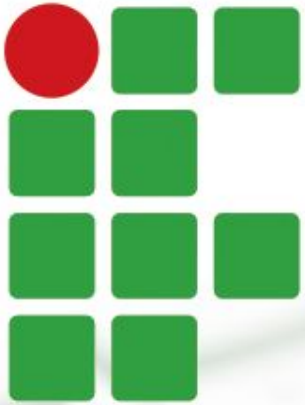


INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados 2
- Programação Dinâmica -

Prof. MSc. Adriano Antunes Prates

Recursão vs. Iteração

- A **sequência Fibonacci**, amplamente encontrada na natureza, é aquela em que o valor do N-ésimo termo da série é a soma dos dois valores anteriores
  - **1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...**
- **Faça em um mesmo código-fonte, duas funções para encontrar o N-ésimo termo da série:**
  - Função Iterativa
  - Função Recursiva
- **Invoque ambas funções para encontrar o 45º termo da série e analise o que acontece...**

Recursão vs. Iteração

n	Iterativo	Recursivo
10	0.17 ms	8 ms
20	0.33 ms	1000 ms
30	0.50 ms	2 min
50	0.75 ms	21 dias
100	1.50 ms	10^9 anos

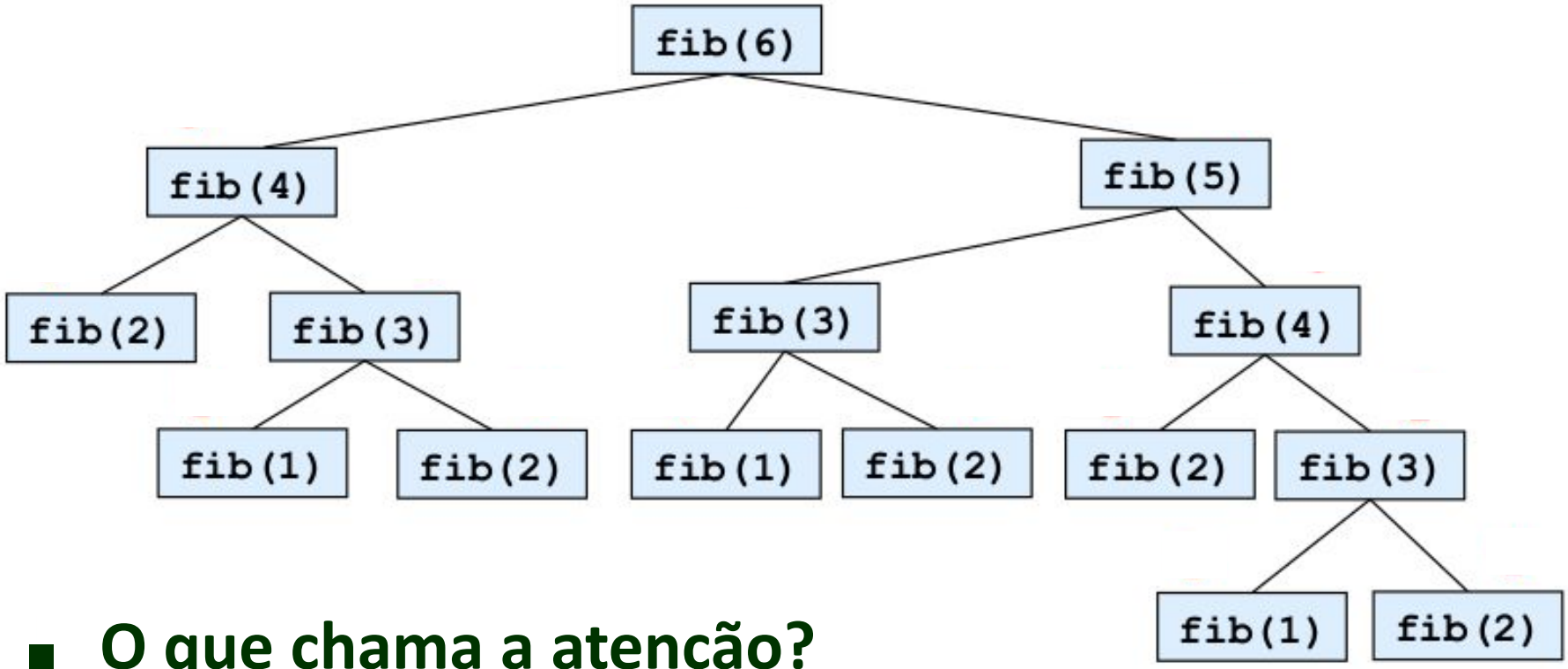
Fonte: Livro Fundamentals of Algorithmics dos autores Gilles Brassard e Paul Bradley.

Recursão vs. Iteração

- Porque a versão **Recursiva** do Algoritmo Fibonacci **é tão menos eficiente** do que a versão **Iterativa**?
- Pelo que estamos estudando, **não deveria ser justamente o contrário?**
- As aparências enganam…
- Por isso a importância de se fazer uma boa **Análise de Algoritmos**.

Recursão vs. Iteração

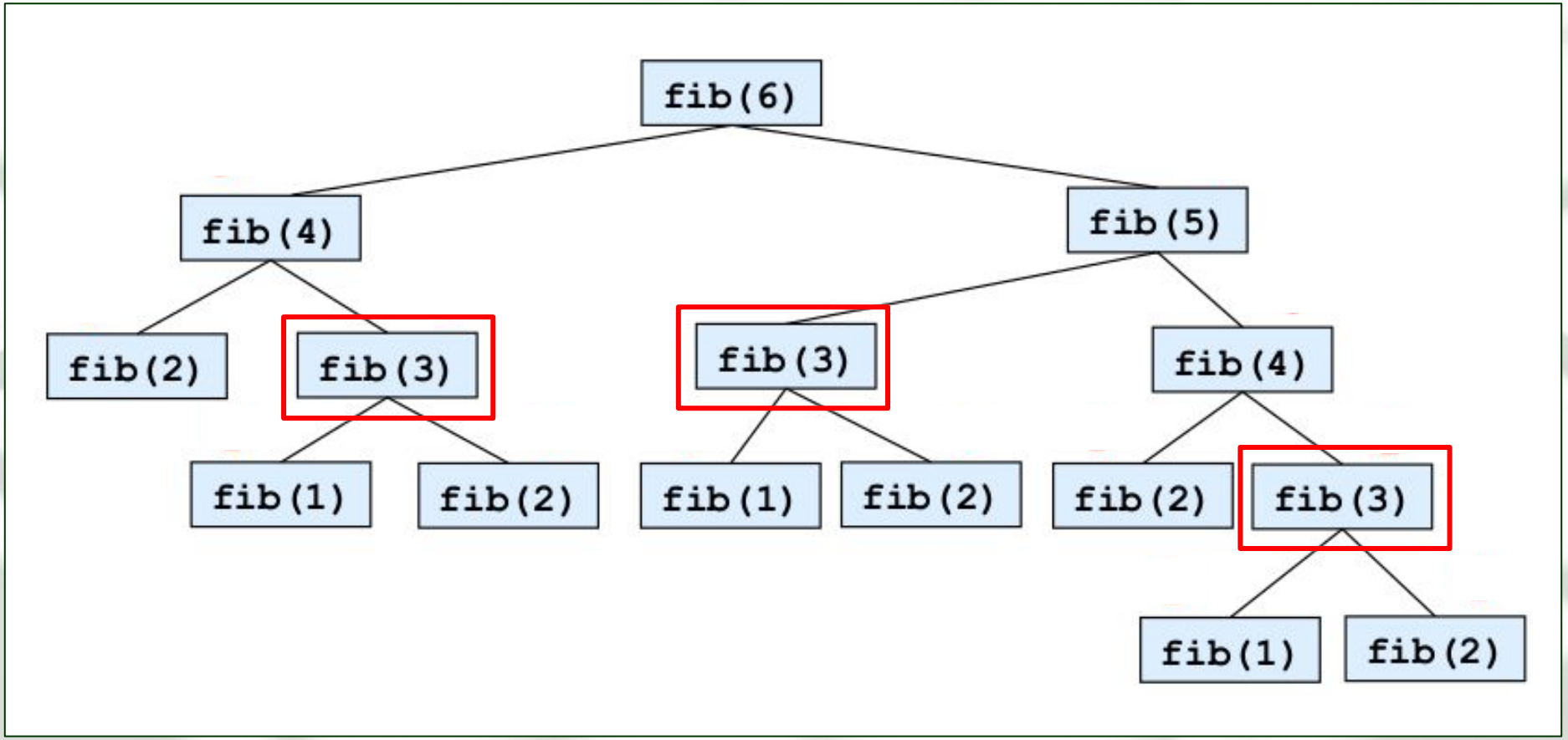
- Vamos analisar a árvore de execução…



- **O que chama a atenção?**

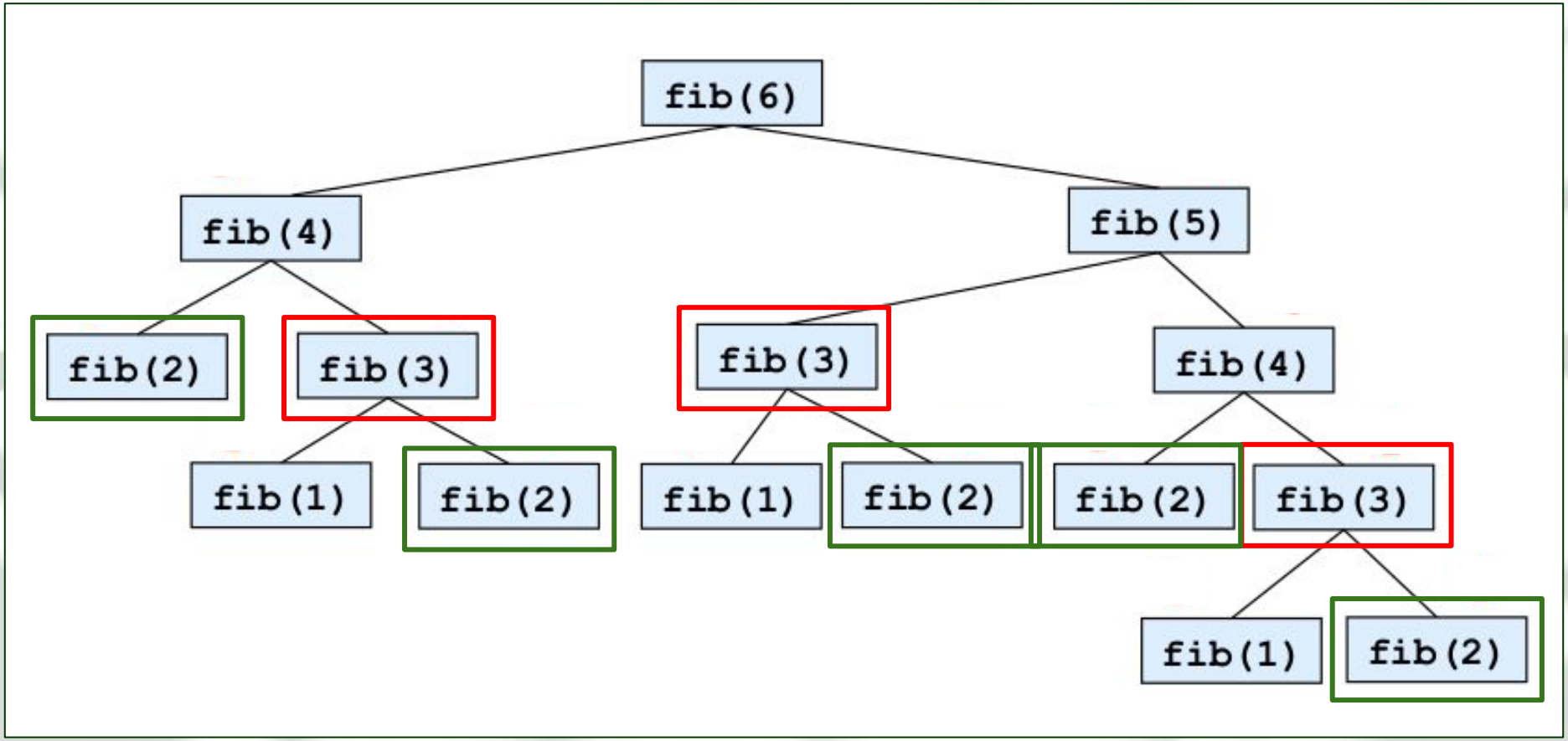
Recursão vs. Iteração

- Vamos analisar a árvore de execução…



Recursão vs. Iteração

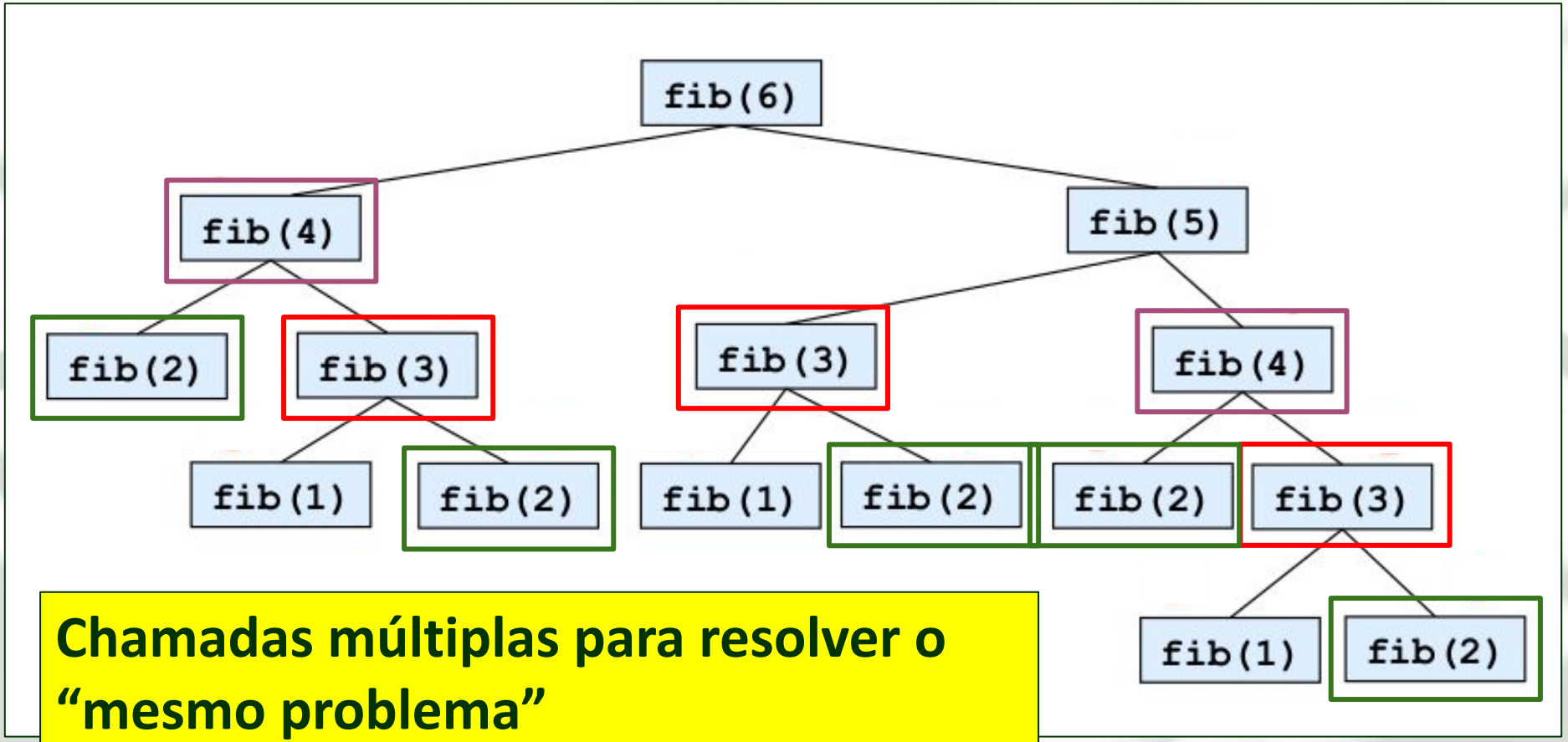
- Vamos analisar a árvore de execução…



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Recursão vs. Iteração

- Vamos analisar a árvore de execução…

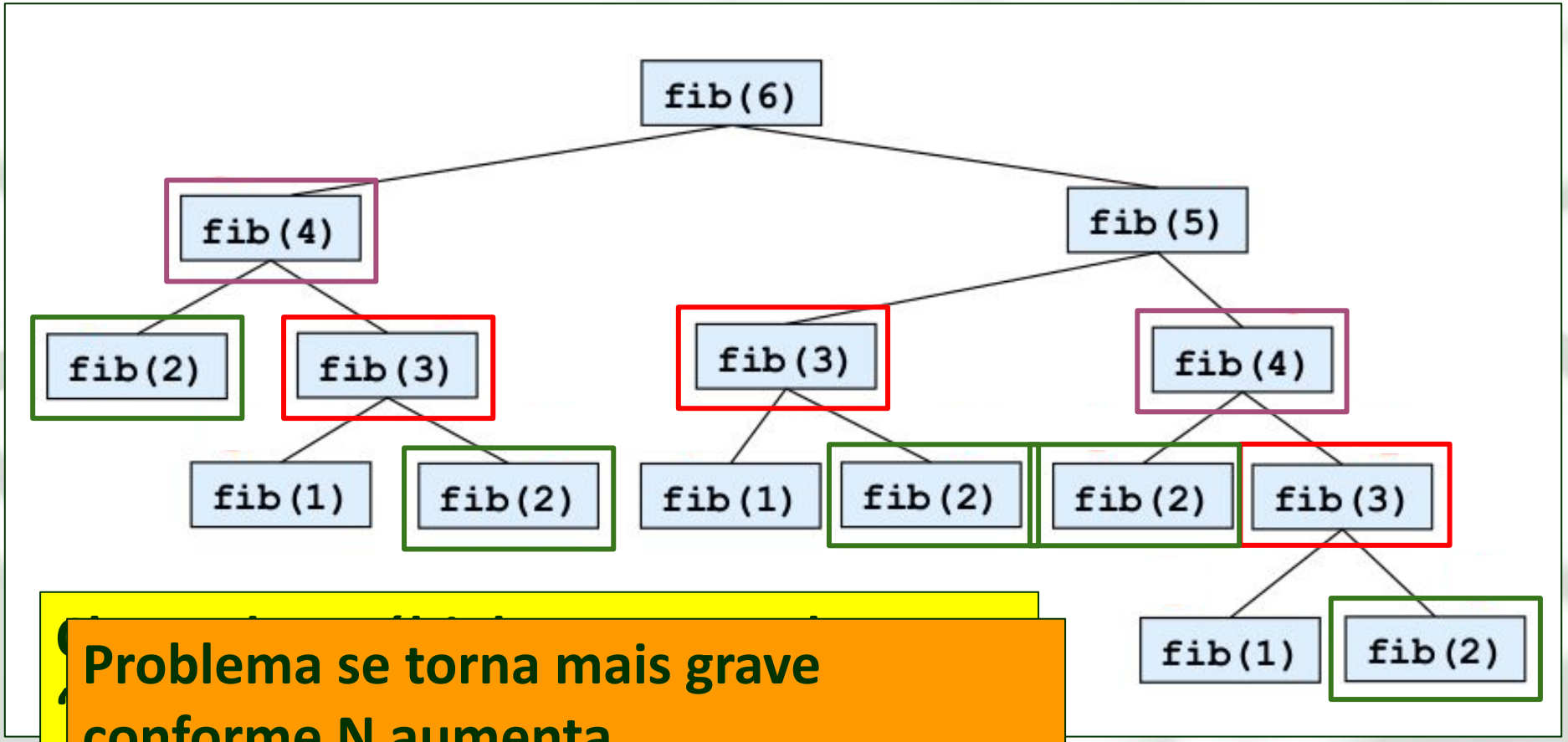


Chamadas múltiplas para resolver o "mesmo problema"

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Recursão vs. Iteração

- Vamos analisar a árvore de execução…



Problema se torna mais grave conforme N aumenta…

Recursão vs. Iteração

- Um algoritmo recursivo **nem sempre** é o melhor caminho para se resolver um problema.
- Na maioria das vezes, no entanto, a recursividade torna o algoritmo mais simples (vide os exemplos anteriores do “Bora Codar”).
- Entretanto, existe uma solução que une a vantagem dos “dois mundos”: a técnica de **Programação Dinâmica**.

Programação Dinâmica

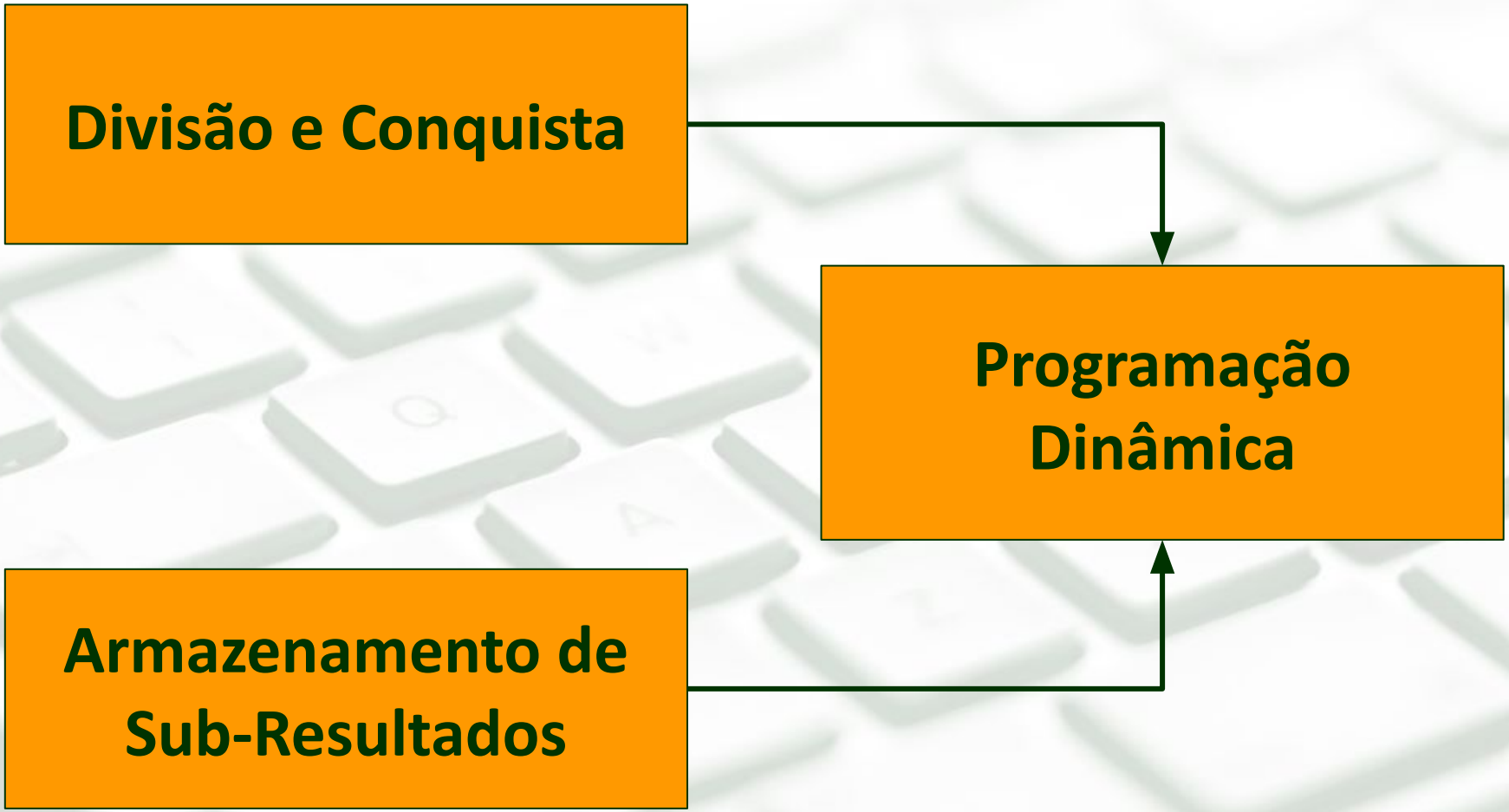
- A **Programação Dinâmica** é uma estratégia de desenvolvimento que busca encontrar a solução de vários sub-problemas para, daí então, encontrar a solução do problema geral.

- A grande novidade dessa metodologia **é que os sub-resultados vão sendo armazenados em memória,** pois poderão ser utilizados em outros momentos oportunos dentro do cômputo da solução.

Programação Dinâmica

- Pode-se imaginar portanto, que esta estratégia busca a implementação de "**recursividade com apoio de uma tabela de sub-resultados**".
- Na prática, antes de tentar resolver uma sub-instância do problema, verifica-se se já ela foi resolvida anteriormente.
- **Muitos algoritmos eficientes são baseados no método de Programação Dinâmica.**

Programação Dinâmica



Bora Codar!

- **Desenvolva a solução do problema do n-ésimo termo da série Fibonacci usando as técnicas de Recursividade + Programação Dinâmica.**

Seminários

- **Seminários de Análise de Algoritmos**
  - Todos os algoritmos em suas versões otimizadas.
  - Análise do código e Demonstração prática.
  - Explicar a Ordem de Complexidade de cada Algoritmo.

1. Ordenação: Radix Sort
2. Ordenação: Quick Sort Recursivo
3. Ordenação: Merge Sort
4. Ordenação: Heap Sort
5. Ordenação: Shell Sort
6. Algoritmo do Segmento de Soma Máxima
7. Algoritmo da Máxima Subsequência Crescente
8. Otimização: Problema da Mochila
9. Percurso mínimo: Algoritmo de Dijkstra