

АиСД лекция 5

Рекурсия



Что такое рекурсия?



Что такое рекурсия?

Рекурсия - это возможность функции вызывать саму себя.

Рекурсия включает в себя

- Базовый случай
- Переход



Ряд Фибоначчи

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

Рекурсивное представление

$$fib_n = fib_{n-1} + fib_{n-2}$$

$$fib_0 = 0$$

$$fib_1 = 1$$



Что такое базовый случай ?

```
def fib(n: int) -> int:
    if n <= 1:                # Базовый случай
        return 1
    return fib(n-1) + fib(n-2) # Переход
```



Что такое зацикливание ?

Когда рекурсия начинает бесконечно вызывать саму себя

```
def fib(n: int) -> int:  
    return fib(n-1) + fib(n-2) + fib(n)
```



Что такое зацикливание ?

Или когда вы плохо учили базовый случай

```
def fib(n: int) -> int:  
    if n == 0 or n == 1:  
        return 1  
    return fib(n-1) + fib(n-2)  
fib(-1)
```



Рассмотрим код

```
def fib(n: int) -> int:  
    if n <= 0:  
        return 0  
    if n == 1:  
        return 1  
    return fib(n-1) + fib(n-2)
```

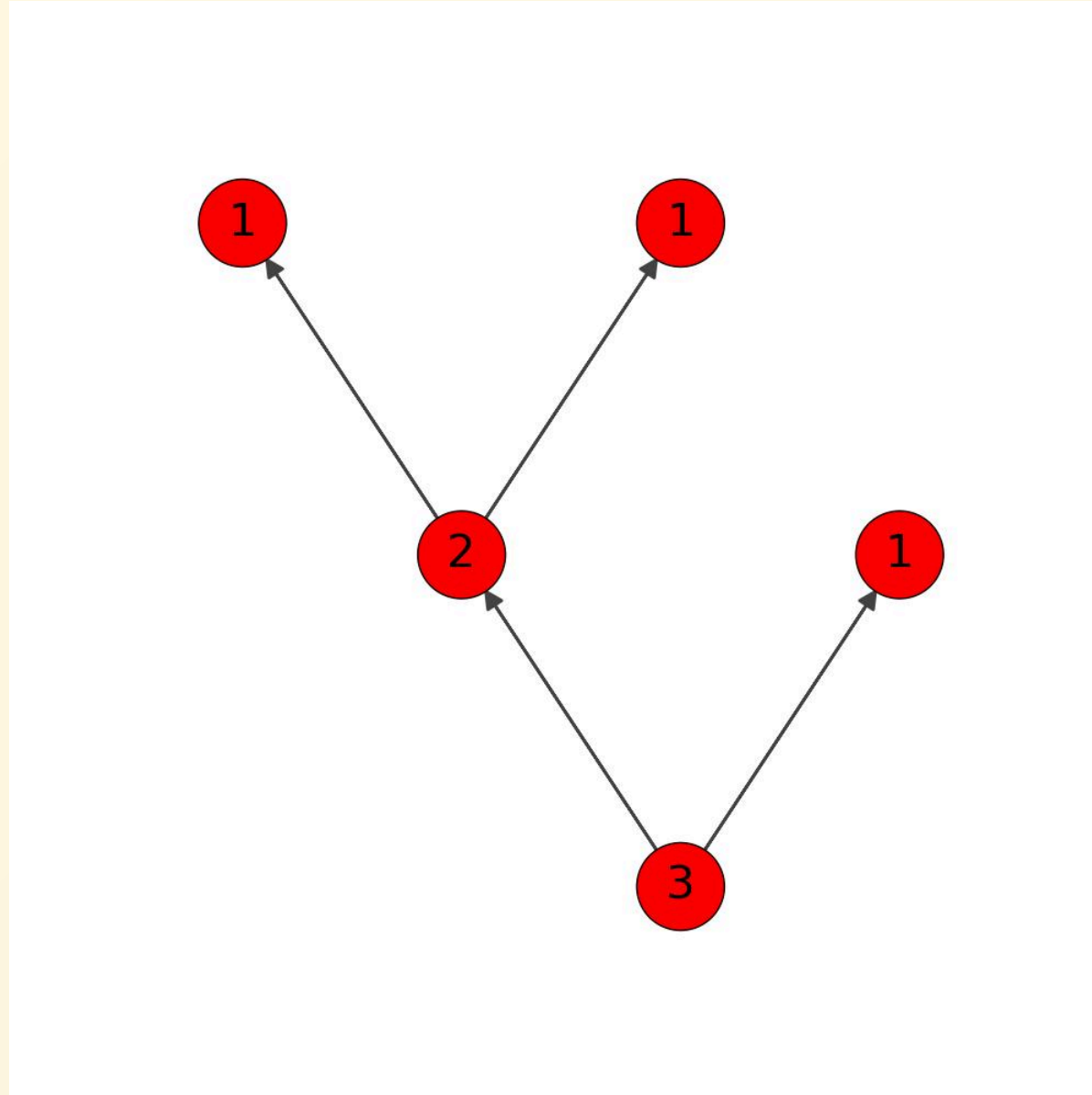

Какие рекурсивные вызовы будут при вызове?

`fib(4)` # Сколько всего будет вызовов `fib`?

$$fib(4) \rightarrow fib(3) + fib(2)$$

$$fib(3) \rightarrow fib(2) + fib(1)$$

$$fib(2) \rightarrow fib(1) + fib(0)$$



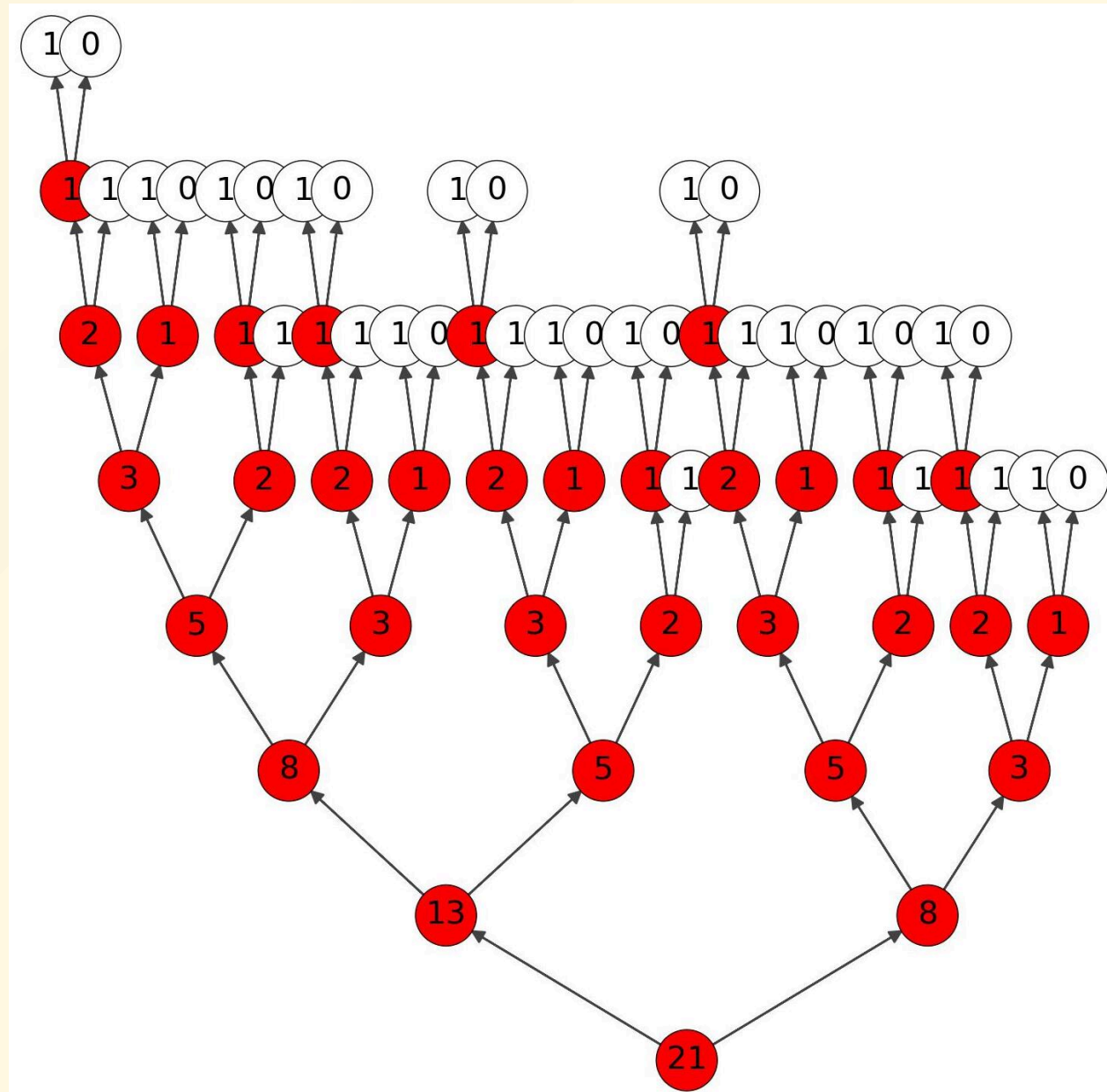


Давайте подумаем

Сколько вызовов fib, будет если передать аргументом число 8?

Какое 8-е число Фибоначчи?

- 67 вызовов функции fib
- восьмое число Фибоначчи это 21

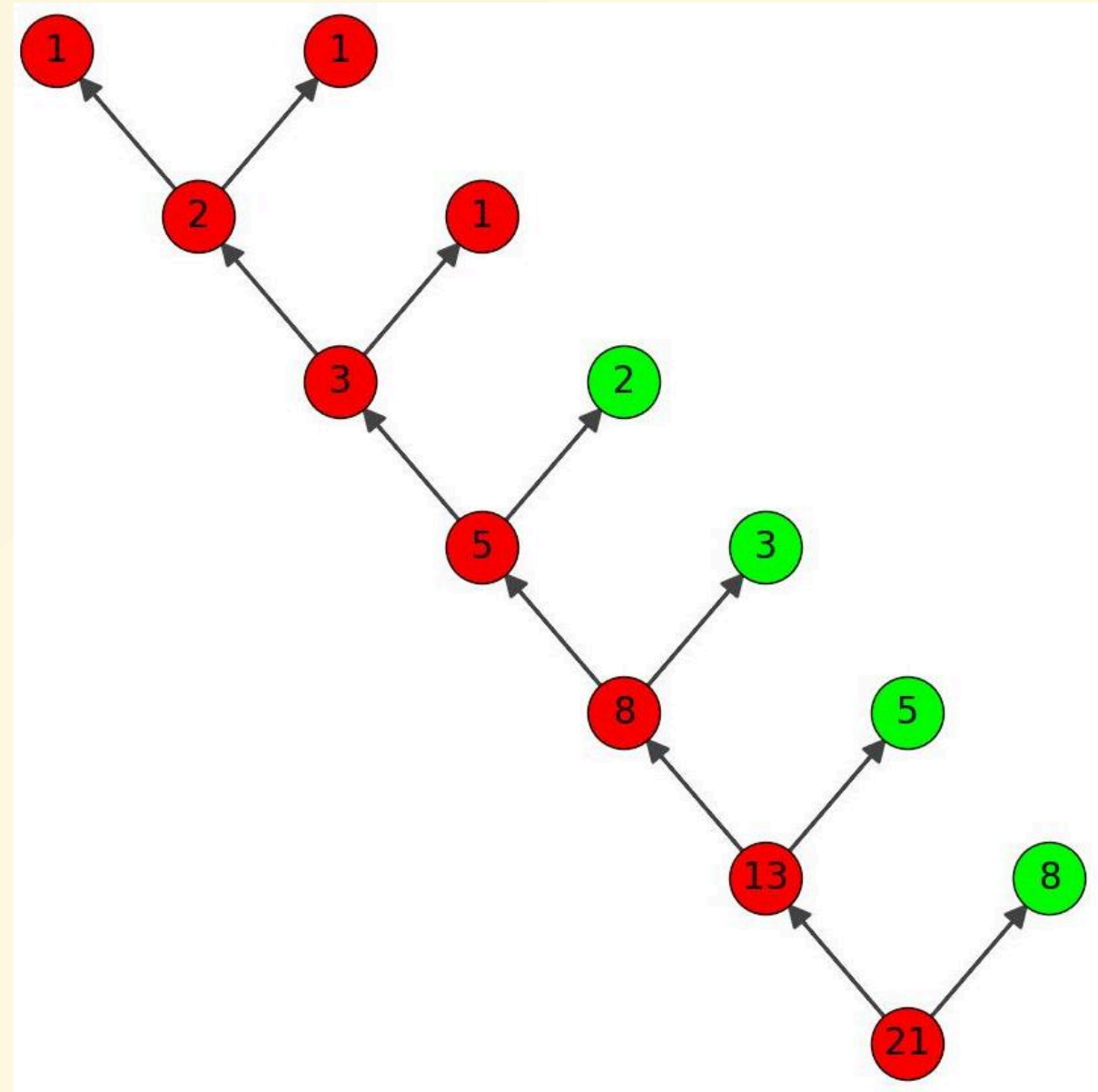




Меморизация

```
def fib(n: int) -> int:
    global memory
    if n in memory:
        return memory[n]
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    memory[n] = fib(n - 1) + fib(n-2)
    return memory[n]
```


- `fib_with_memory`
- 13 вызовов функции `fib`




Стек вызовов функций

Ошибка `stack overflow` - переполнение стека
ВЫЗОВОВ.

 Есть рекурсивный цикл

 Слишком много вызовов, нужна мемоизация

 Выделенный стек маленький, необходимо его
расширить

Стек вызовов функций (call stack) - это механизм хранения информации о вызванных функциях во время выполнения программы. Каждый раз, когда функция вызывается, информация о её выполнении добавляется в верхнюю часть стека. При завершении выполнения функции, она удаляется из стека.

Стек вызовов функций позволяет программе отслеживать порядок вызовов функций и правильно возвращаться к вызывающей функции после завершения выполнения текущей. Он также помогает избежать переполнения стека (stack overflow) путем ограничения максимальной глубины стека вызовов.

Thank you for attention

@pr0bk4 - Степан

@googlewaitme - Булат

https://vk.com/suai_it