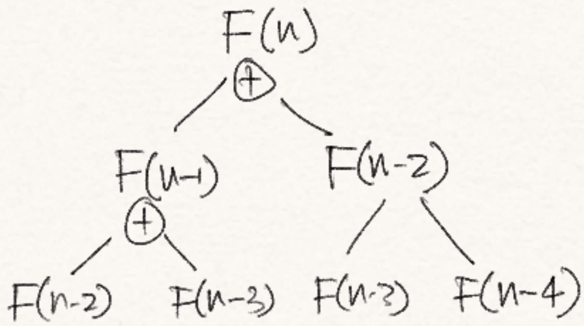
Dynamic Programming

Recall: "Divide and Conquer"

- → Find "independent" subproblems.
- → Merge the solutions obtained from subproblems and construct a solution for the whole problem.

Fibonacci Series:



$F(n) = F(n-1) + F(n-2)$.

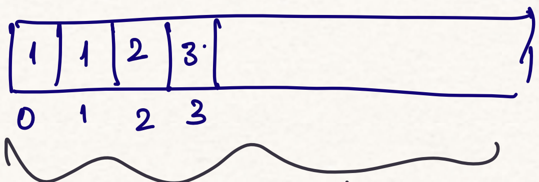
Fib(n):
<handle base cases>
return Fib(n-1) + Fib(n-2).

$\widehat{\text{Fib}}(n)$: Fib ← array of length n+1.

Fib(0) = 1
Fib(1) = 1 } array

For i in [2, n]:
Fib(i) = Fib(i-1) + Fib(i-2)

Return Fib(n).



Space to store our computations so that they can be reused.

- Divide main problem into subproblems again but they may overlap.
- Using space to help store and reuse computations } memoization.

Longest Increasing Subsequence.

$a_i \in \mathbb{Z}_+ \ \forall i$ (a_i's are distinct)

Sequence: $a_1, a_2, \ldots, a_n$

Want: length of a longest subsequence that is increasing.

$$a_{i_1} < a_{i_2} < \cdots < a_{i_k} \quad \text{and} \quad i_1 < i_2 < \cdots < i_k.$$

$\{1\}$ → $\{1\}$ $\{\}$

$\{1,2\}$ → $\{1\}$ $\{\}$ $\{1,2\}$ $\{2\}$

3 5 6 1 2 8 4 7

3568   3567   1 2 4 7 | 4

Length of

1. Longest increasing subsequences with a_n in them
2. " " " without a_n

max

$\hookrightarrow$ LLIS(A[1,…,n-1]).

We need to look for increasing subseq. where all element in it are $< a_n$.

$\hookrightarrow$ LLIS_smaller (A[1,…,n-1], a_n)

returns the maximum over the lengths of all incr. subseq. whose elements are all $< a_n$.

- LLIS_smaller (A[1,…,i], x):

  i = 1/0 <do something>.

  if $a_i > x$:

  m = LLIS_smaller(A[1,…,i-1], x).

  else:

$$m = \max \{ \text{LLIS\_smaller}(A[1,\dots,i-1], a_i) + 1$$
$$\text{LLIS\_smaller}(A[1,\dots,i-1], x) \}$$

return m.

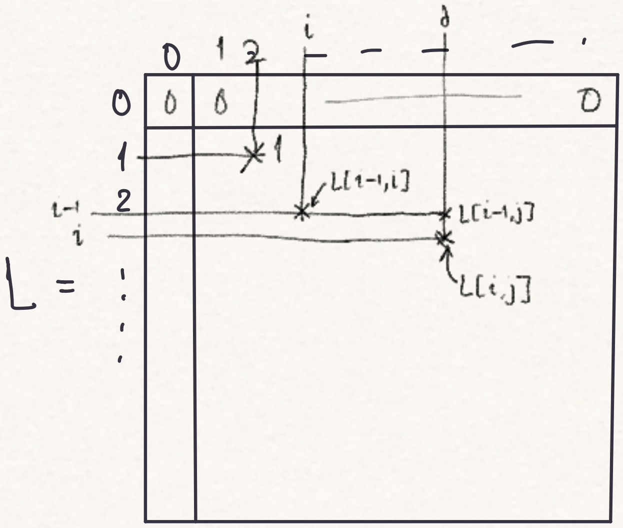
- LLIS $(A[1,\dots,n])$:

  max + 1.

  return LLIS_smaller$(A[1,\dots,n], \infty)$

Subproblems: LLIS_smaller$(A[1,\dots,i], a_j)$ $\quad i<j$

$\hookrightarrow L[i,j]$



$$L[0,j] = 0$$

$$L[i,j] = \begin{cases} L[i-1,j] & \text{if } a_i > a_j \\ \max\{L[i-1,j],\ 1 + L[i-1,i]\} & \text{otherwise} \end{cases}$$