Dynamic Programming

November 9, 2021

Introduction

- A powerful algorithm design methodology that can solve a wider range of optimization problems
- Similar to divide and conquer method, DP also solves problem by dividing the problems into subproblems and combining the solutions
- Dynamic programming applies when the subproblems overlap- that is, when they share sub-subproblems.
- When subproblems are shared - Divide an conquer algorithms do more work by computing them repeatedly.
- A dynamic-programming algorithm solves each subproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer.

Properties

DP is useful when the problem exhibits two properties.

1. Optimal substructure

   DP constructs the solution from the optimal solution of its subproblems.

2. Overlapping subproblems

   DP stores the solutions of some subproblems for resusing.

Storing solutions is called *memoization*

An example

Let us consider the recursive algorithm for computing n^{th} Fibonacci number.

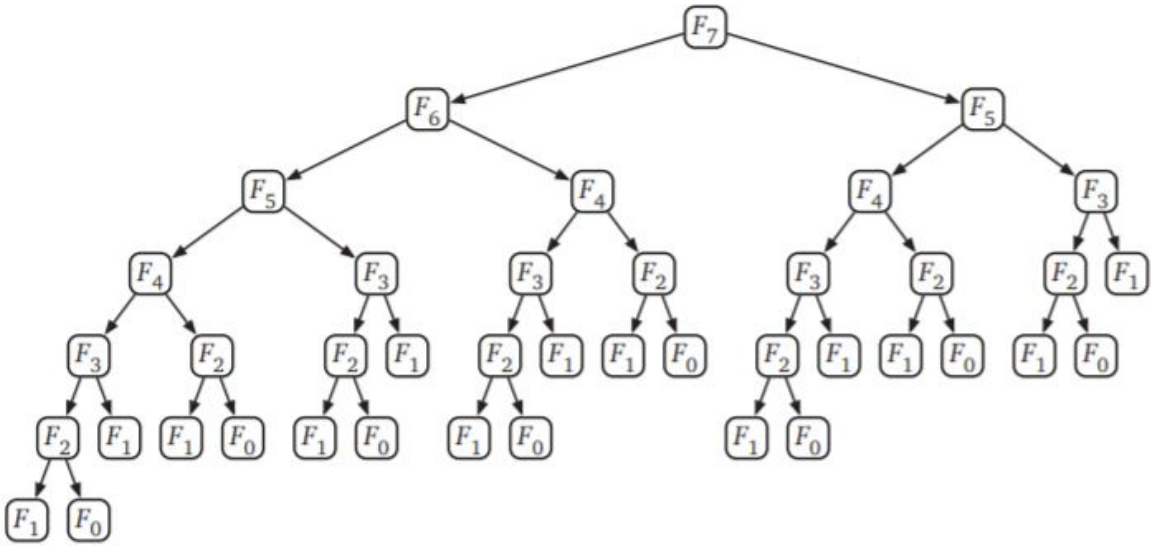
Algorithm: Recursive Fibonacci, $fib(n)$

---

if n==*0* **then**
 | return 0
else if n==*1* **then**
 | return 1
else
 | return $fib(n-1) + fib(n-2)$
end

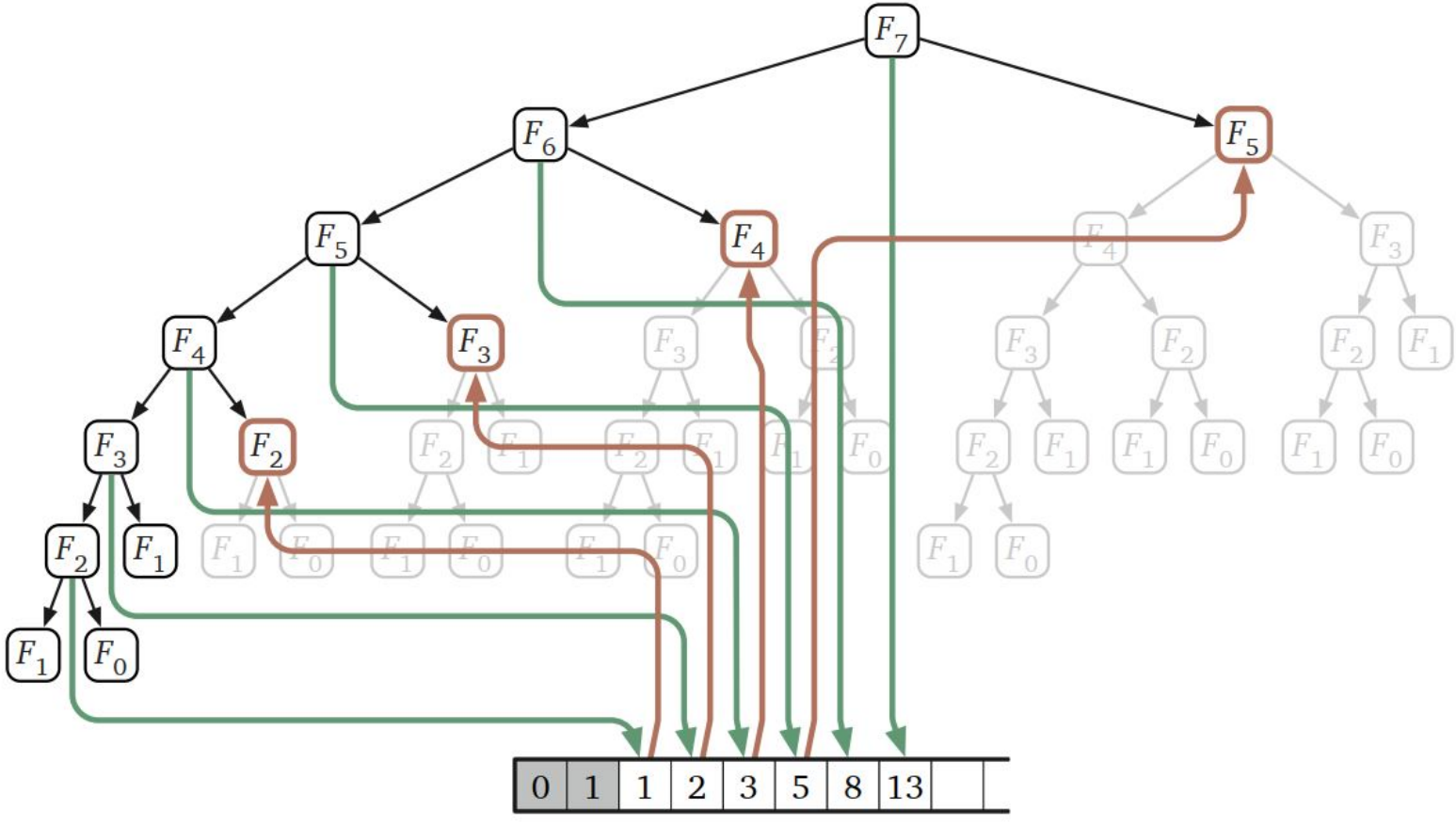
Recursive fibonacci

- Recursive fibonacci has optimal substructure property. The n^{th} fibonacci number is calculated using the $n-1^{th}$ and $n-2^{th}$ fibonacci numbers.
- Recursive fibonacci has overlapping subproblems



The nearest leaf node, F_3 on the right most branch in the figure, is at a height $n/2$. So total number of recursions = $\Omega(2^{n/2})$

Store results of overlapping subproblems



DP Fibonacci algorithm

Algorithm 2: DPFib(n)

```
F[0]=0
F[1]=1
for i = 2 to n do
  |  F[n] = F[n-1]+F[n-2]
end
return F[n]
```

DPFib uses $\theta(n)$ time and space.
We can improve the space complexity since we use only the last two fibonacii numbers to compute a new one.

Algorithm 3: DPFib2(n)

```
prev=0
curr=1
for i = 1 to n do
  |  next=curr+prev
  |  prev=curr
  |  curr=next
end
return curr
```

DPFib2 uses $\theta(n)$ time and $\mathcal{O}(1)$ space.

Two Types of DP

1. Memoization

   Topdown approach. Only those sub problems that are previously computed, is stored.

2. Tabulation

   Bottom up approach. All sub problems are solved initially and the results are stored and goes to the next higher level.