Programação Dinâmica - Parte 1

Projeto e Análise de Algoritmos II

Antonio Luiz Basile

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie

August 15, 2018

Programação Dinâmica

- Algoritmos baseados em Divisão e Conquista:
  - particionam o problema em subproblemas disjuntos,
  - resolvem recursivamente os subproblemas e
  - combinam as soluções para resolver o problema original.
- Programação Dinâmica se aplica quando:
  - há sobreposição entre os subproblemas, ou seja,
  - quando subproblemas compartilham subsubproblemas.
  - Tipicamente se aplicam a problemas de otimização:
    - ★ Problemas que tem várias soluções possíveis e
    - ★ cada solução tem um valor e
    - ★ devemos encontrar a solução com valor ótimo (mínimo ou máximo).

Programação Dinâmica

Quando desenvolvemos um algoritmo de programação-dinâmica, seguimos uma sequência de 4 passos:

- Caracterize a estrutura de uma solução ótima.
- Recursivamente defina o valor de uma solução ótima.
- Compute o valor de uma solução ótima.
- Construa uma solução ótima a partir da informação computada.

Exemplo 1: Fibonacci

Os números de Fibonacci são assim definidos:

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(N) = F(N-1) + F(N-2) \end{cases}$$

Uma implementação recursiva direta da recorrência que define os números de Fibonacci é espetacularmente ineficiente.

```
int F(int n)
{
  if (n < 1) return 0;
  if (n == 1) return 1;
  else return F(n-1) + F(n-2);
}
```

Fibonacci: entendendo o problema

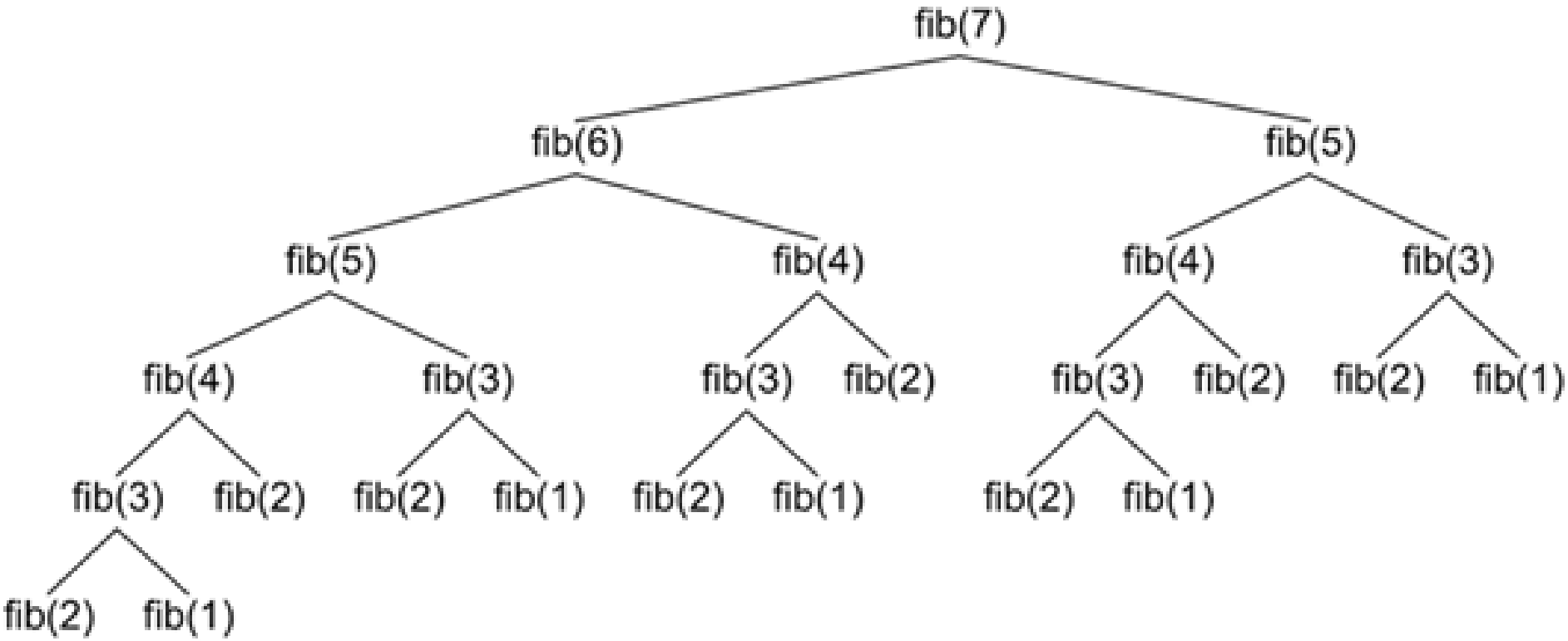

Figure: Árvore do Fibonacci de 7

Fibonacci: entendendo o problema

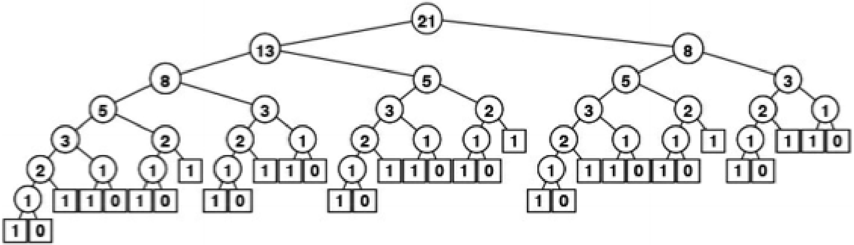

Figure: Árvore calculada do Fibonacci de 8

Fibonacci: entendendo o problema

```
8 F(6)
  5 F(5)
    3 F(4)
      2 F(3)
        1 F(2)
          1 F(1)
          0 F(0)
        1 F(1)
      1 F(2)
        1 F(1)
          0 F(0)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
  3 F(4)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
    1 F(2)
      1 F(1)
        0 F(0)
```

Figure: Chamadas recursivas no Fibonacci

Fibonacci: entendendo o problema

A fórmula fechada aproximada para

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(N) = F(N-1) + F(N-2) \end{cases}$$

é

$$F(N) = O(1,62^N)$$

ou seja, crescimento exponencial!

Será que há um modo de tornar este problema tratável?

Fibonacci: resolvendo o problema via *memoization*

- Para resolver o problema podemos usar uma técnica chamada memoização.
- Memoização é uma forma de programação dinâmica top-down.
- Consiste do seguinte:
  - instrumentamos o programa recursivo para salvar cada valor computado e
  - antes de efetuar uma chamada recursiva, verificamos se seu valor já foi salvo anteriormente.

Fibonacci: resolvendo o problema via *memoization*

Tarefa:

Escreva uma função para o fibonacci que utiliza memoization, ou seja, aplique programação-dinâmica ao fibo.

Fibonacci: resolvendo o problema via *memoization*

```
int F(int n)
{
    int t;

    if (vfib[n] != -1) return vfib[n];
    if (n == 0) t = 0;
    if (n == 1) t = 1;
    if (n > 1) t = F(n-1) + F(n-2);

    return vfib[n] = t;
}
```

Fibonacci: resolvendo o problema via *memoization*

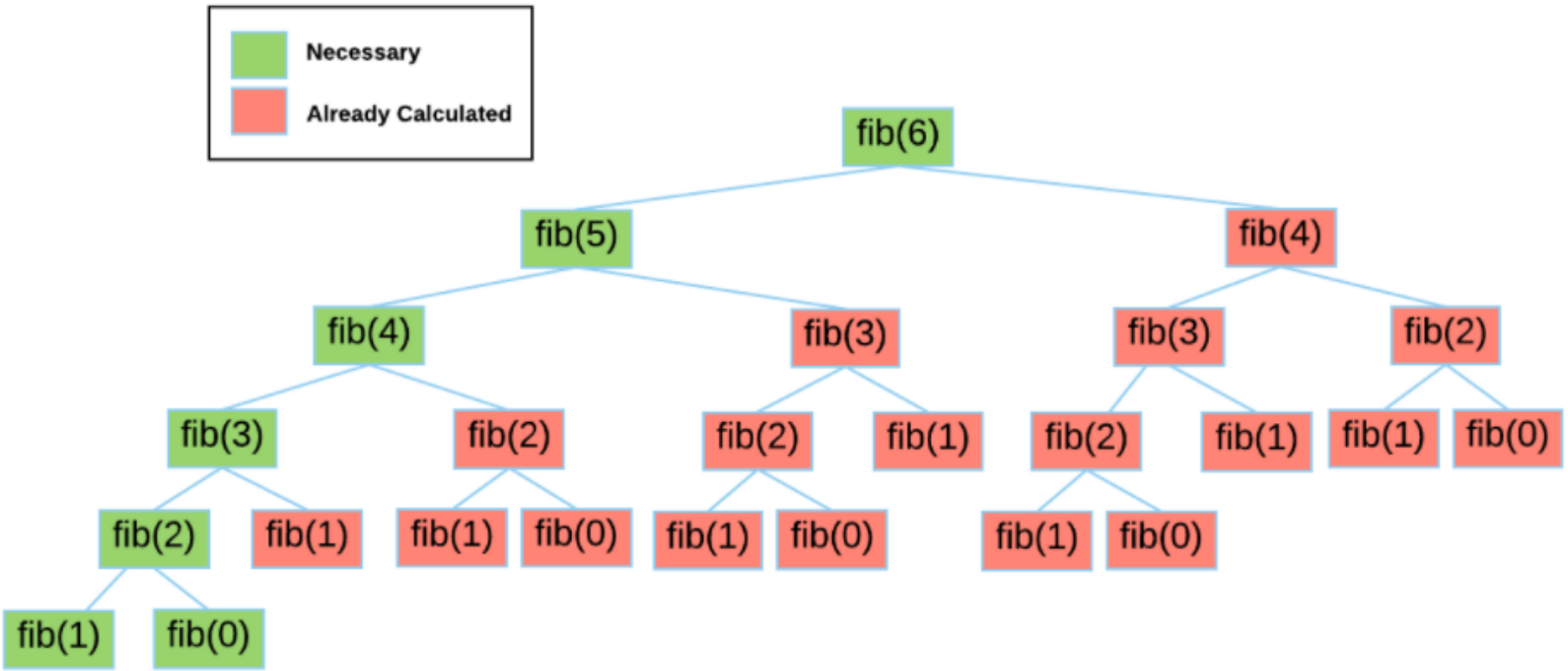

Fig: Fibonacci Number Recursive Implemention

Fibonacci: resolvendo o problema via *memoization*

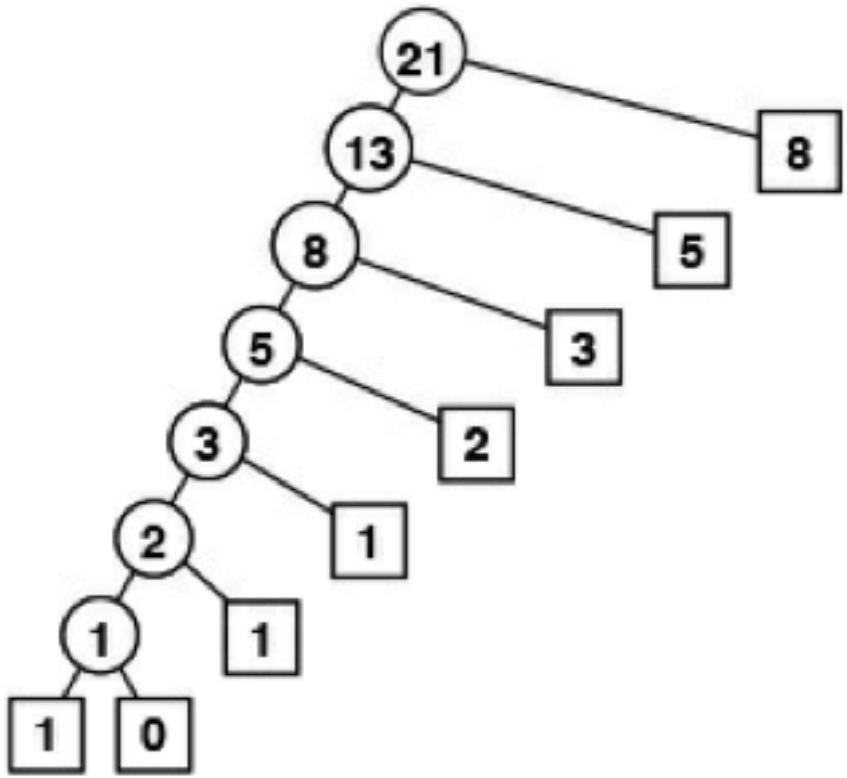

Figure: Fibonacci: memoization