

# Aula 13.1 – Funções Recursivas

Prof. Me. Hugo Régis

# Apresentação

- Introdução
  - Algoritmos Recursivos vs Iterativos
- Sequência de Fibonacci - Solução Recursiva vs Iterativa
- Comparação de tempos de execução (C e Python)

# Algoritmos Recursivos vs Iterativos

- Algoritmos Recursivos **top-down**
  - Do **n** para o **caso base**
- Algoritmos Iterativos **bottom-up**
  - Do **caso base** para **n**

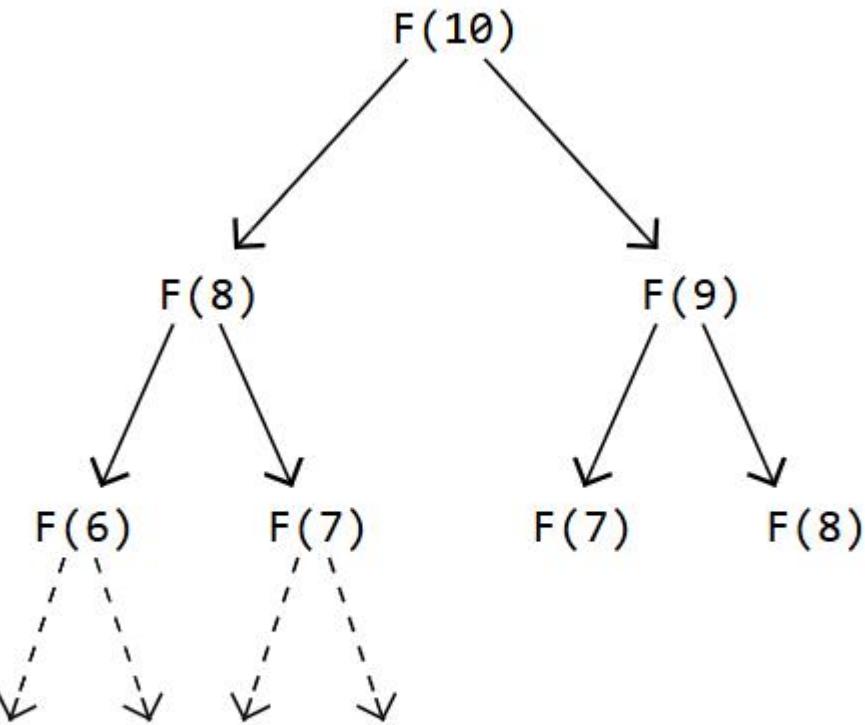
# Sequência de Fibonacci - Recursiva

Calcular o Fibonacci de um número n

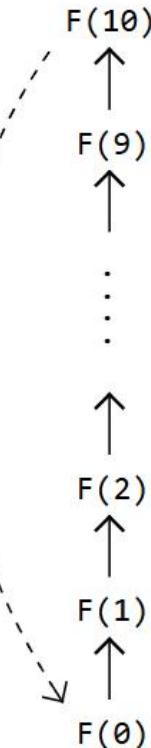
$$F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n - 1) + F(n - 2) & n > 1 \end{cases}$$

n	0	1	2	3	4	5	6	7
F(n)	0	1	1	2	3	5	8	13

# Algoritmos Recursivos vs Iterativos



*The top-down memoization approach to finding the 10th Fibonacci number.*



*The bottom-up tabulation approach to finding the 10th Fibonacci number.*

# Sequência de Fibonacci - Iterativa

Implementação em C



# Sequência de Fibonacci - Iterativa

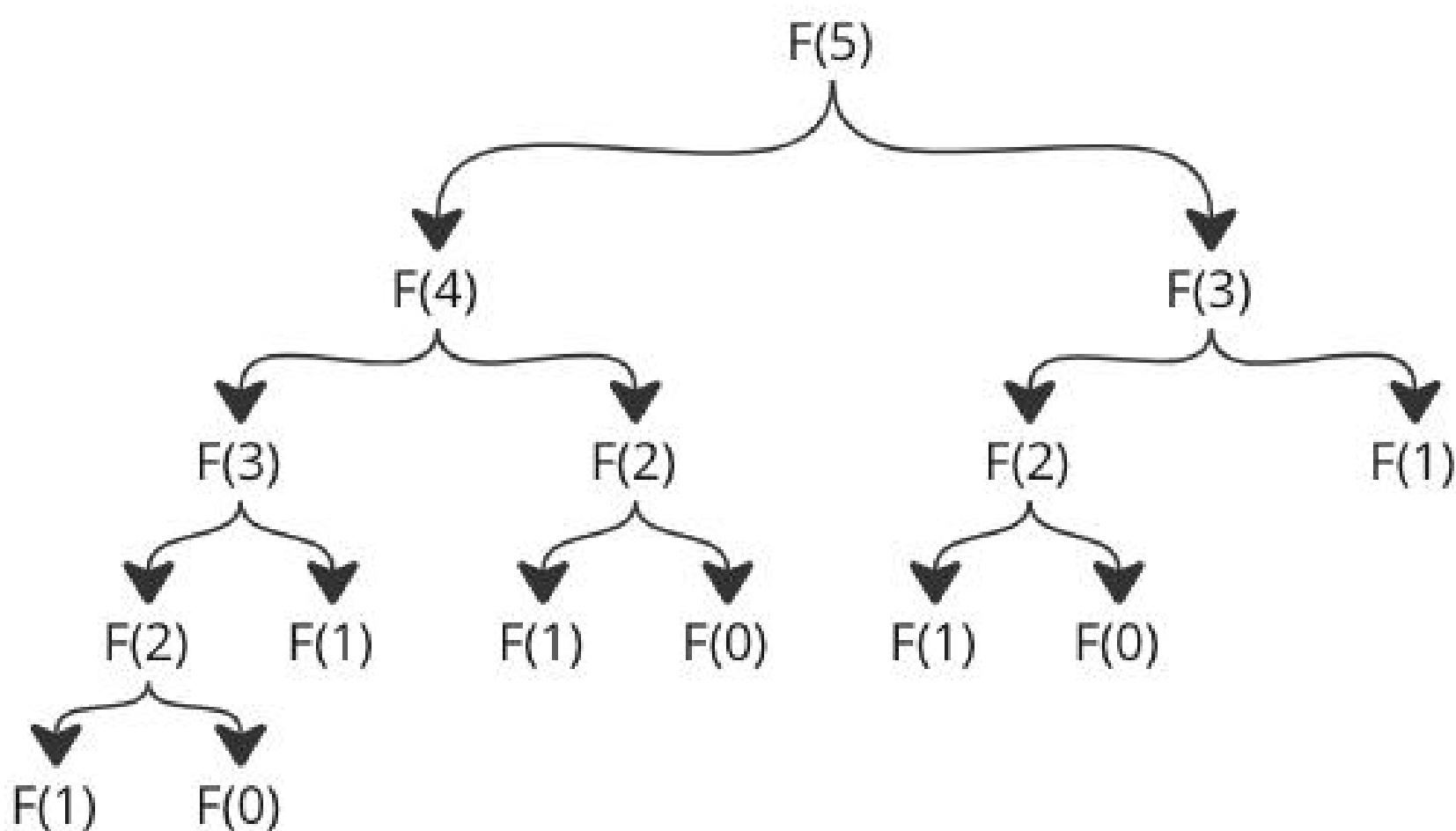
```
#include <stdio.h>

int fibonacci(int n) {
    if (n <= 1)
        return n;

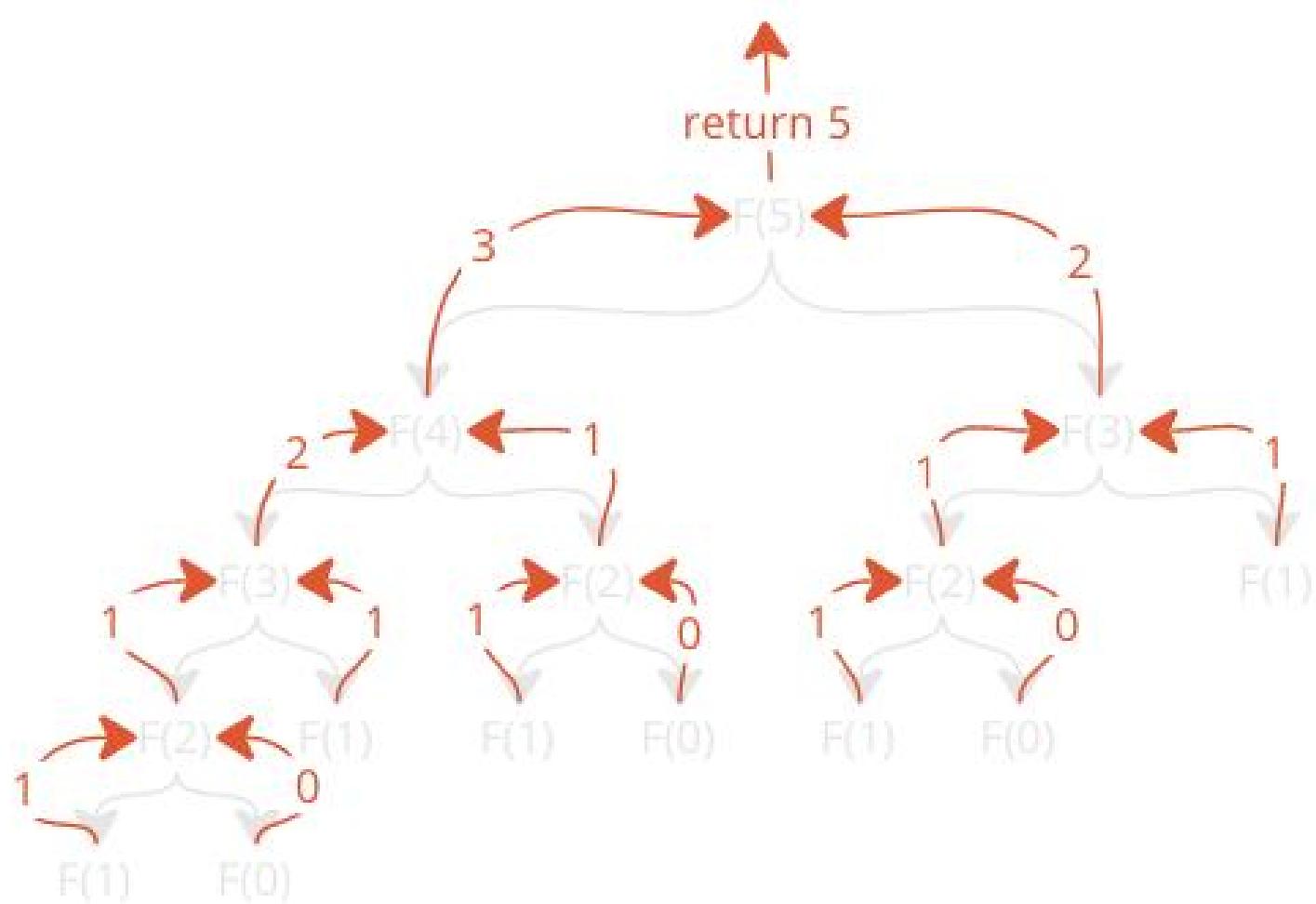
    int anterior = 0, atual = 1, proximo;
    for (int i = 2; i <= n; i++) {
        proximo = anterior + atual;
        anterior = atual;
        atual = proximo;
    }
    return atual;
}

int main() {
    int n;
    //printf("Digite o valor de n: ");
    //scanf("%d", &n);
    n = 40;
    printf("Fibonacci(%d) = %d\n", n, fibonacci(n));
    return 0;
}
```

# Sequência de Fibonacci - Recursiva



# Sequência de Fibonacci - Recursiva



# Sequência de Fibonacci - Recursiva

Implementação em C



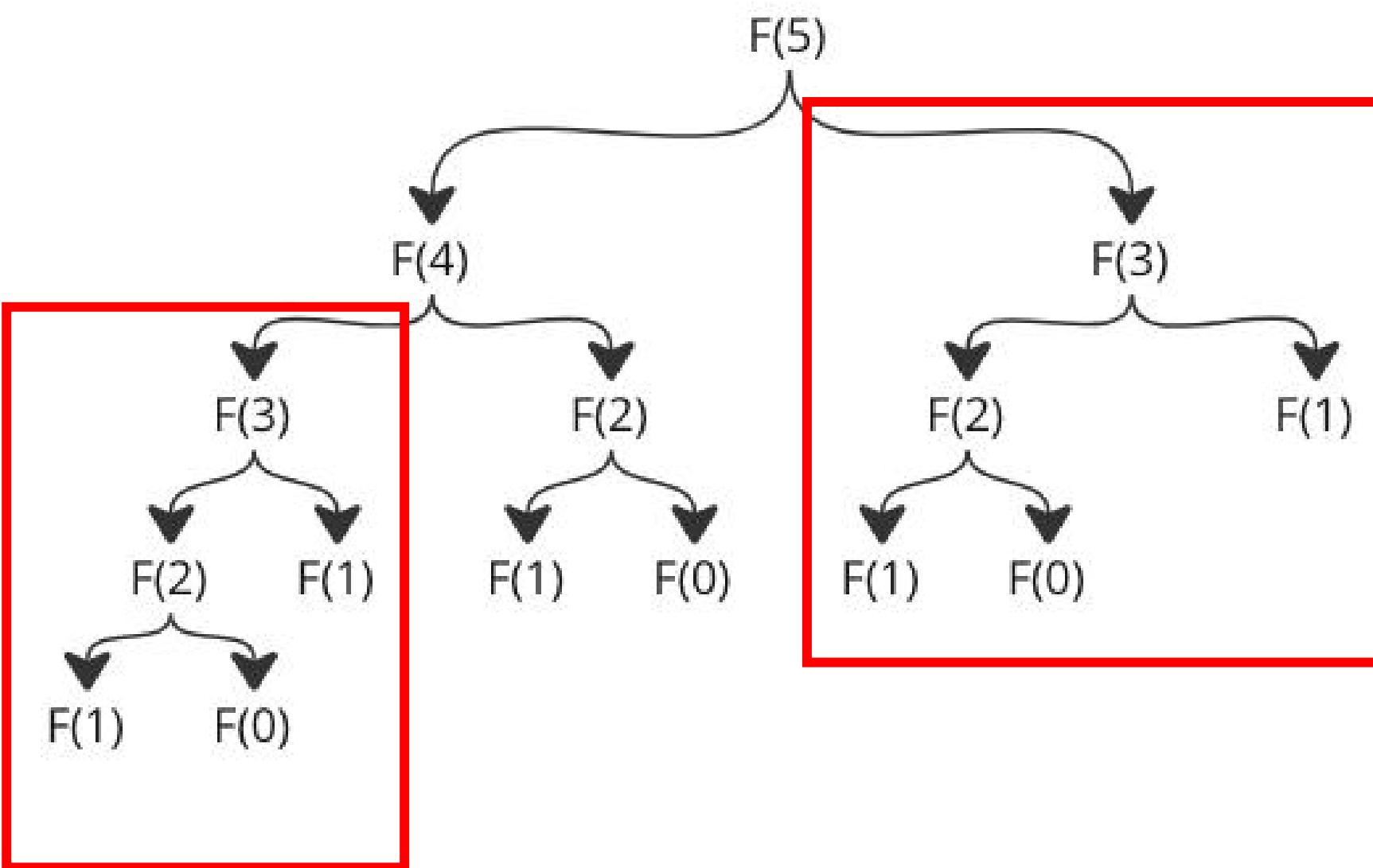
# Sequência de Fibonacci - Recursiva

```
#include <stdio.h>

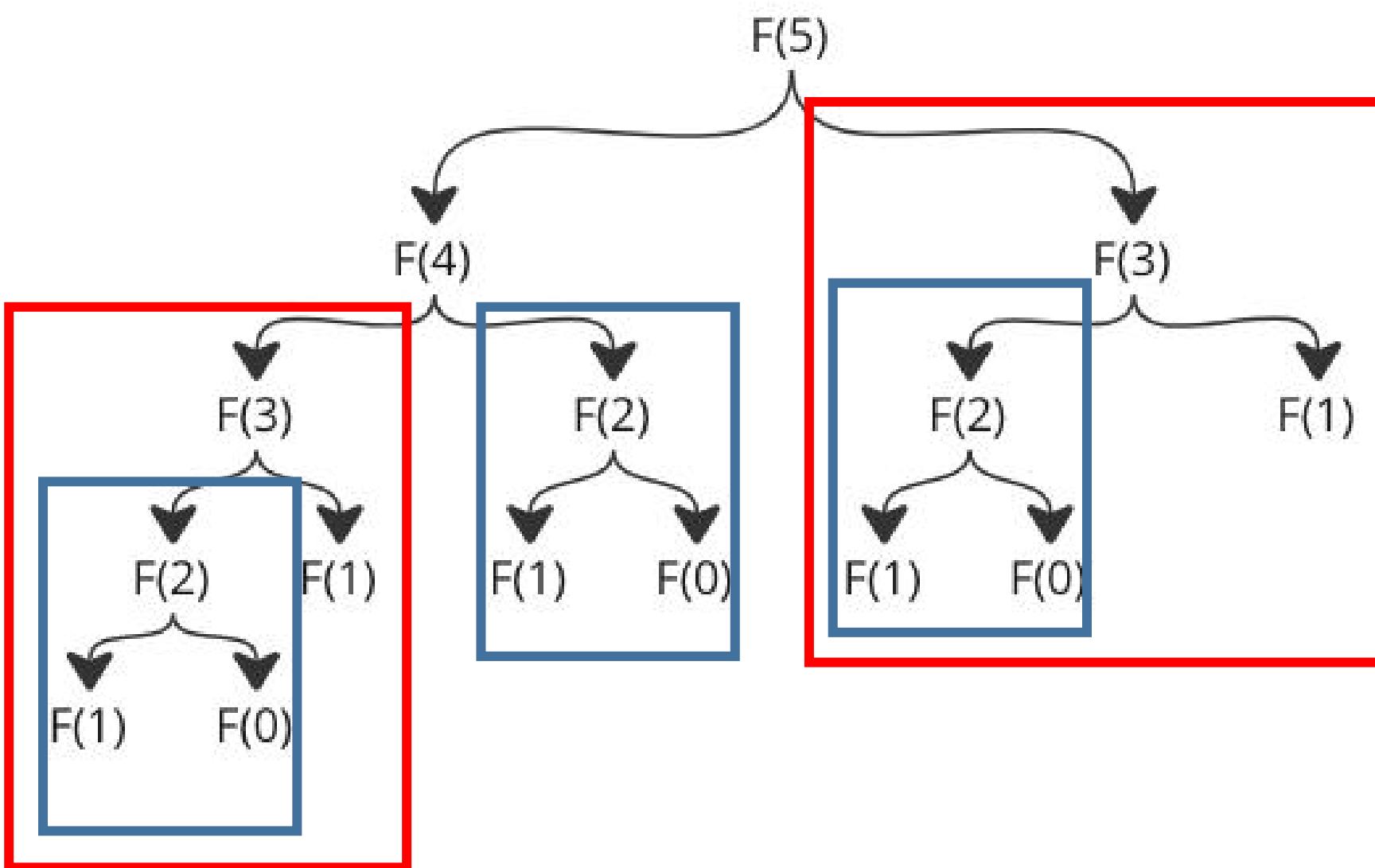
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n;
    //printf("Digite o valor de n: ");
    //scanf("%d", &n);
    n = 40;
    printf("Fibonacci(%d) = %d\n", n, fibonacci(n));
    return 0;
}
```

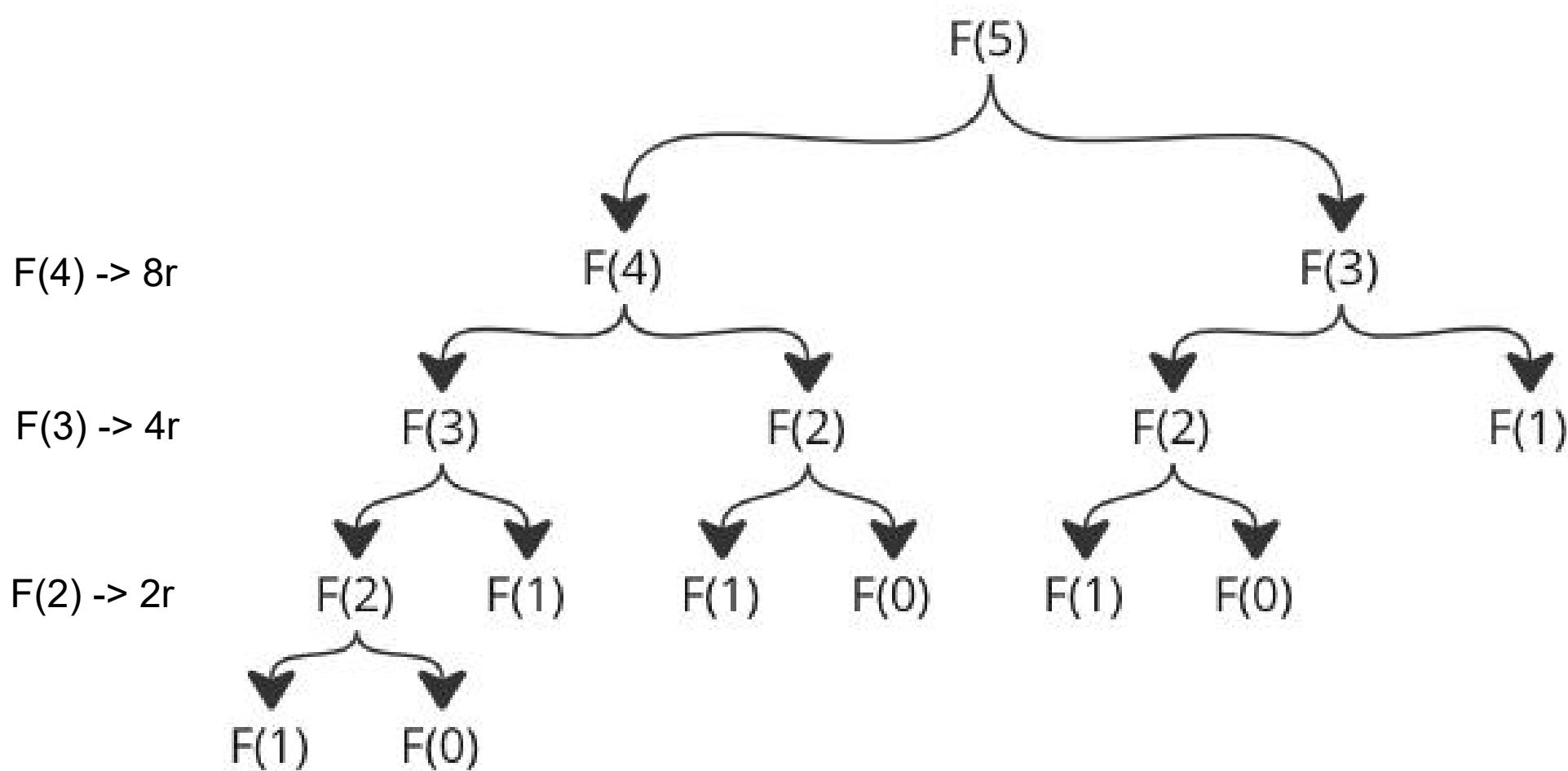
# Sequência de Fibonacci - Recursiva



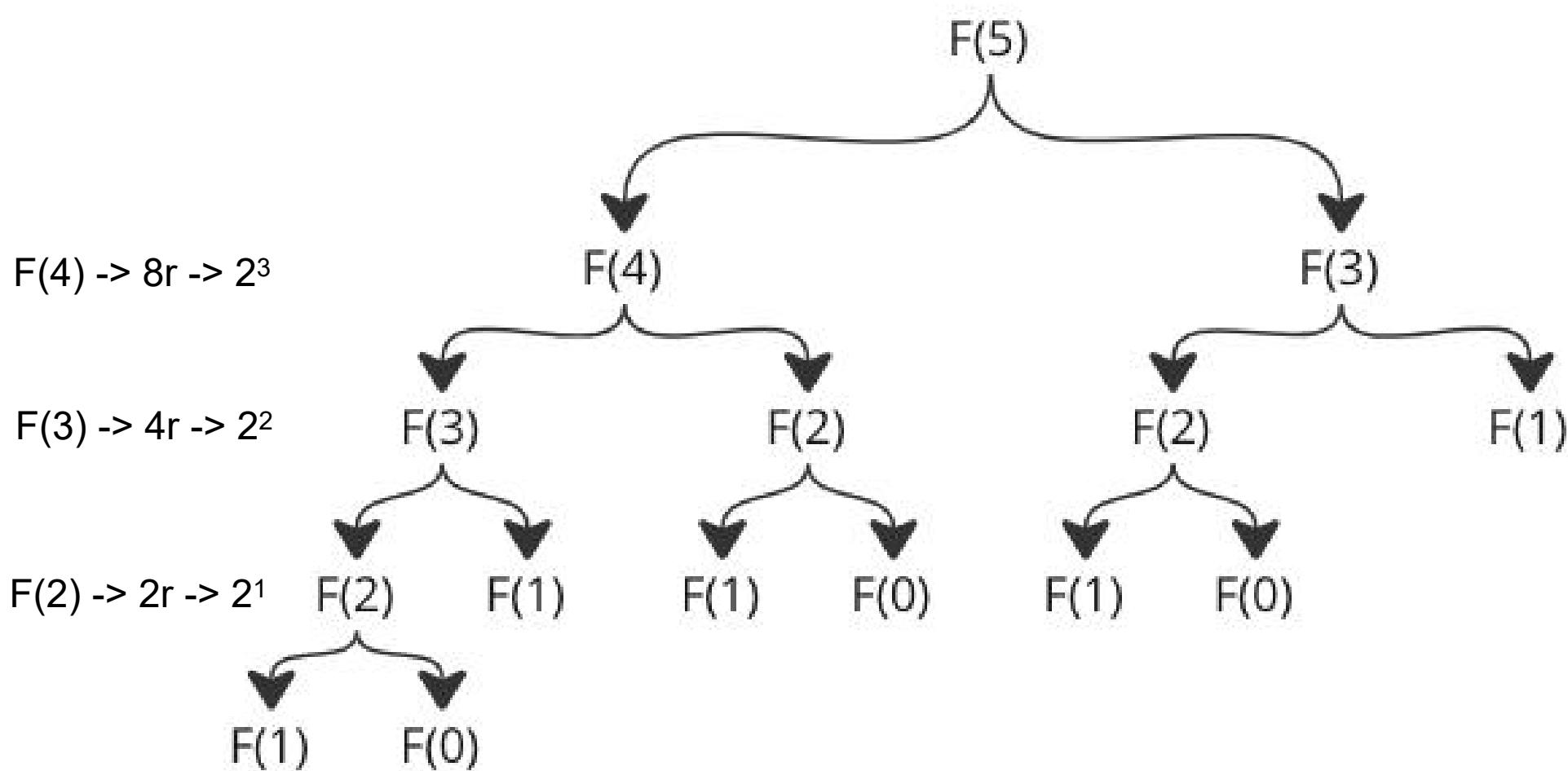
# Sequência de Fibonacci - Recursiva



# Sequência de Fibonacci - Recursiva



# Sequência de Fibonacci - Recursiva



# Sequência de Fibonacci - Recursiva

$$F(n) \rightarrow 2^{n-1} \quad O(2^n)$$

1

$$F(4) \rightarrow 8r \rightarrow 2^3$$

F(5)

F(4)

F(3)

$$F(3) \rightarrow 4r \rightarrow 2^2$$

F(3)

F(2)

F(2)

F/1

F(2) -> 2r -> 2<sup>1</sup>

F/2

F11

F11

F/0

F11

F(0)

F11

FON

# Conclusão

- Algoritmos recursivos: são mais simples e intuitivos de implementar de acordo com a natureza do problema, porém, sem o tratamento adequado, efetuam cálculos repetidos (da árvore de recursão) e utilizam mais memória (podendo gerar stack overflow).
- Algoritmos iterativos: adicionam uma certa complexidade na resolução do problema, porém, tendem a executar mais rapidamente do que os recursivos.

# Sequência de Fibonacci – Comparaçõa no tempo de execução

time C e Python

\$-

# Exercício

Escreva 2 algoritmos, um recursivo e outro iterativo, que calcule o fatorial de um número n.

Sabendo que

$$\text{Ex: } 5! = 1 * 2 * 3 * 4 * 5 = 120$$

$$5! = 5 * 4! = 120$$

# Exercício - Solução

03\_fat\_recursivo > C app.c > ...

```
1 #include <stdio.h>
• 2
3 int fatorial(int n) {
4     if (n <= 1)
5         return 1;
6     else
7         return n * fatorial(n - 1);
8 }
9
10 int main() {
11     int numero;
12
13     printf("Digite um numero inteiro: ");
14     scanf("%d", &numero);
15     printf("Fatorial de %d (recursivo) = %d\n", numero, fatorial
16           (numero));
17
18     return 0;
19 }
```

# Exercício - Solução

02\_fat\_iterativo > C app.c > ...

```
1 #include <stdio.h>
2
3 int fatorial(int n) {
4     int resultado = 1;
5     for (int i = 2; i <= n; i++) {
6         resultado *= i;
7     }
8     return resultado;
9 }
10
11 int main() {
12     int numero;
13
14     printf("Digite um numero inteiro: ");
15     scanf("%d", &numero);
16     printf("Fatorial de %d (iterativo) = %d\n", numero, fatorial
17     (numero));
18
19 }
```

# Aula 13.1 – Funções Recursivas

Prof. Me. Hugo Régis