Is Plain Recursion Good Enough?

This lesson will argue for using dynamic programming instead of recursion to solve complex problems.

We'll cover the following

- Recursion = brute force
- Recursion's inefficiency visualized

Recursion = brute force

We saw some examples of recursion earlier in this chapter. One recurring theme (pun intended) in all those problems was high time complexity. Why was that the case? Because all those problems required a brute force search. We had to search in the whole solution space to look for an answer. For example, in the permutation challenge, we had to find all the possible combinations of characters. In the N queens problem, we had to look at all possible placements of queens until we found an optimal one. So, recursion in most cases is an excellent brute force technique. But as the name also suggests, it is not a very elegant one; thus, we will be learning about techniques that make recursion more efficient.

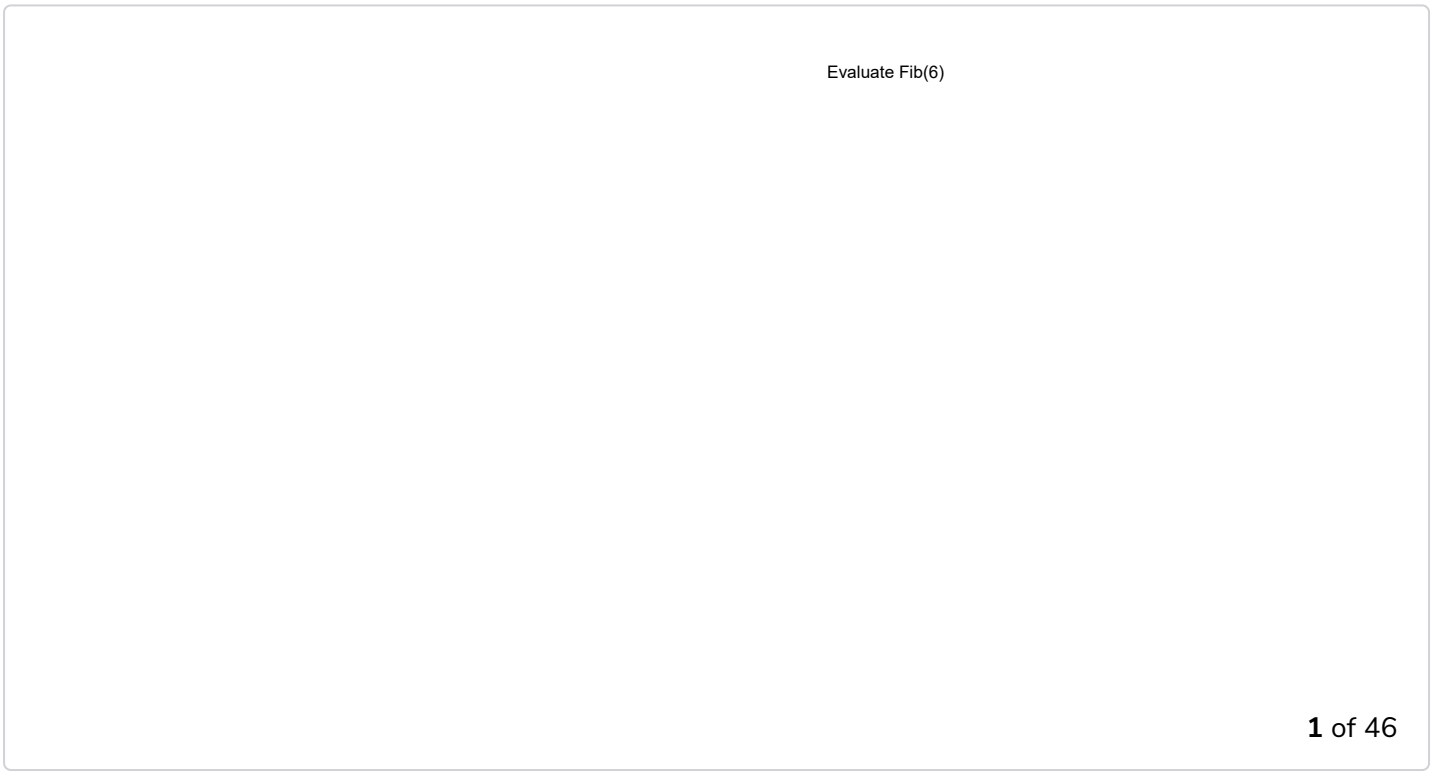
Recursion's inefficiency visualized

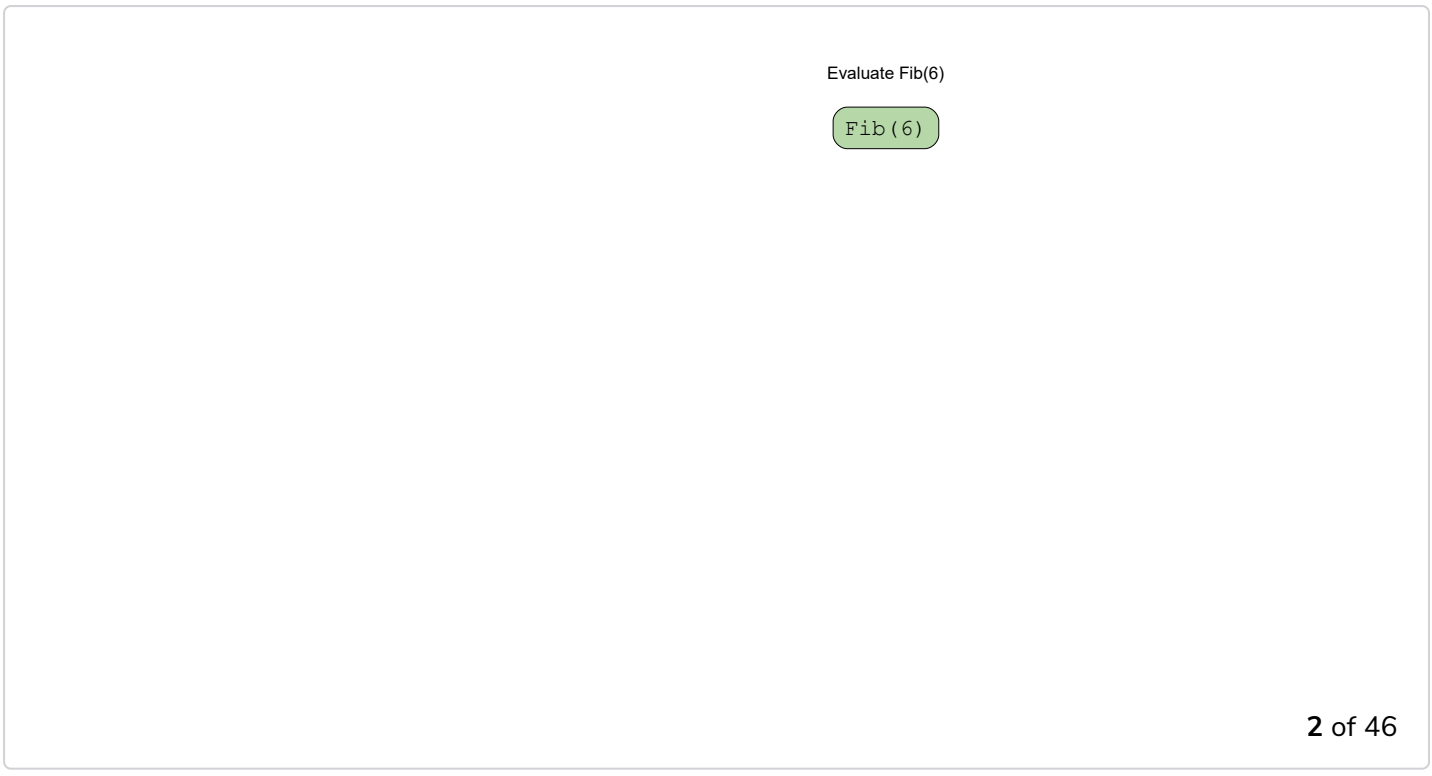
Let's rerun this Fibonacci numbers algorithm given in the start of this chapter. This time, we have plotted the time taken by each Fibonacci number to compute. Try different values by updating the value of variable `numbers`:

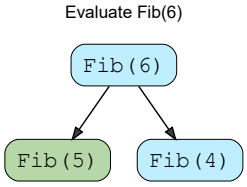
```
import time
import matplotlib.pyplot as plt

def fib(n):
  if n <= 0: # base case 1
    return 0
  if n <= 1: # base case 2
    return 1
  else: # recursive step
    return fib(n-1) + fib(n-2)

numbers = 10
```

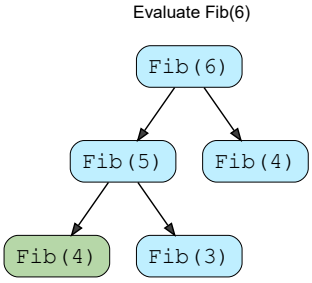
If you tried bigger values, you noticed how the curve became more and more prominent. This shows how bad the exponential time complexity is. Can we do better than this?

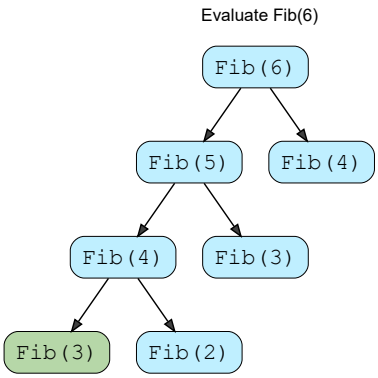
Let’s revisit this visualization of Fibonacci numbers from the previous chapter.

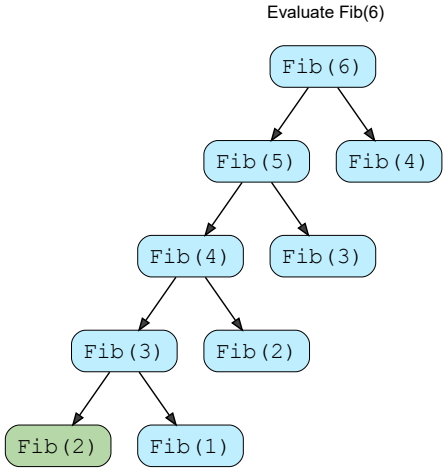


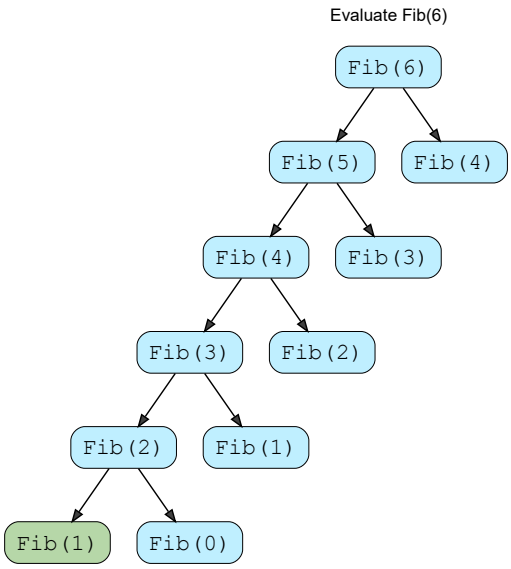


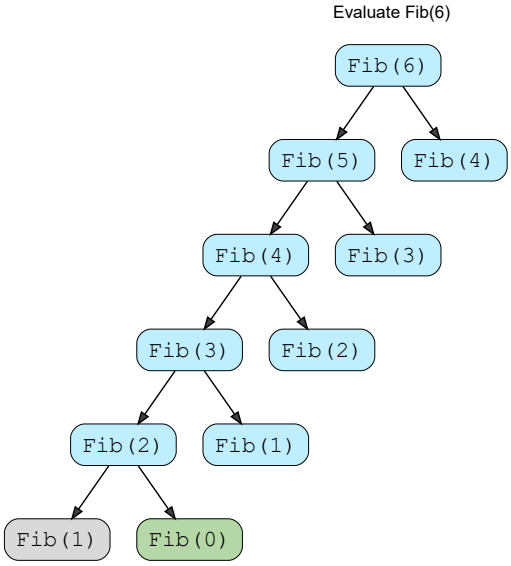

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)

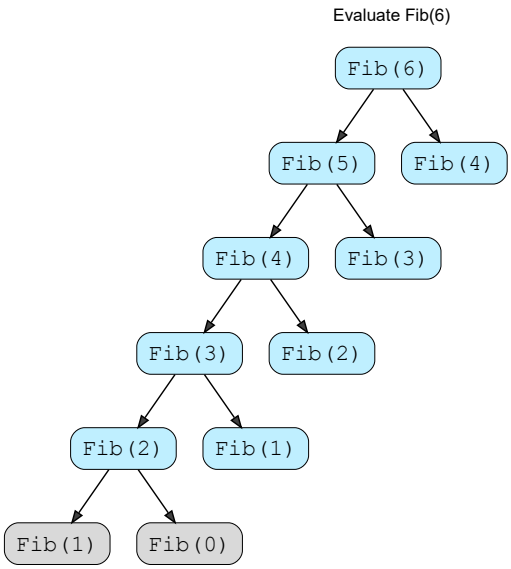

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)

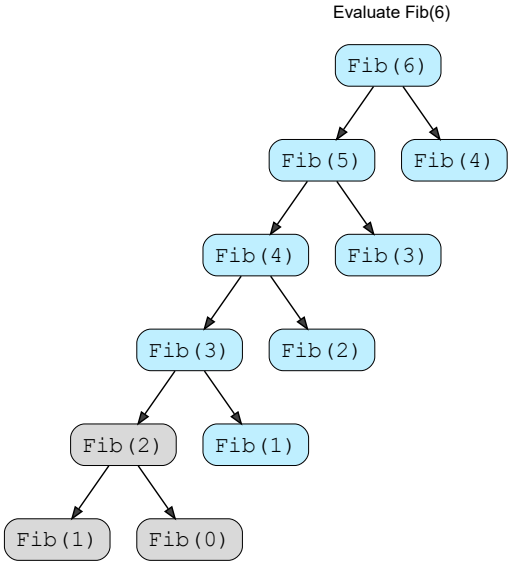

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)

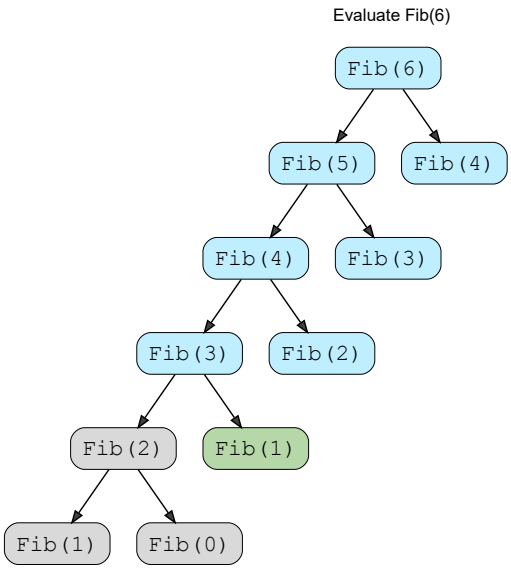

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)

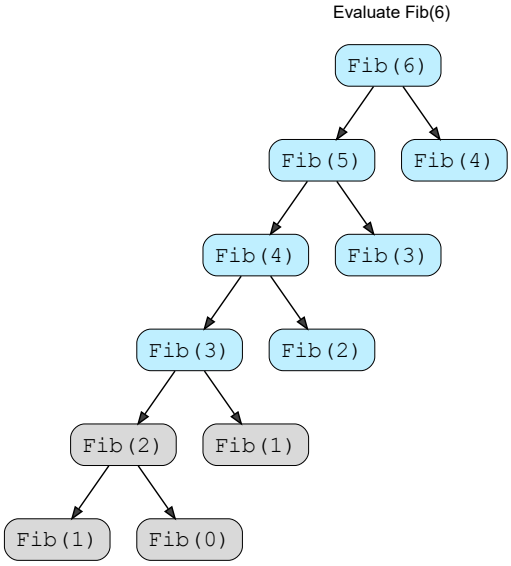

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

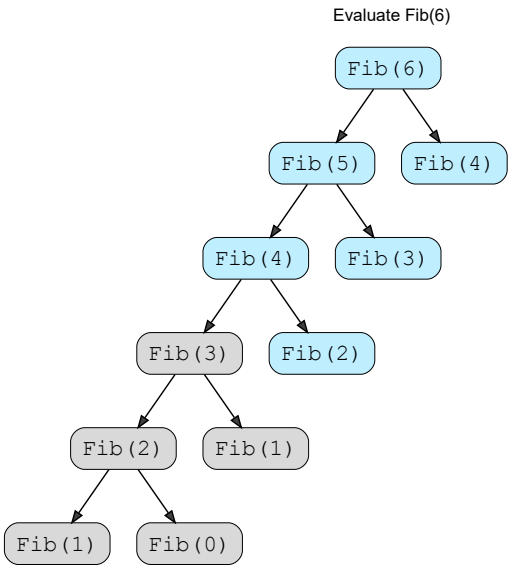

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

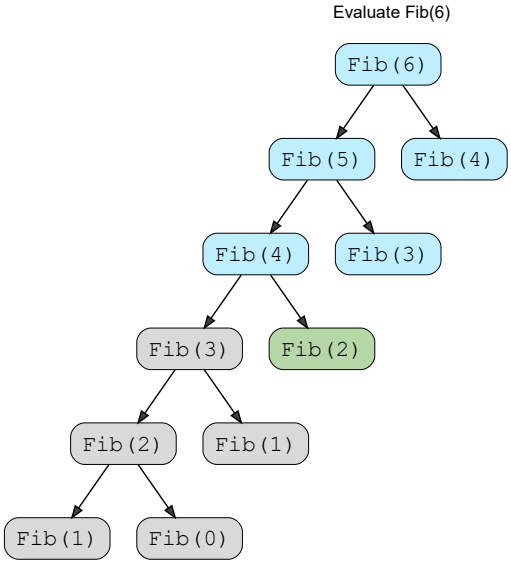

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

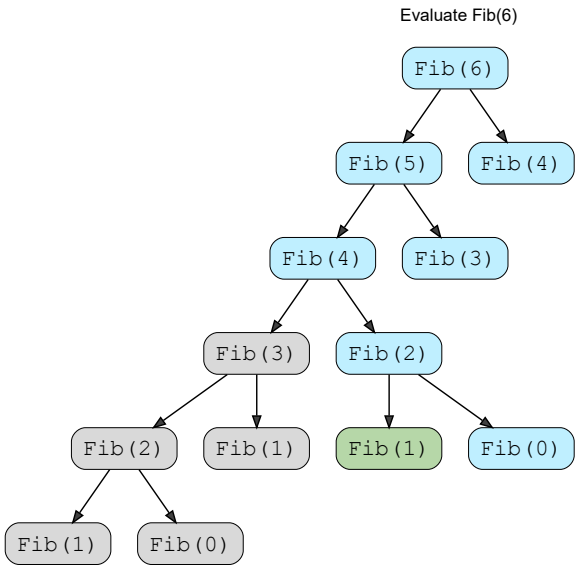

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

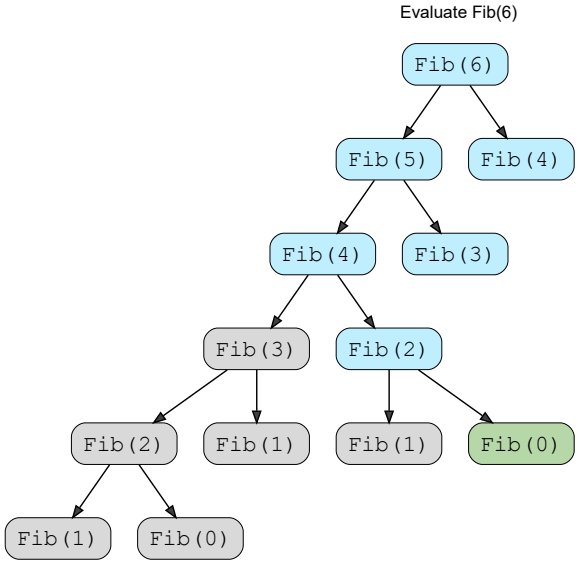

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

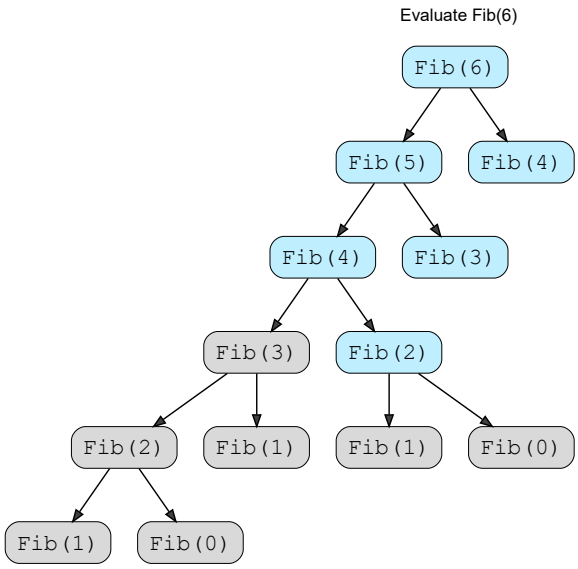

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

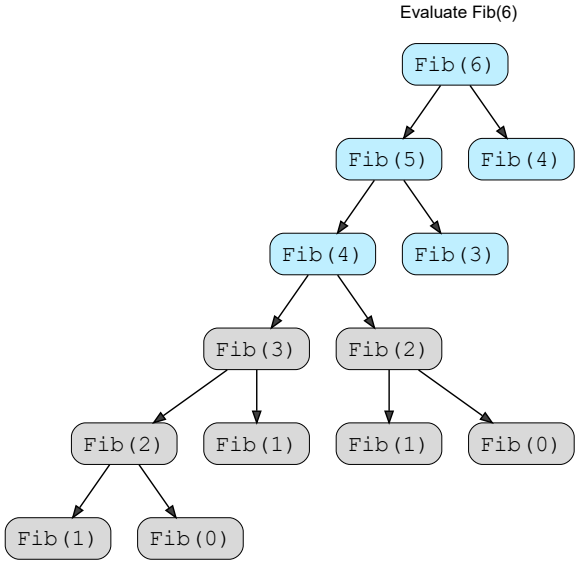

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

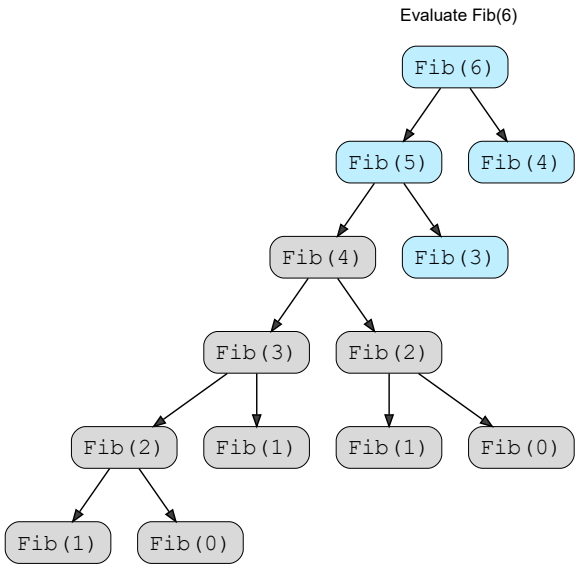

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

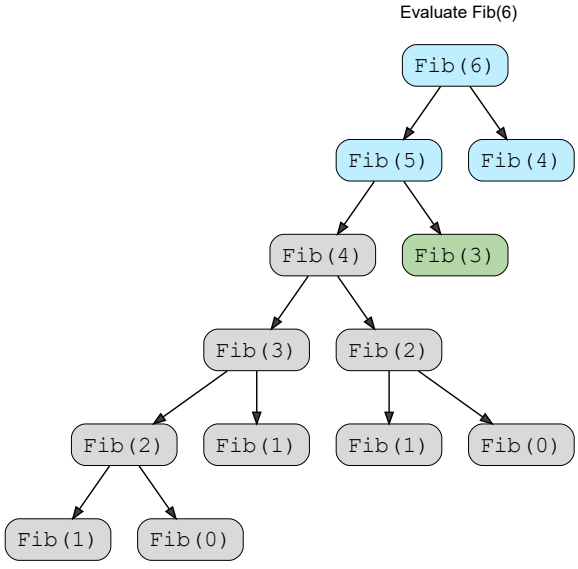

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

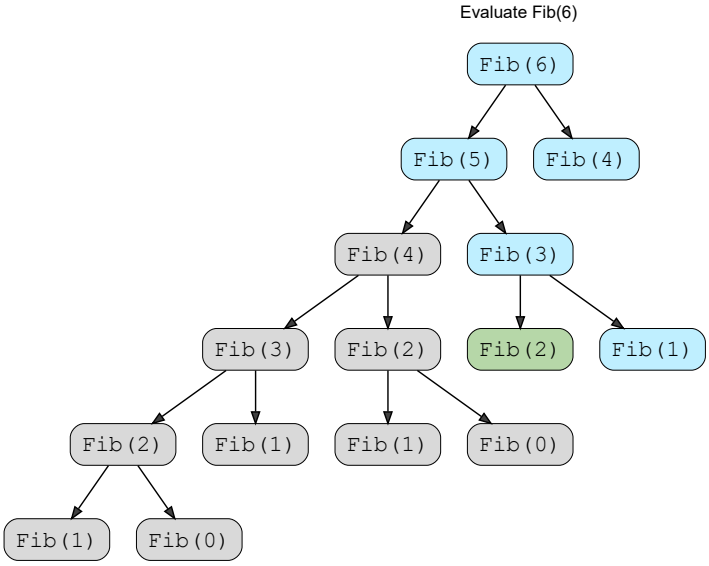

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

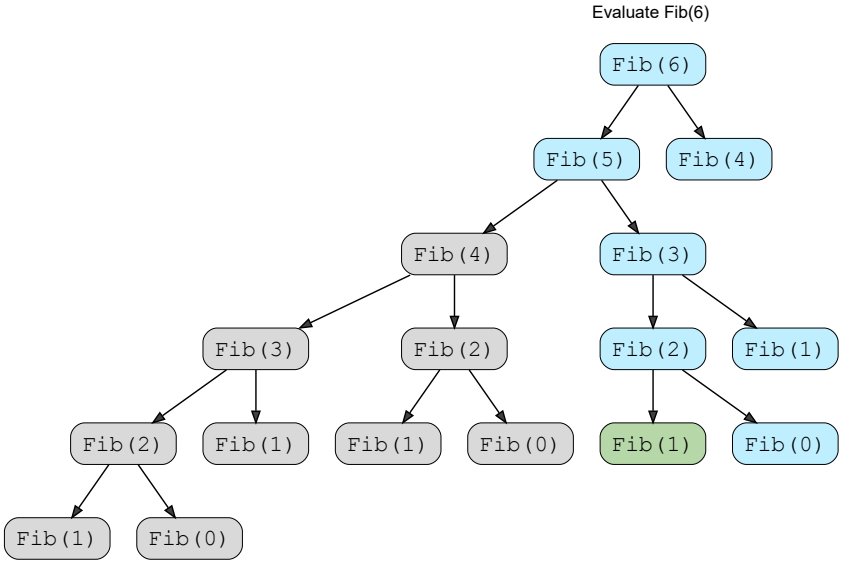

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

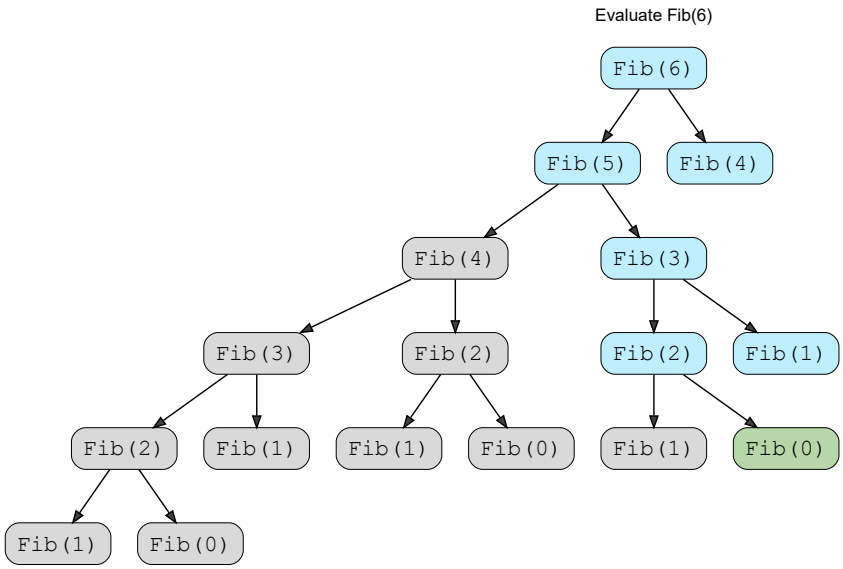

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

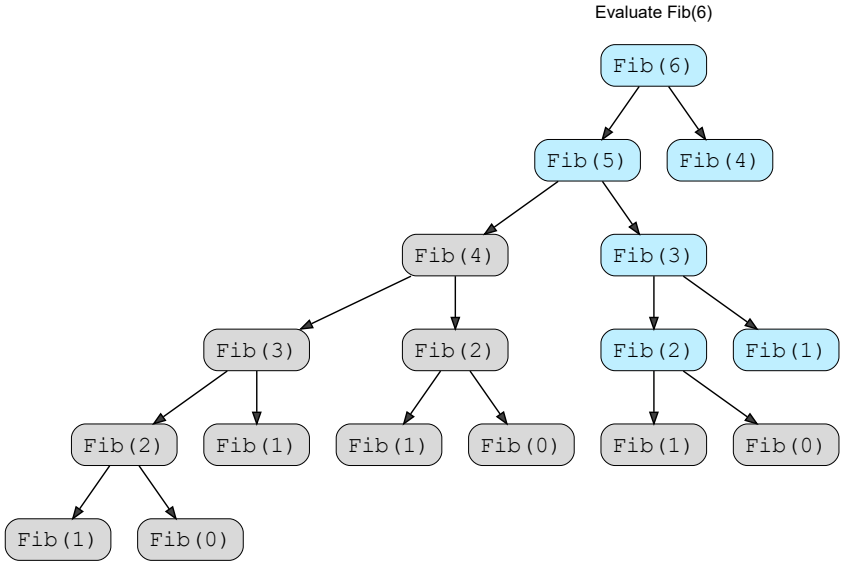

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

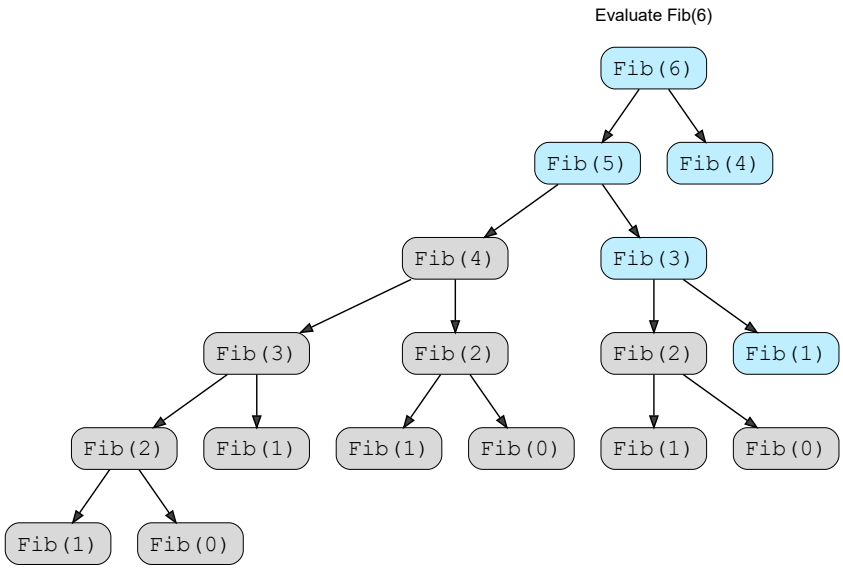

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

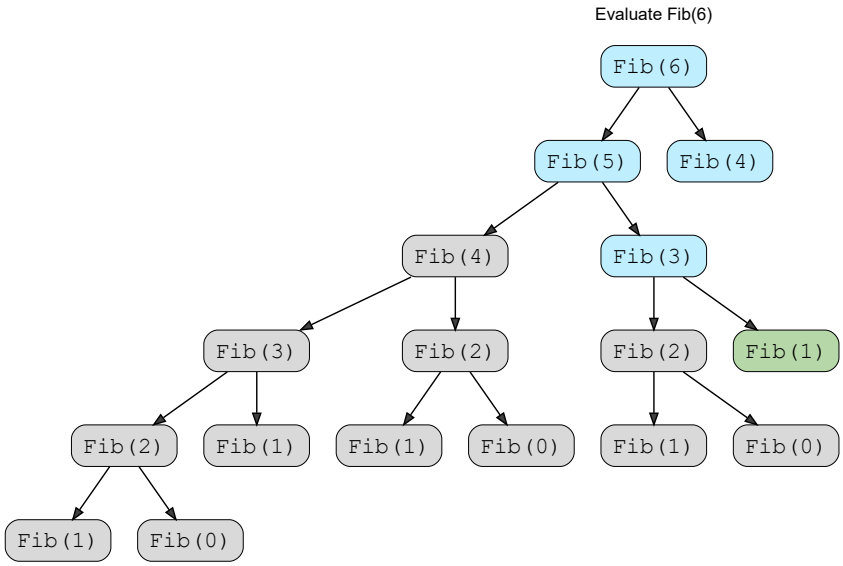

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

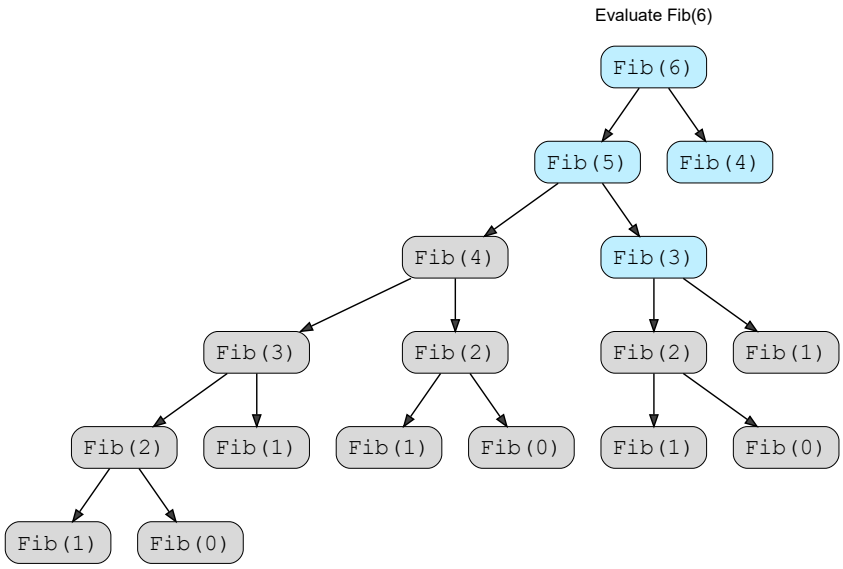

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

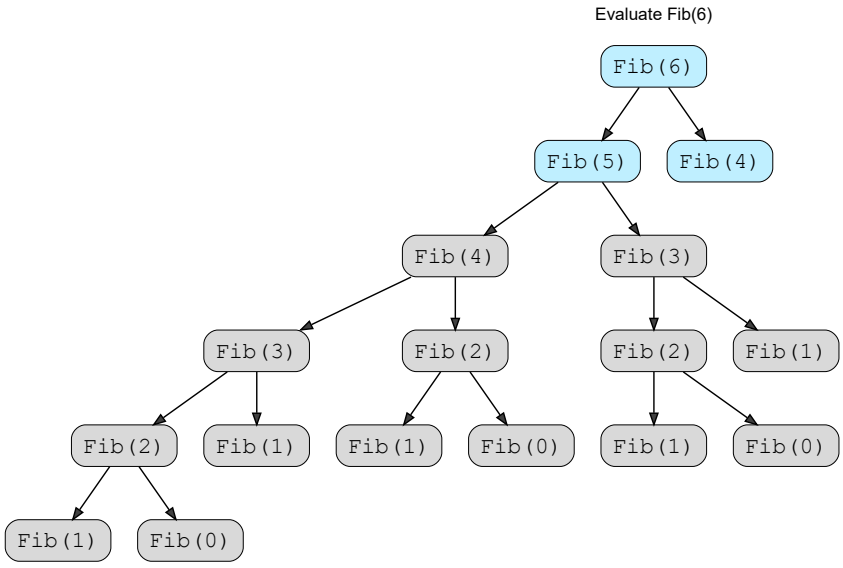

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

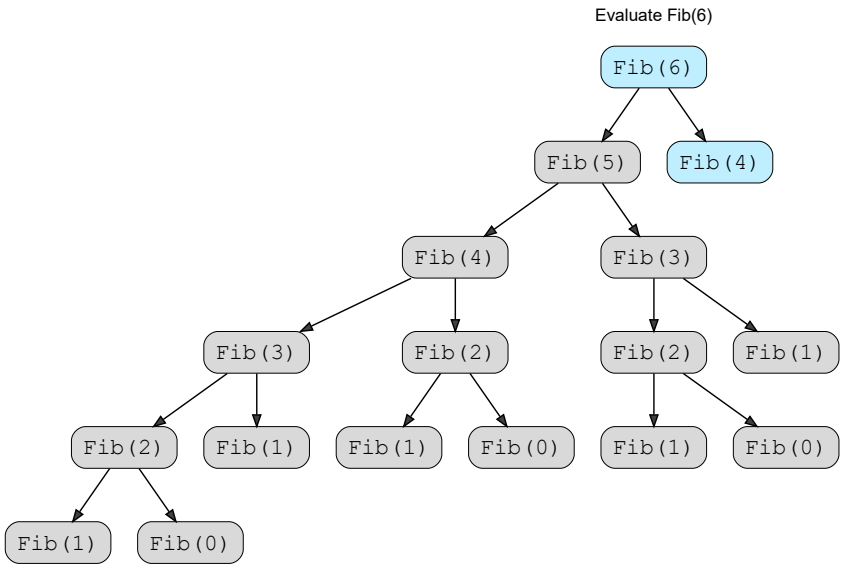

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

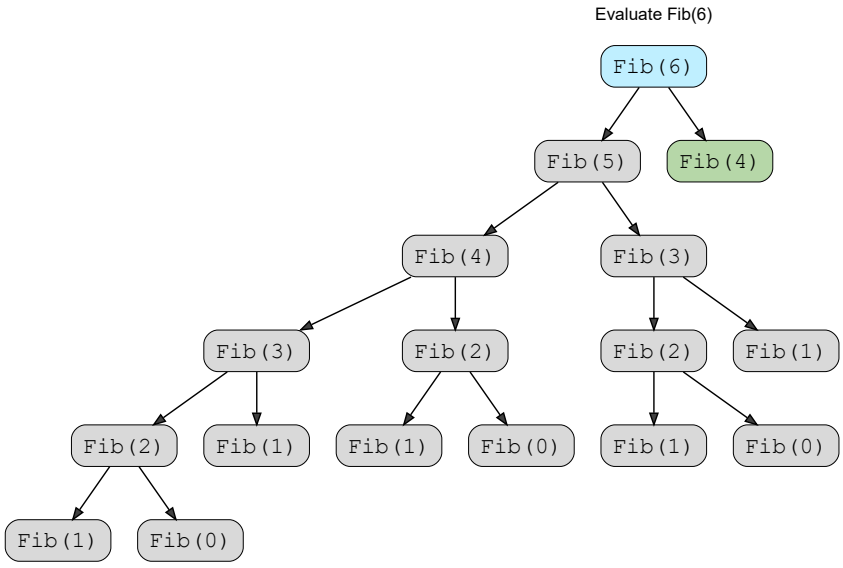

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

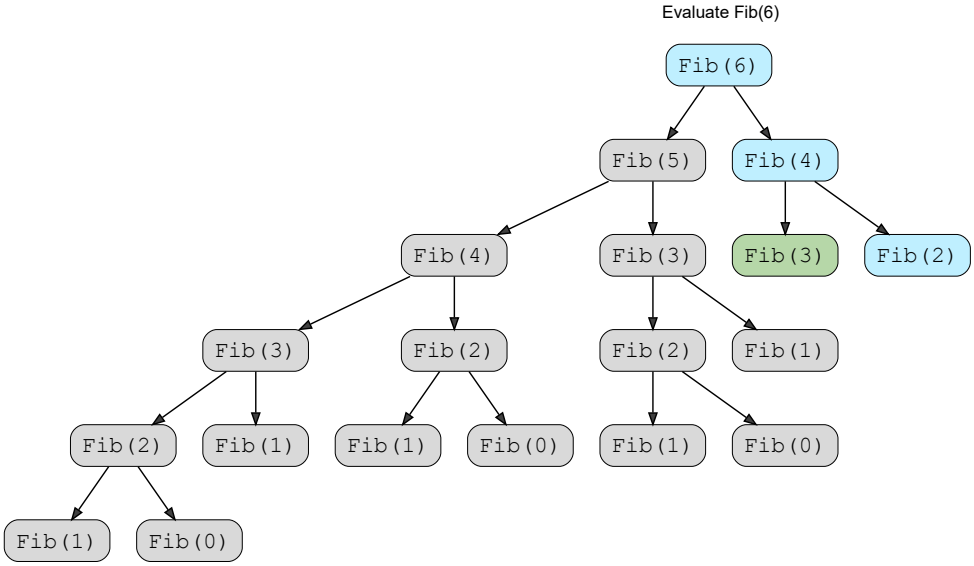

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

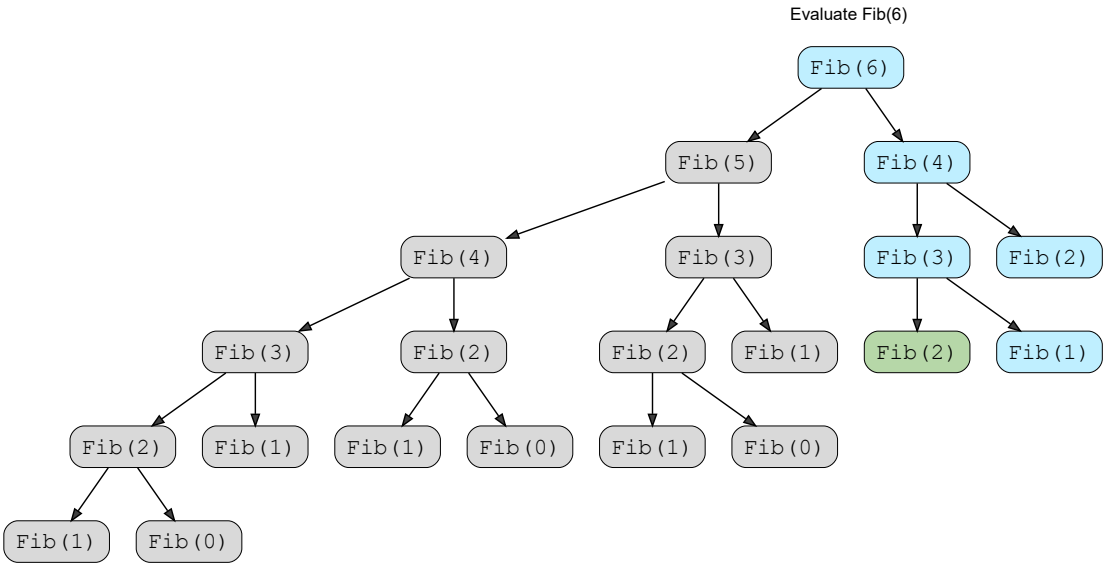

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

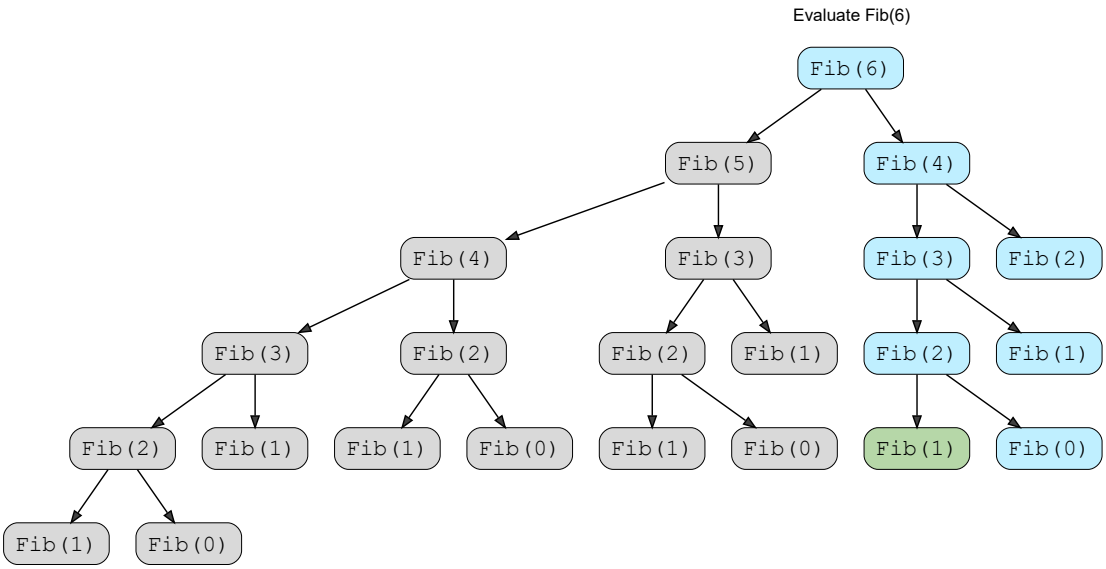

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

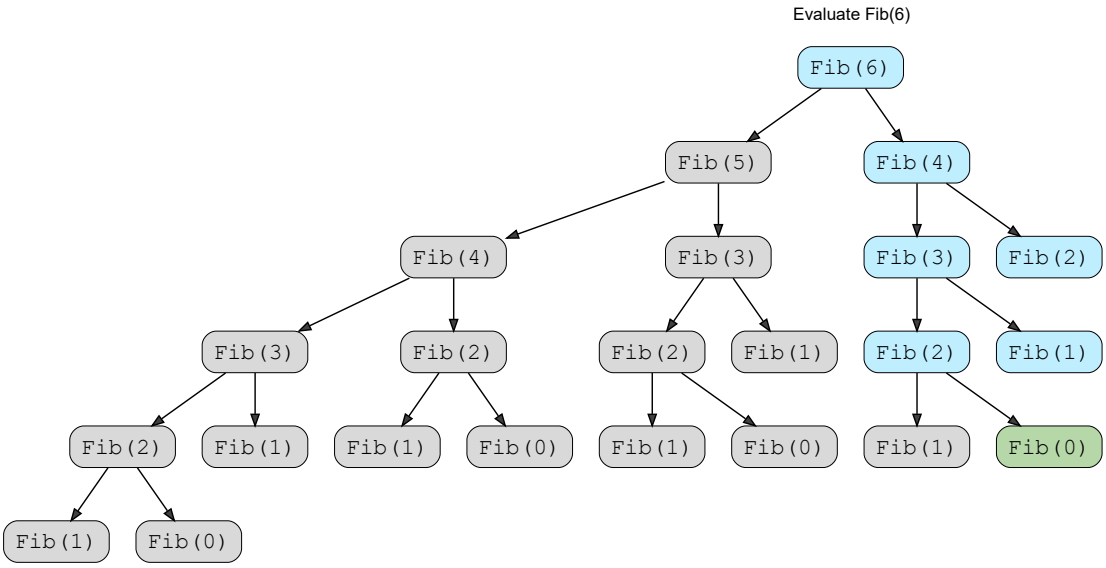

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

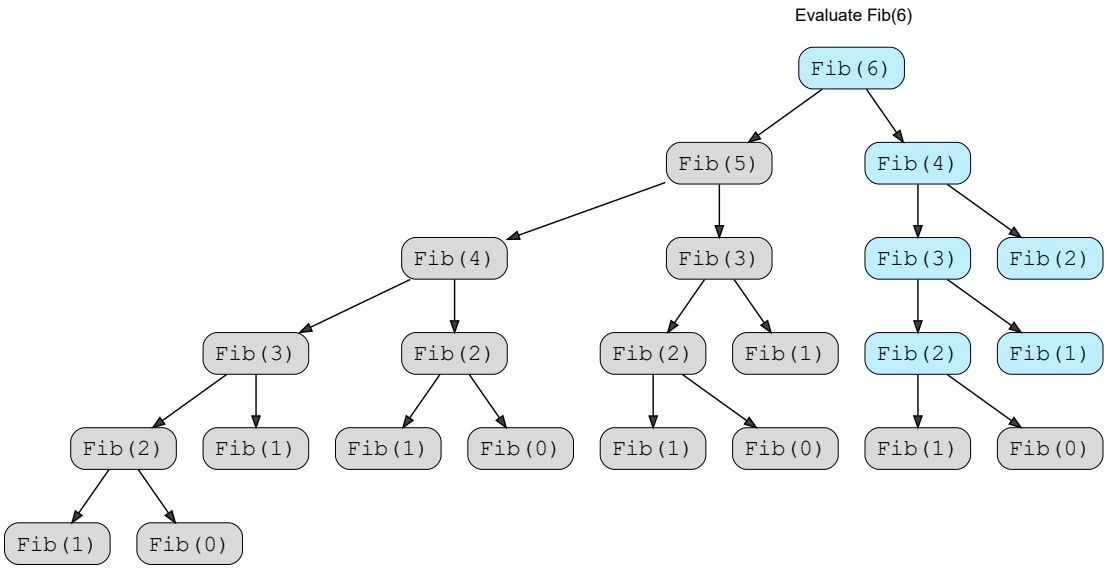

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

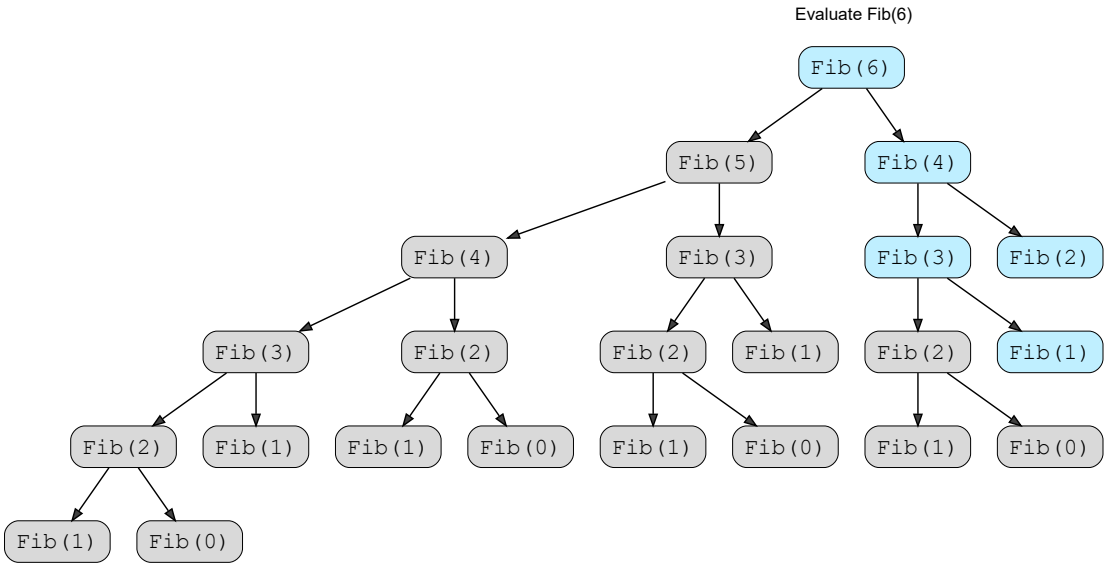

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

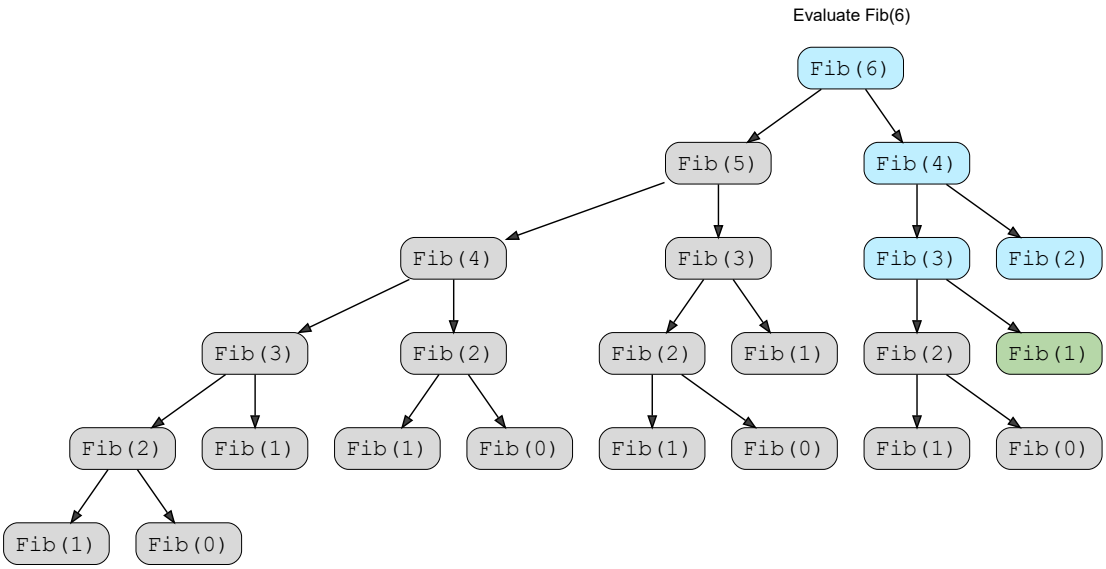

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

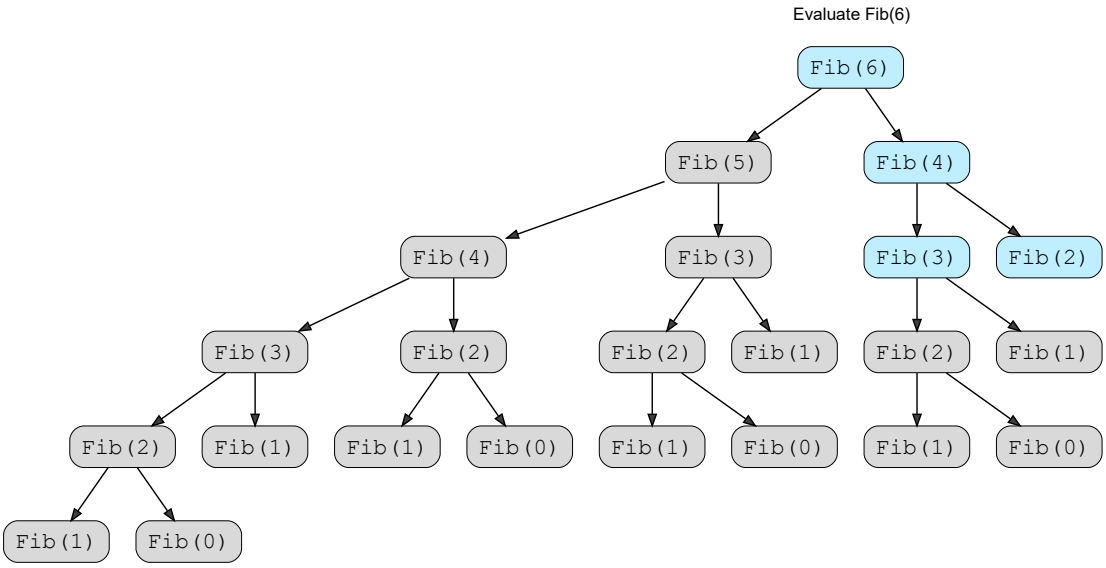

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

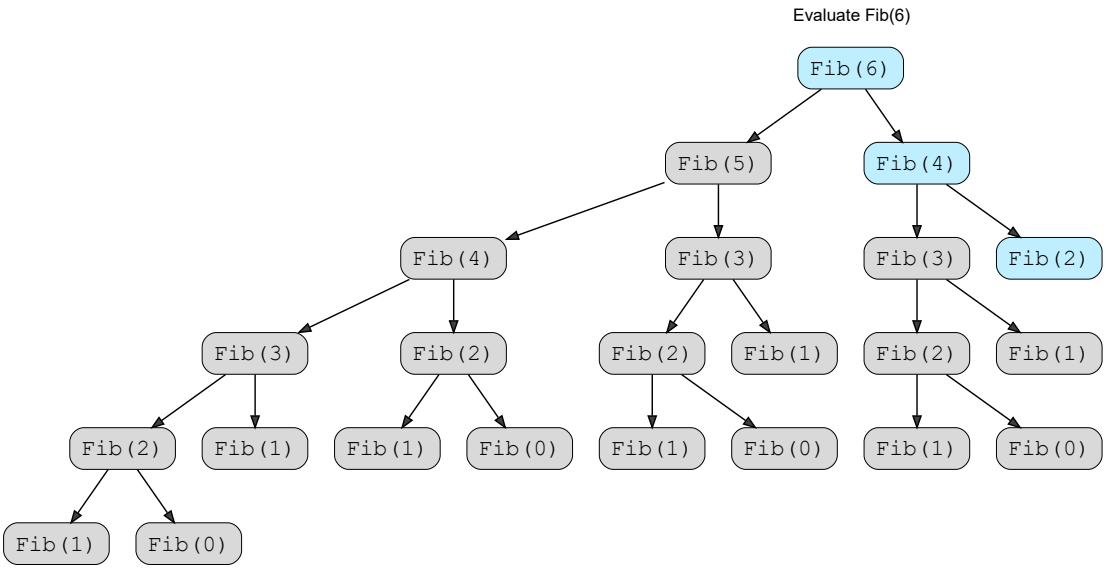

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

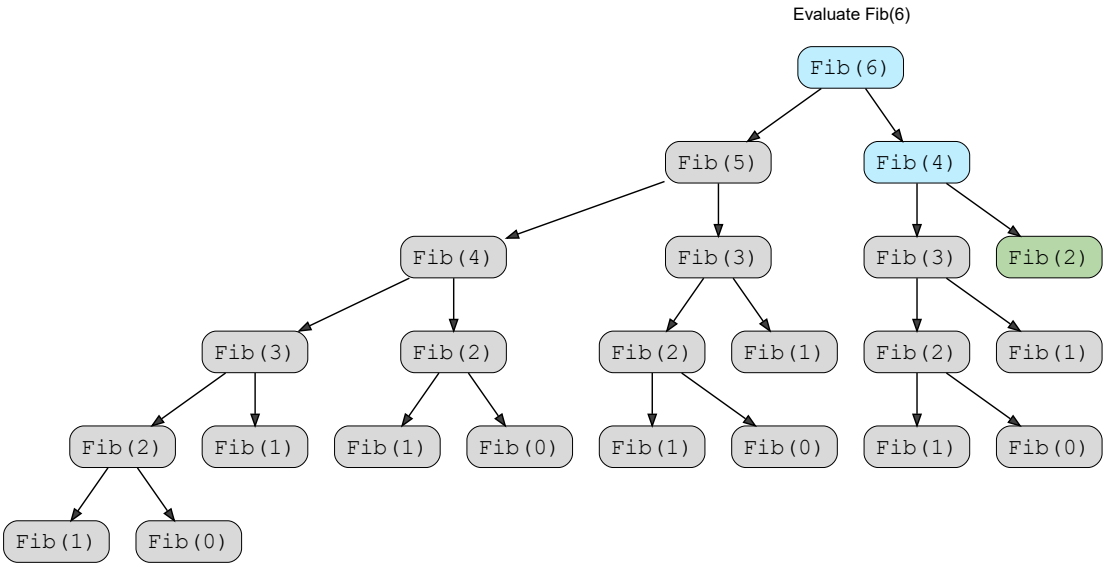

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

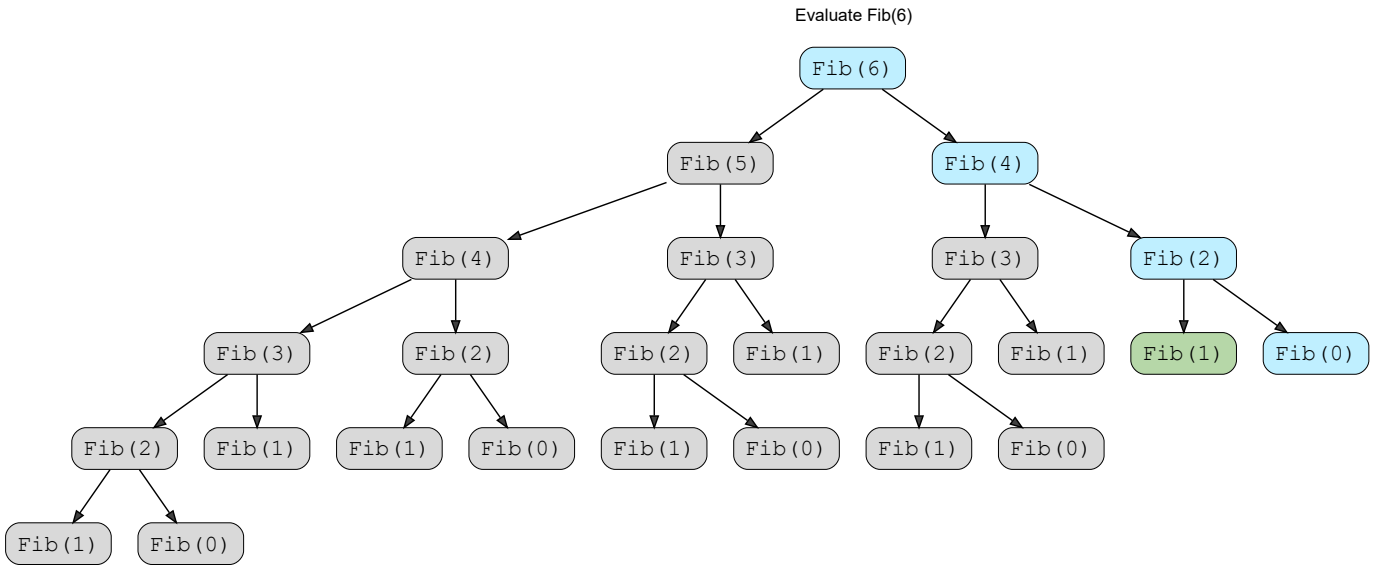

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

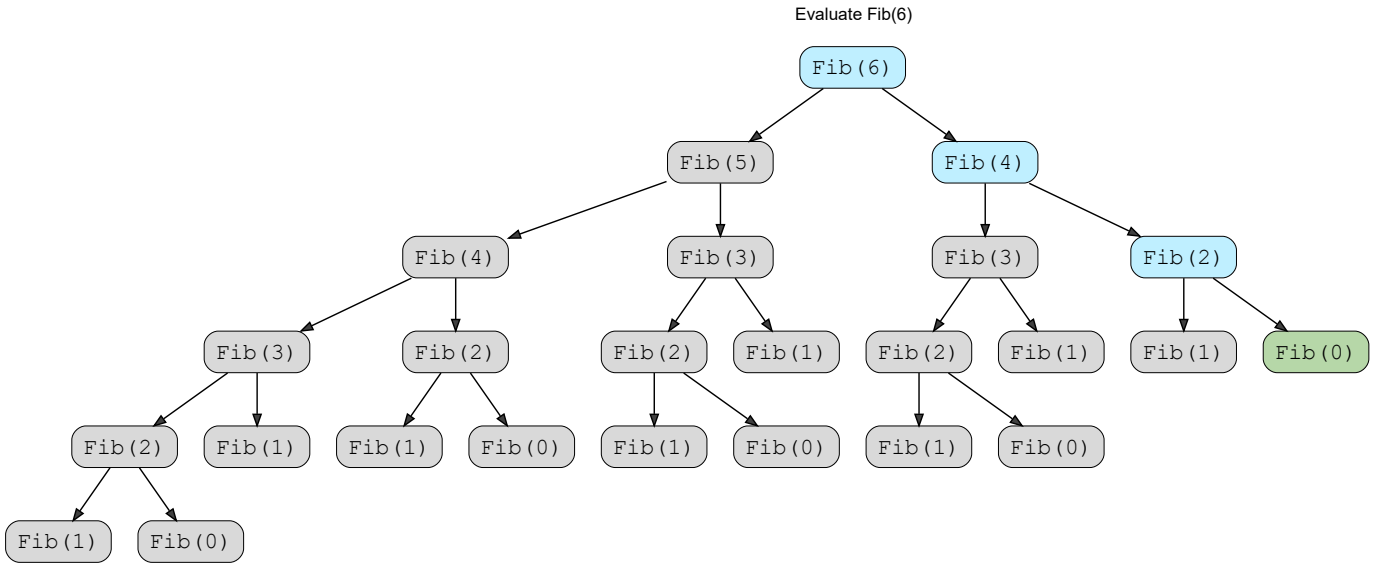

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

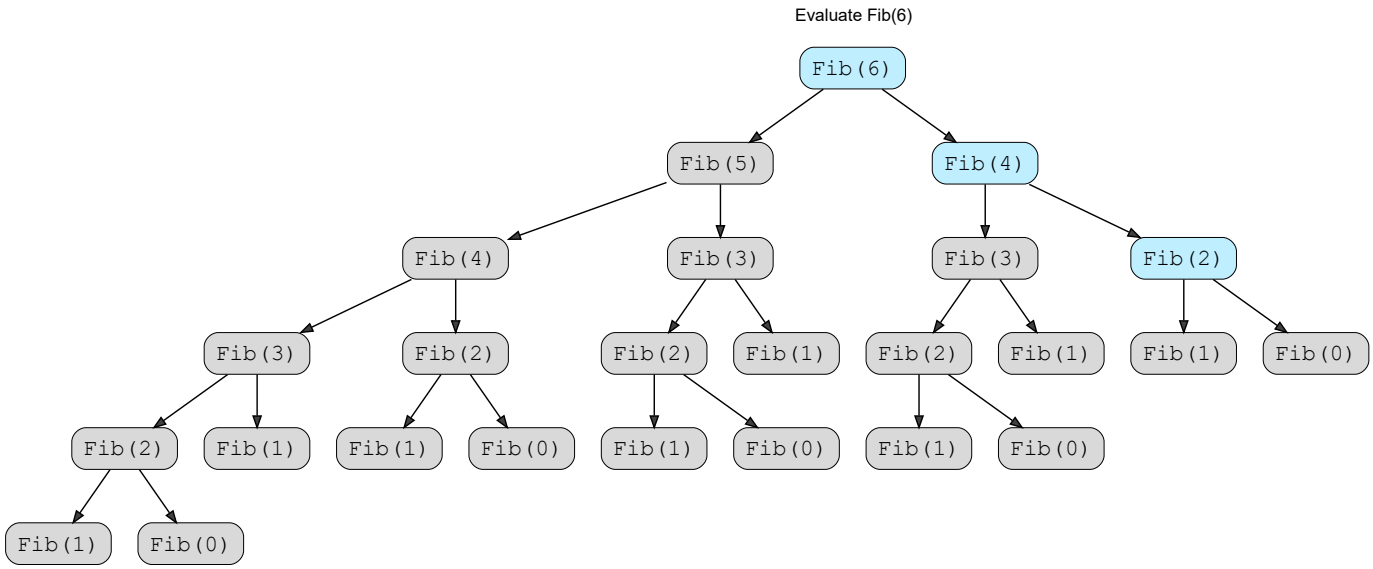

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

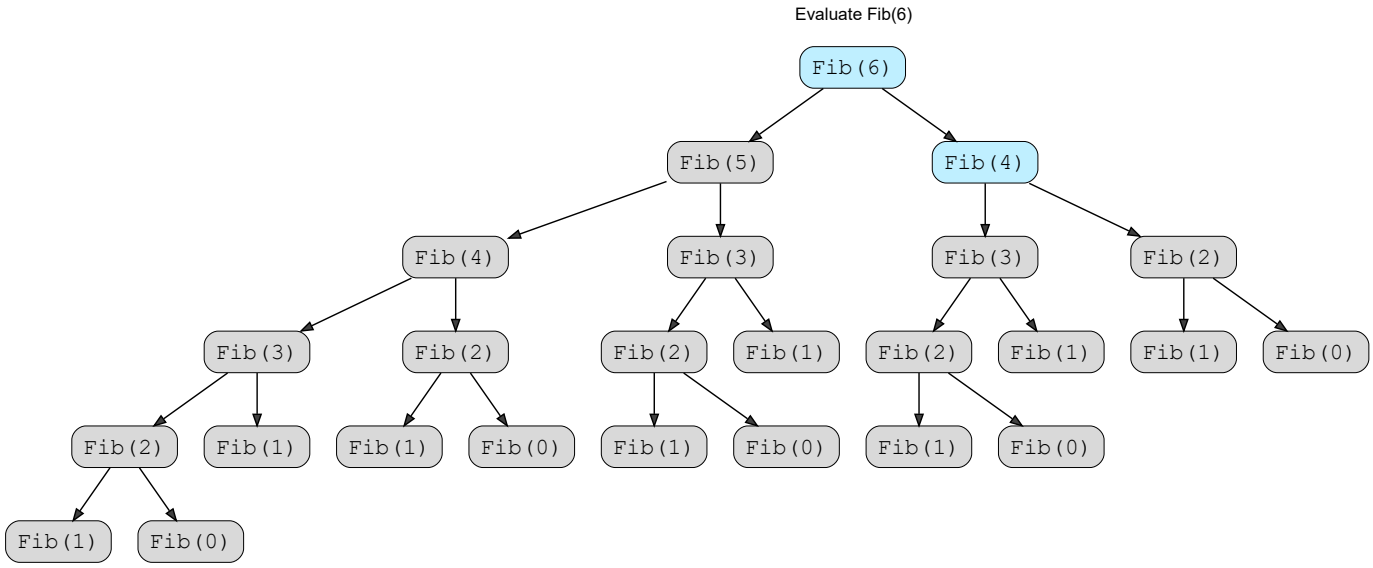

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

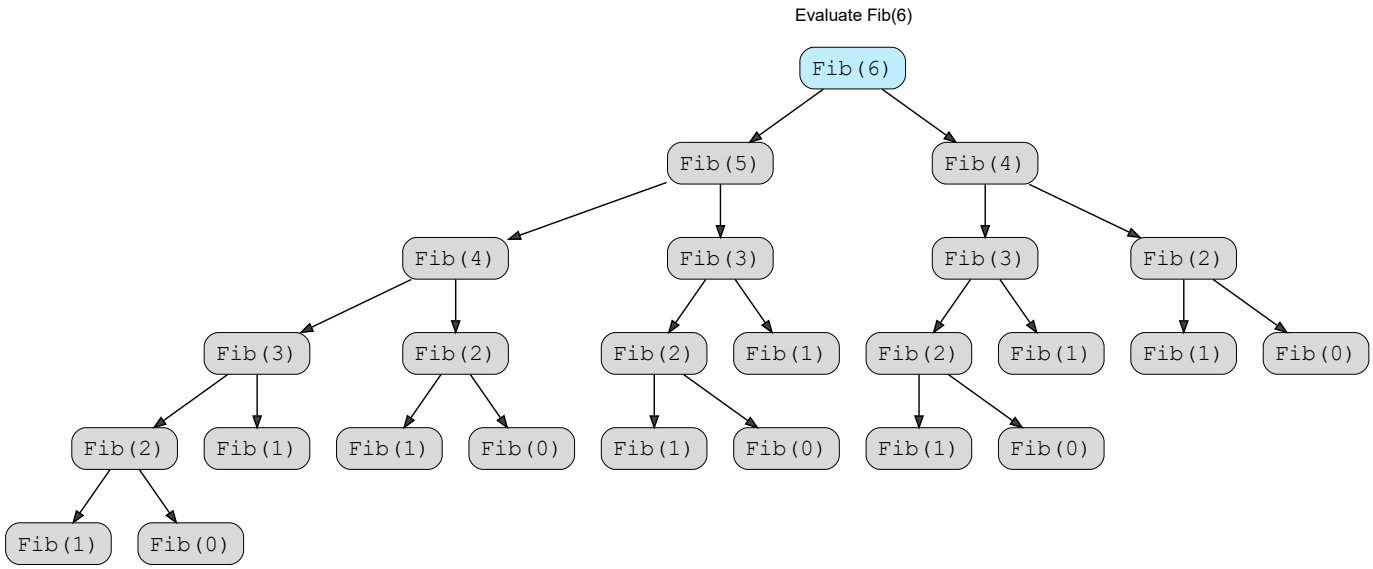

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)


Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

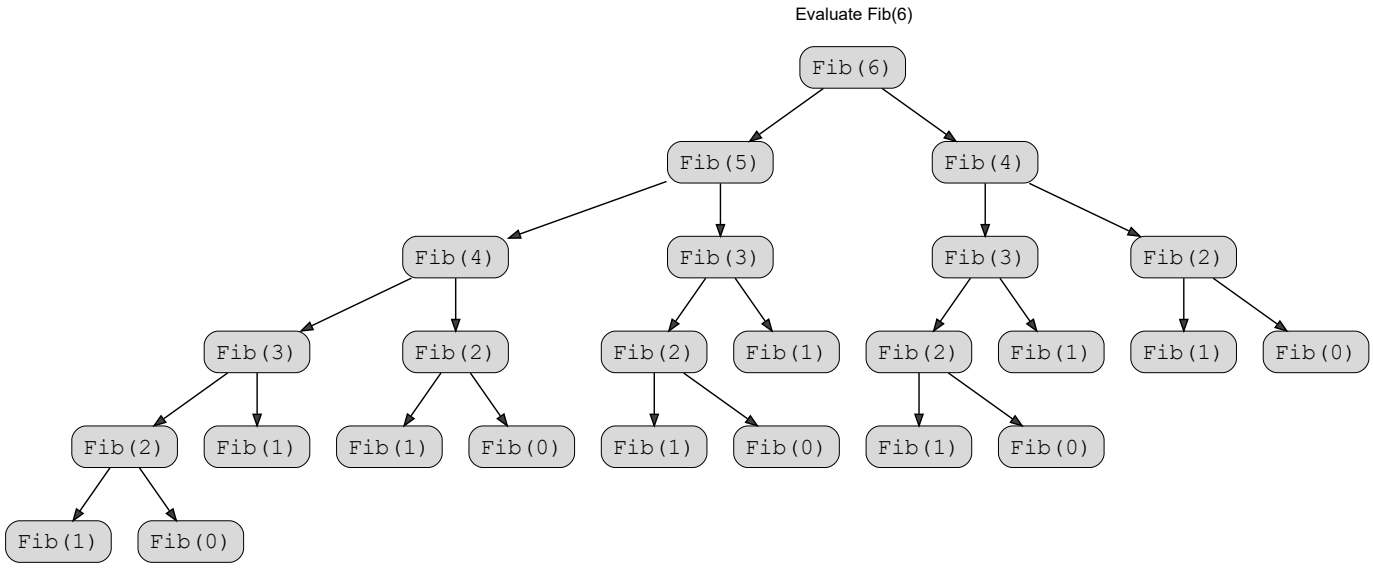

Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)


Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)


Evaluate Fib(6)
Fib(6)
Fib(5)
Fib(4)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)
Fib(1)
Fib(0)



45 of 46



46 of 46

Notice how we are recomputing numbers over and over again. For example `Fib(4)` is evaluated two times, once as part of `Fib(6)` and another time as part of `Fib(5)`. `Fib(4)`'s value is not going to change, so why do we need to recompute it? If there was a way to remember the values we had already computed as part of some other problem, we would be able to reduce our run time greatly.

Dynamic programming is the answer to all these questions. We will begin talking

Dynamic programming is the answer to all these questions. We will begin talking about dynamic programming in the next lesson.