Dynamic programming algorithms

Silvio Peroni

✉ silvio.peroni@unibo.it iD 0000-0003-0530-4305 @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)

Second Cycle Degree in Digital Humanities and Digital Knowledge

Alma Mater Studiorum - Università di Bologna

Communication 1

Wednesday the 6th of December there will be the introduction to the project specifications and rules

Please do not miss this lecture!

Communication 2

In addition to that, the same day (the 6th of December) there will be also the official assessment of the course run by the University

Any question about the previous lecture?

Historic hero: Fibonacci

He was a mathematician

First person to introduce in Europe the Hindu-Arabic number system (i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Publication *Liber Abaci* (Book of Calculation) in 1202: how to use such numeral system for addressing situations related to commerce, and for solving generic mathematical problems

Fibonacci sequence

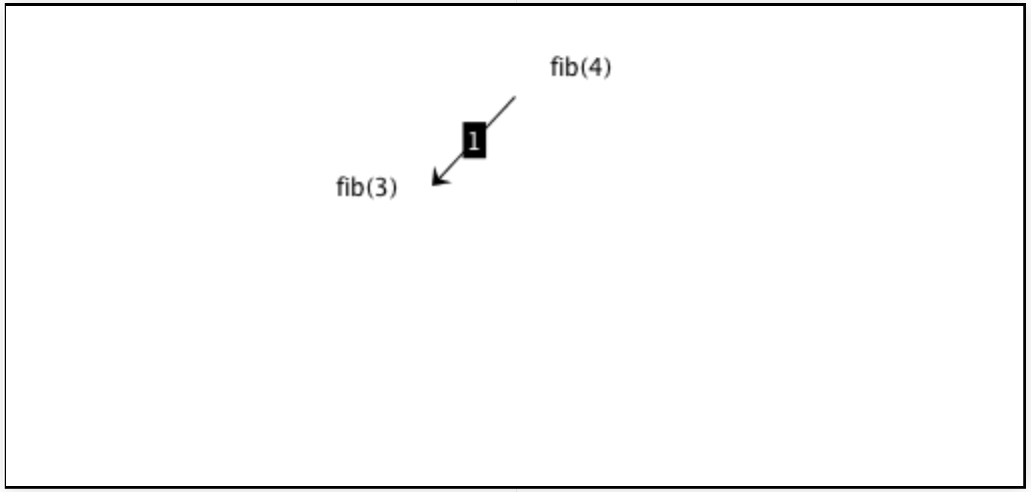
Fibonacci developed an infinite sequence of numbers, named after him, that described ideally the number of male-female pairs of rabbits at a given month

`fib(0) = 0` **[base case 1]**

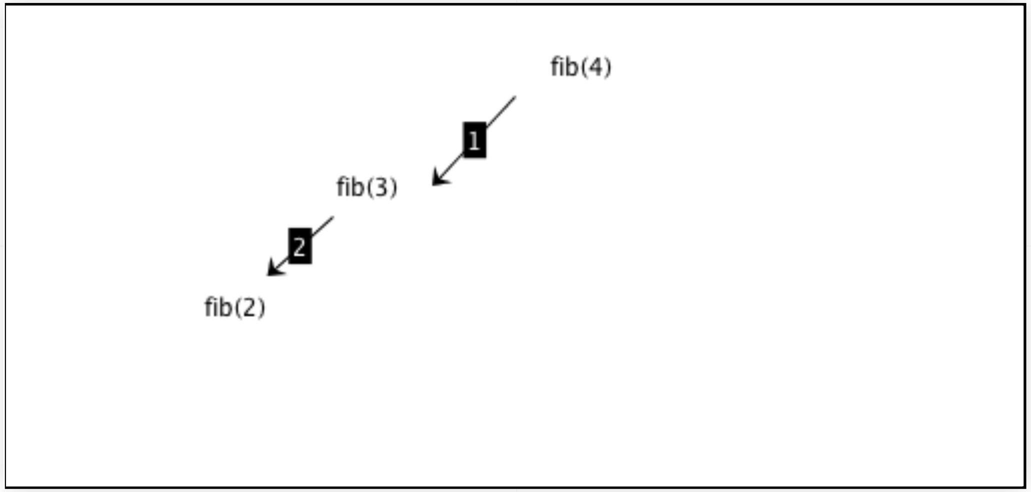
`fib(1) = 1` **[base case 2]**

`fib(n) = fib(n-1) + fib(n-2)` **[recursive step]**

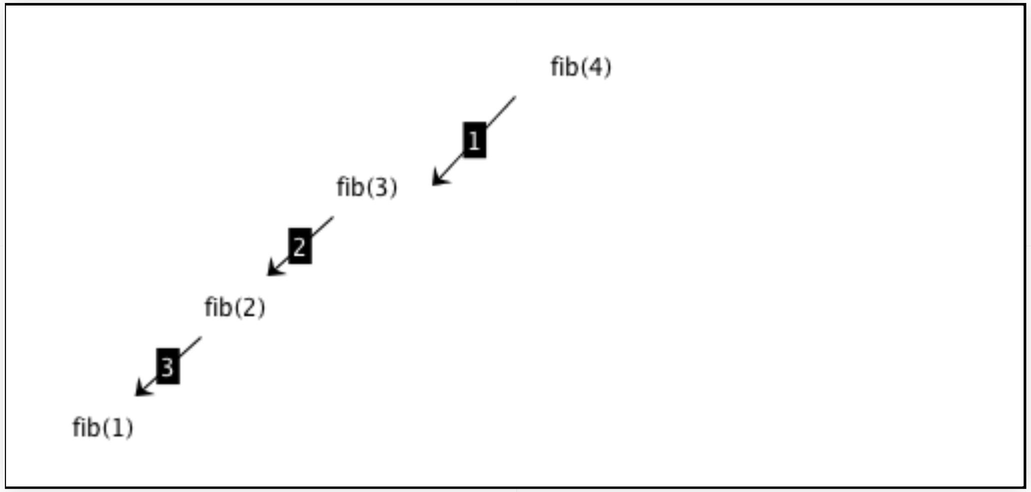
Fibonacci: divide and conquer



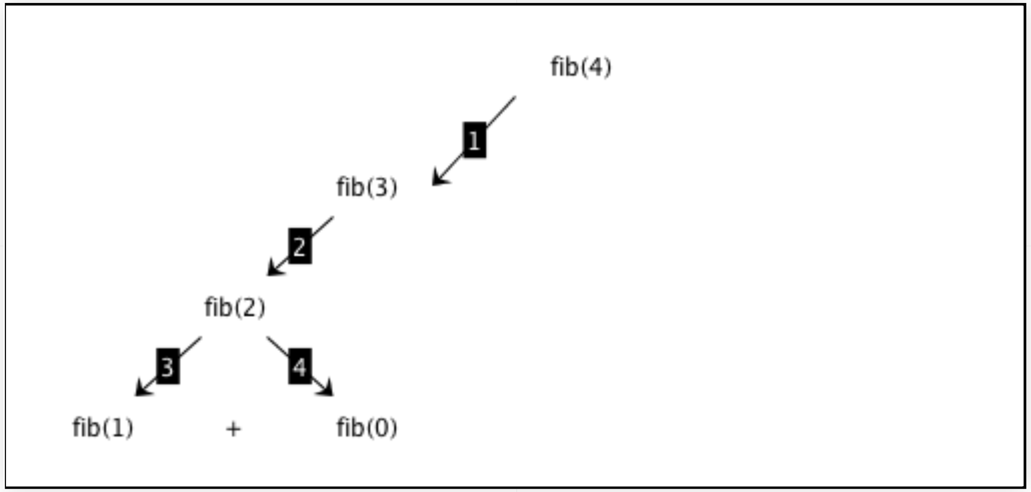
Fibonacci: divide and conquer



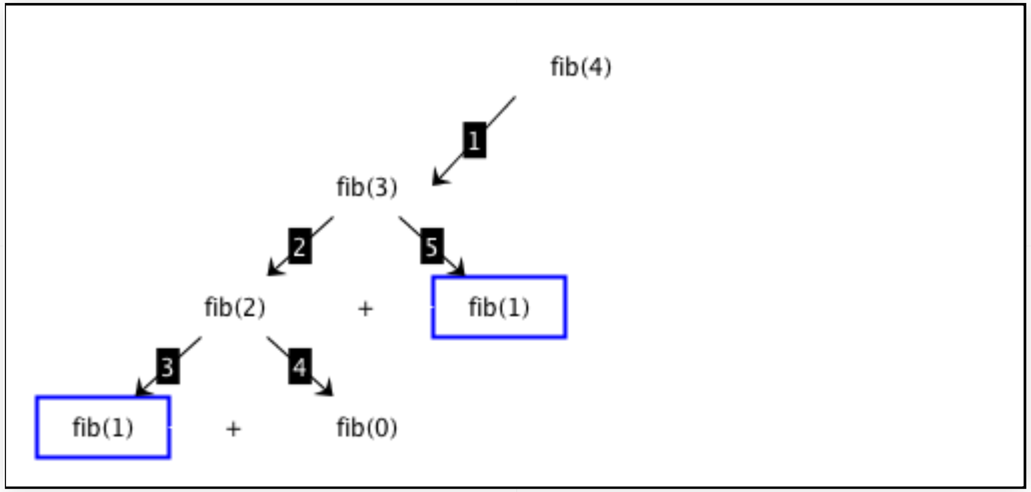
Fibonacci: divide and conquer



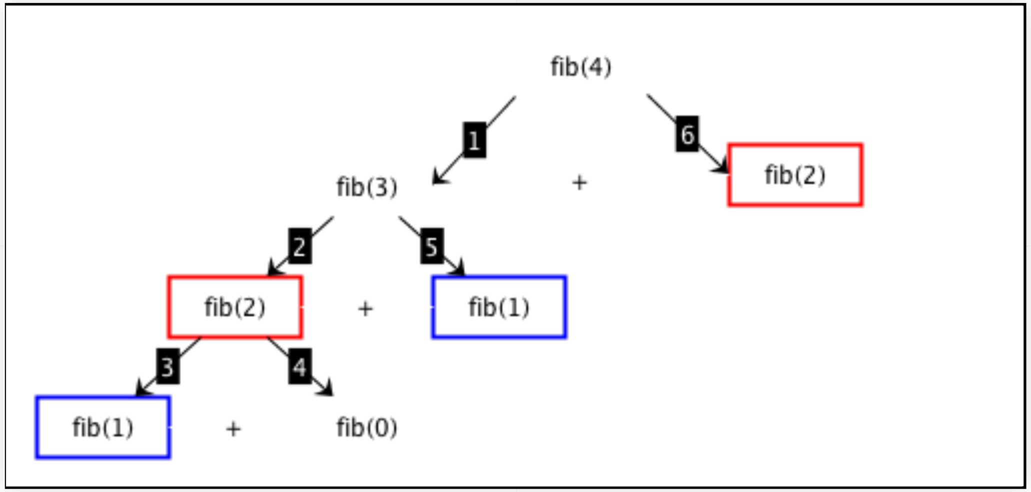
Fibonacci: divide and conquer



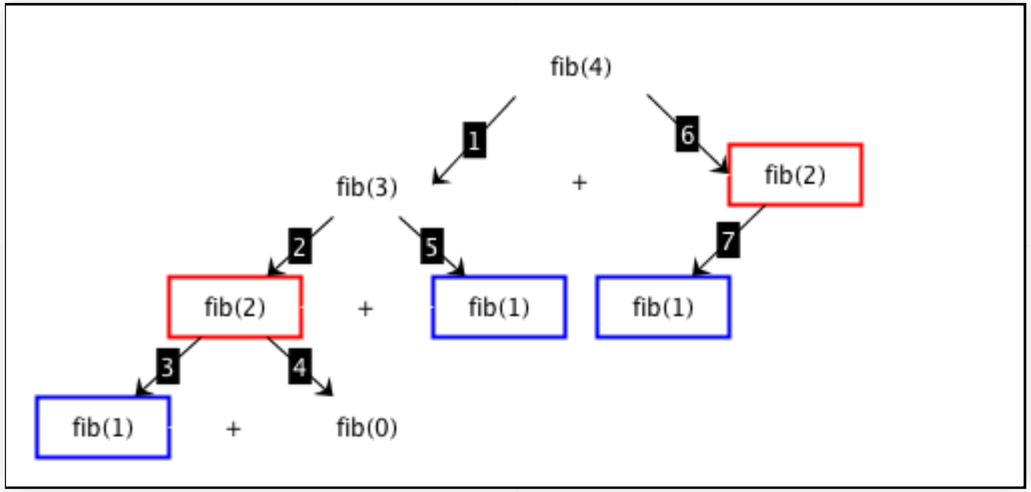
Fibonacci: divide and conquer



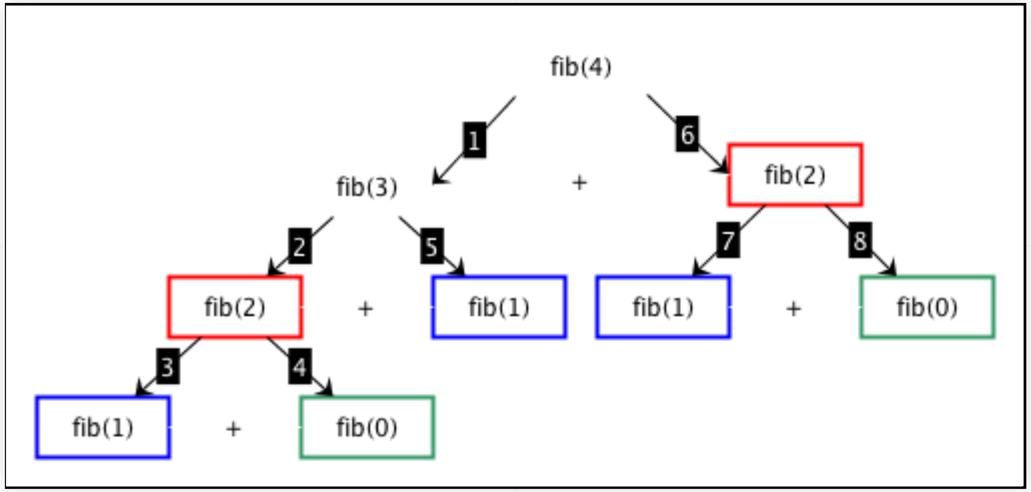
Fibonacci: divide and conquer



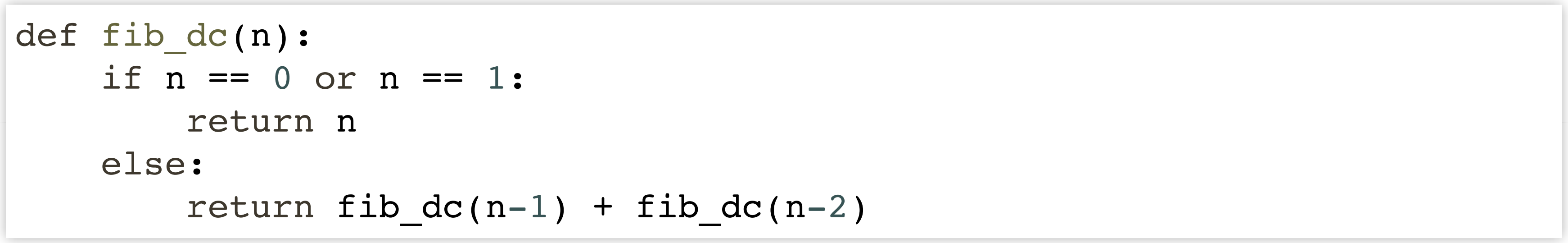
Fibonacci: divide and conquer



Fibonacci: divide and conquer



Fibonacci (divide and conquer): algorithm



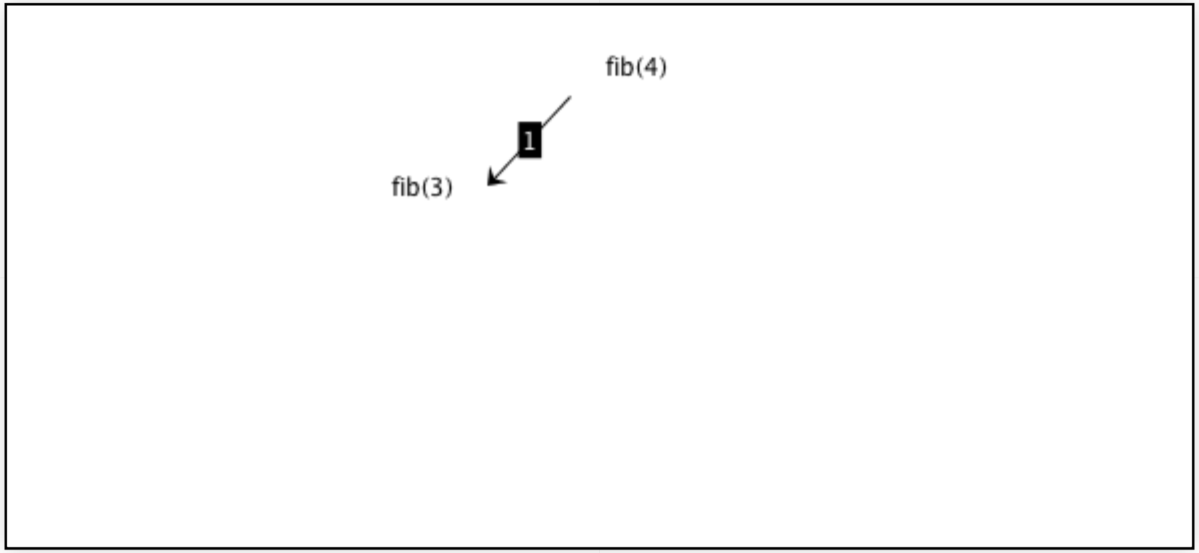
```
def fib_dc(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib_dc(n-1) + fib_dc(n-2)
```

Dynamic programming approach

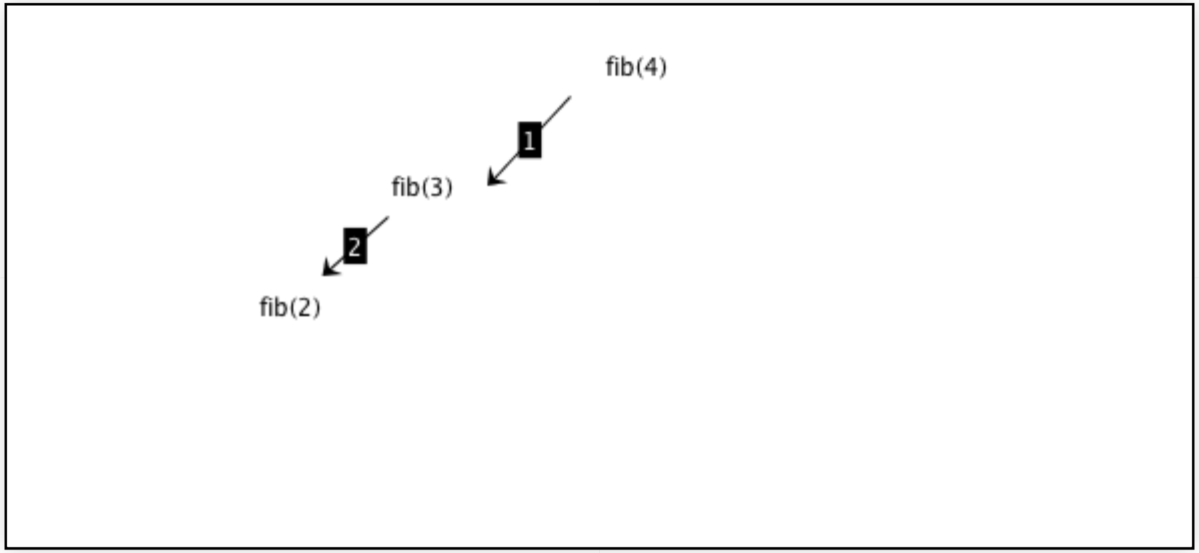
Dynamic programming algorithm is based on six steps

1. **[base case: solution exists]** return the solution calculated previously, otherwise
2. **[base case: address directly]** address directly if it is an easy-to-solve problem, otherwise
3. **[divide]** split the input material into two or more balanced parts, each depicting a sub-problem of the original one
4. **[conquer]** run the same algorithm recursively for every balanced parts obtained in the previous step
5. **[combine]** reconstruct the final solution of the problem by means of the partial solutions
6. **[memorize]** store the solution to the problem for reusing it

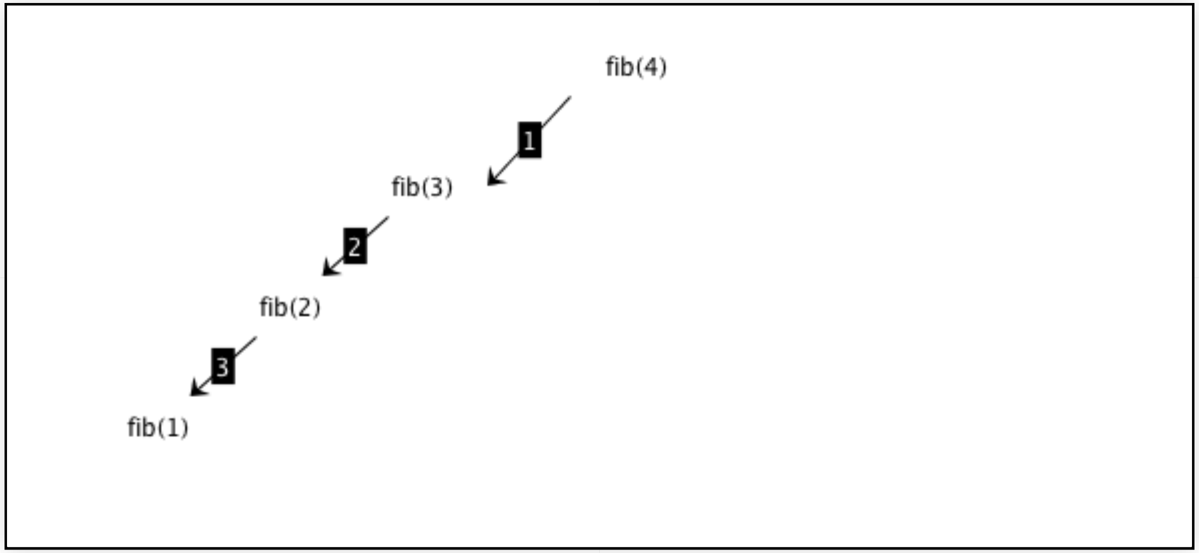
Fibonacci: dynamic programming



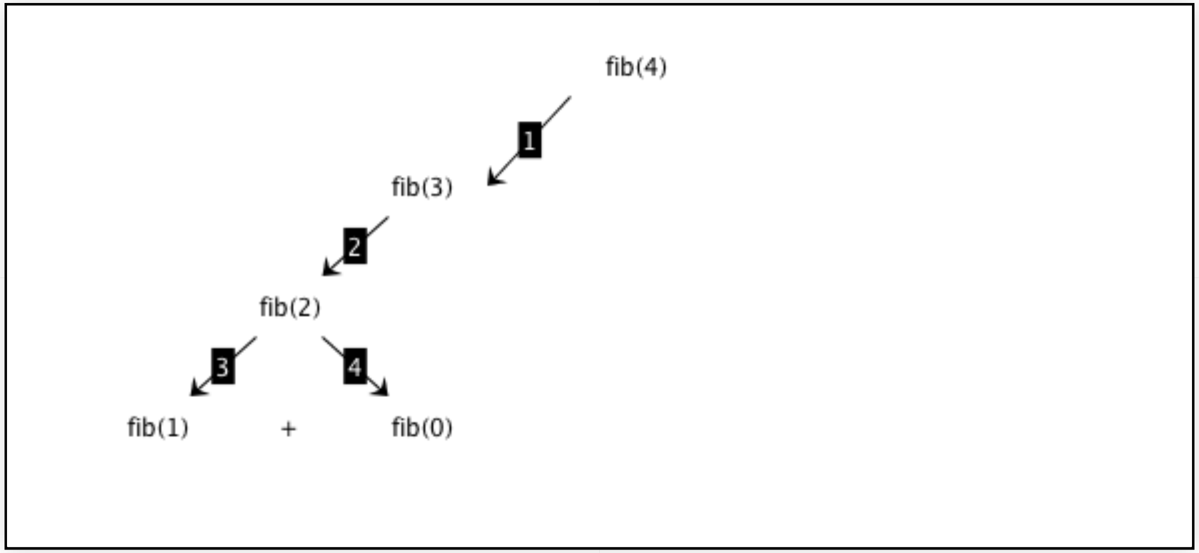
Fibonacci: dynamic programming



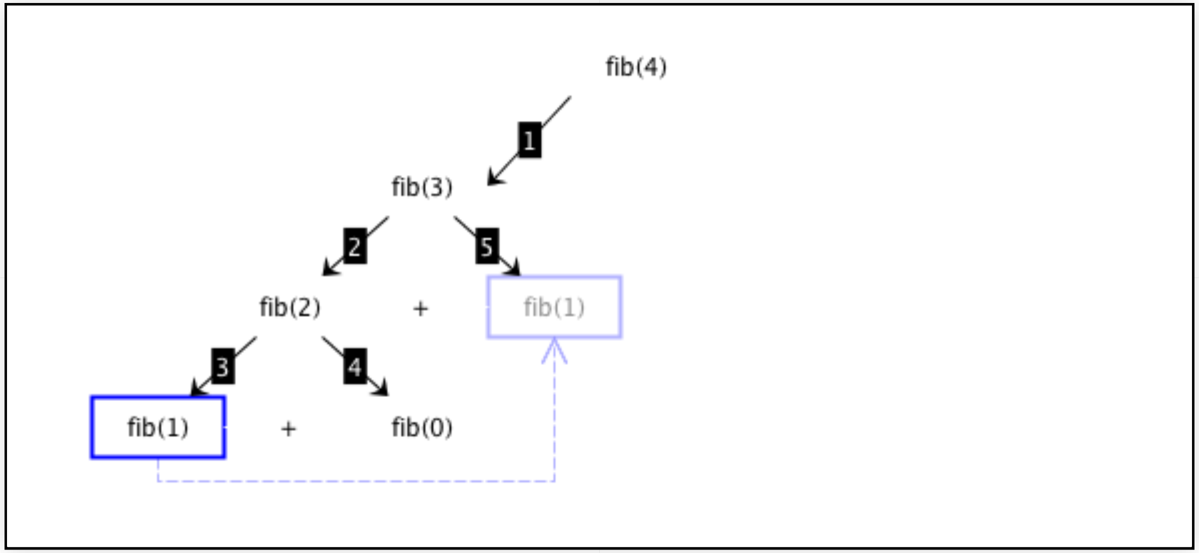
Fibonacci: dynamic programming



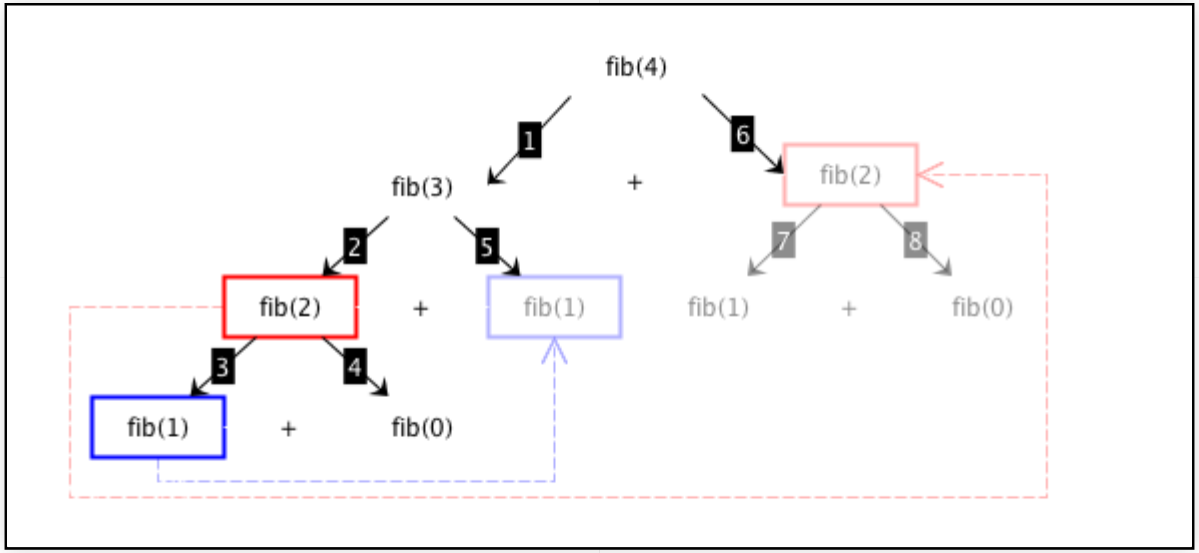
Fibonacci: dynamic programming



Fibonacci: dynamic programming



Fibonacci: dynamic programming



Fibonacci: ancillary operations

Non-inclusion in dictionary:

```
<key> not in <dictionary>
```

Comparison that returns `True` if <key> is not included as key in any pair of <dictionary>

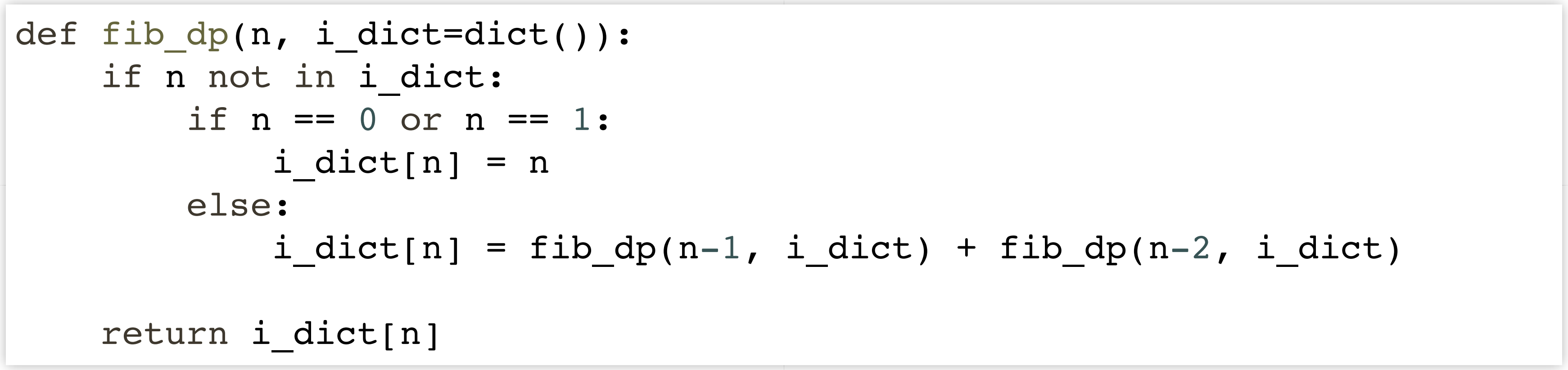
Parameter with default assignment

```
def <algorithm>(<param_d>=<default>)
```

Initialises `<param_d>` with the default value specified if no value is passed for the execution of the algorithm

E.g., considering `def test(n=0)`, executing `test(4)` assigns the specified number to `n`, while executing `test()` assigns the default value *0* to `n`

Fibonacci (dynamic programming): algorithm



```python
def fib_dp(n, i_dict=dict()):
    if n not in i_dict:
        if n == 0 or n == 1:
            i_dict[n] = n
        else:
            i_dict[n] = fib_dp(n-1, i_dict) + fib_dp(n-2, i_dict)

    return i_dict[n]
```

END

Dynamic programming algorithms

Silvio Peroni

✉ silvio.peroni@unibo.it iD 0000-0003-0530-4305 @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)

Second Cycle Degree in Digital Humanities and Digital Knowledge

Alma Mater Studiorum - Università di Bologna