iguais a 1 e cada termo que se segue é a soma dos dois termos imediatamente anteriores. Exemplo: 1, 1, 2, 3, 5, 8, 13, 21 ...
- A solução trivial é:

```
1  def fib(n):
2      if n == 1 or n == 2:
3          return 1
4      return fib(n - 1) + fib(n - 2)
5
6  print(fib(7)) # imprime 13
```

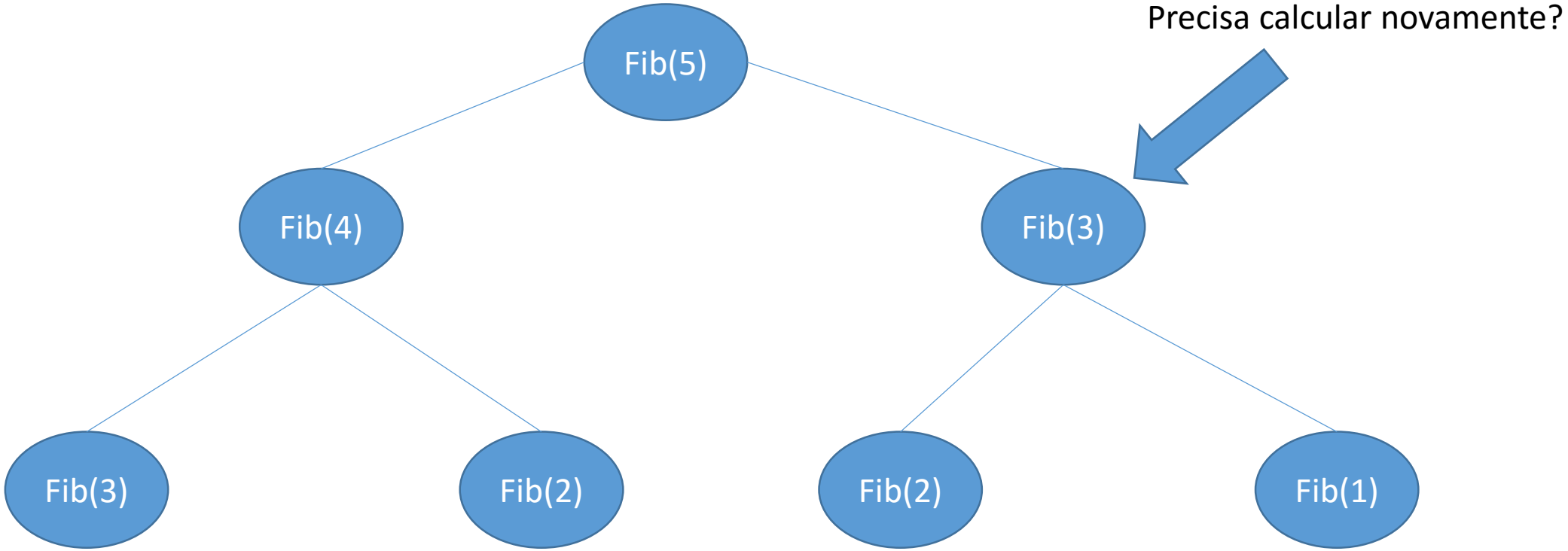
Programação dinâmica

- A solução anterior recalcula subproblemas, veja:



Programação dinâmica

- A solução anterior recalcula subproblemas, veja:



Programação dinâmica

- A solução anterior recalcula subproblemas, por isso é lento!
- Trata-se de uma complexidade exponencial.
- Com programação dinâmica iremos evitar o recálculo e, assim, o programa ficará bem mais otimizado.
- Para isso, basta memorizar os resultados para não recalcular.
- Com o uso de PD, vamos conseguir transformar algo exponencial em linear!

Programação dinâmica

- Solução com PD:

```
def fib_pd(n):

    if mem[n - 1] != -1:
        return mem[n - 1]
    mem[n - 1] = fib_pd(n - 1) + fib_pd(n - 2)
    return mem[n - 1]
```

Programação dinâmica

- Solução com PD:

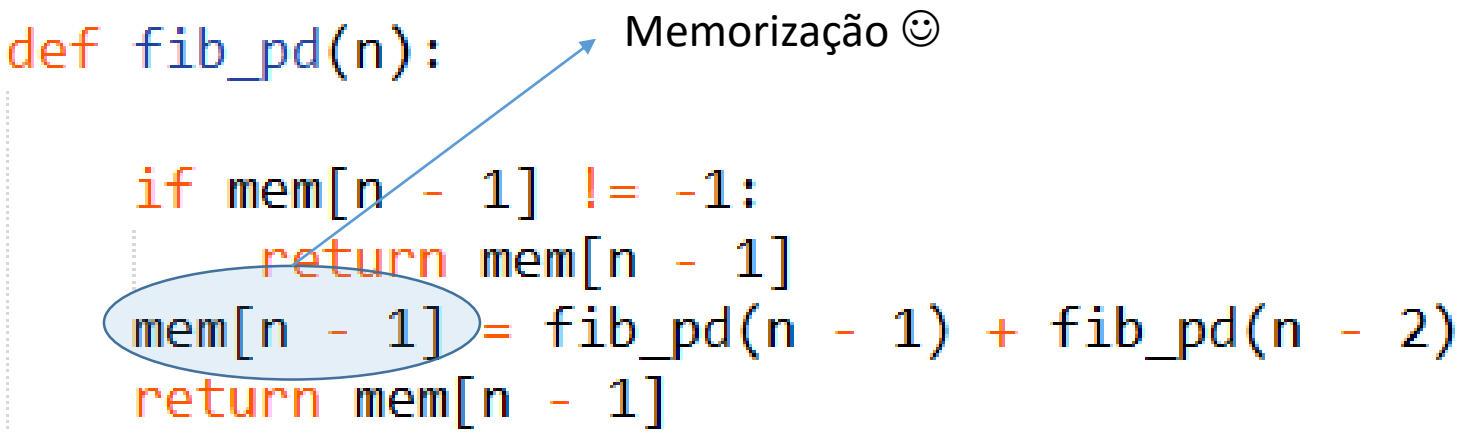
```
def fib_pd(n):

    if mem[n - 1] != -1:
        return mem[n - 1]
    mem[n - 1] = fib_pd(n - 1) + fib_pd(n - 2)
    return mem[n - 1]
```

Memorização ☺

Programação dinâmica

- Solução com PD:



Programação dinâmica

- Com a nossa primeira solução (sem PD), o Fibonacci de 35 demorou cerca de 8.2 segundos.
- Com PD, esse Fibonacci demorou menos de 1 segundo!
- Já o Fibonacci de 40 com a solução sem PD demorou 92.1 segundos.
- O Fibonacci de 40 usando PD demorou menos de 1 segundo!
- Código: https://goo.gl/phLhh3

Contato

mcastrosouza@live.com

www.geeksbr.com

http://youtube.com/c/marcoscastrosouza

https://twitter.com/mcastrosouza

Curso de Algoritmos e Estruturas de Dados com Python 3:

http://ude.my/hn2gg