Dynamic Programming

$$F_n = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ F_{n-1}+F_{n-2} & \text{if } n>1 \end{cases}$$

$F_0 = 0$ $\quad F_4 = 3$
$F_1 = 1$ $\quad F_5 = 5$
$F_2 = 1$ $\quad F_6 = 8$
$F_3 = 2$ $\quad F_7 = 13$
$\vdots$

Fibo(n):
  if $n=0$: return 0
  if $n=1$: return 1
  if $n>1$:
    return Fibo(n-1)
      + Fibo(n-2)

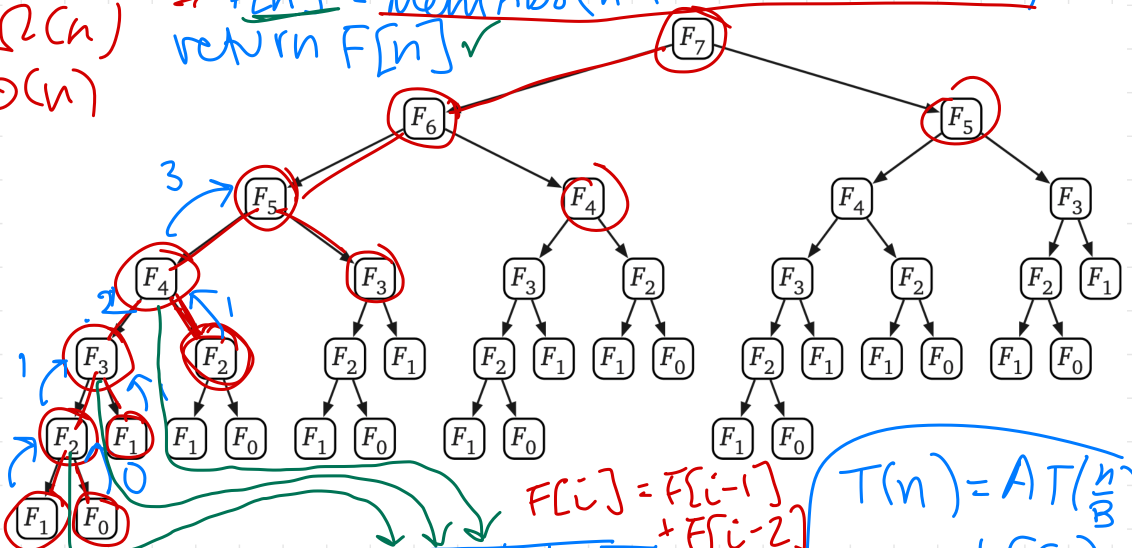
$\rightarrow$

$T(n) = T(n-1)+T(n-2)+1$
"runtime of Fibo on input of n"
$T(1)=1$ $\quad T(n)=O(2^n)$
$T(0)=1$

idea: Memoization

MemFibo(n): $\quad n=1$
  if $n=0$: return 0
  if $n=1$: return 1
  if $n>1$:
    if F[n] is unfilled:
      $\rightarrow$ F[n] = MemFibo(n-1) + MemFibo(n-2) ✓
    return F[n] ✓

$\Omega(n)$
$O(n)$



$F[i] = F[i-1] + F[i-2]$

$T(n) = A\,T(\frac{n}{B}) + f(n)$

F = | | | 1 | 2 | 3 | | | |
0 1 2 3 4 5 6 7

To write a DP alg:

1. Write the english def. of the subproblem we will solve.
2. write the recursive def. of that subproblem.
3. Figure out how to memoize
4. Write our DP alg

example: Max Candy

input: list of amnt of candy that each house gives out

output: max amnt of candy you can get by trick or treating, but you can't go to two houses in a row.

$$C = [\overset{1}{3}, \overset{2}{5}, \overset{3}{1}, \overset{4}{2}, \overset{5}{6}, \overset{6}{4}, \overset{7}{6}] \qquad n = 7$$

output: 17

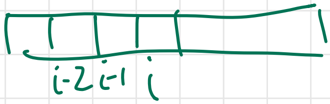
1. Let $MC(i)$ be the max amount of candy you can get trick or treating up to house i.

$MC(1) = 1 = C[1]$ t

$MC(2) = \max(C[1], C[2]) = \max(3, 5) = 5$

$MC(7) = 17 =$ the final answer

② Write the recursive def. of the subprob.

C = 

MC(i) is either:

→ • candy at house i + max candy up to house i-2

→ • max candy ~~from~~ up to house i-1

$$MC(i) = \begin{cases} C[1] & \text{if } n=1 \\ \max(C[1], C(2)) & \text{if } n=2 \\ \max(C[i]+MC(i-2), MC(i-1)) & \text{if } n>2 \end{cases}$$

③ Memoize: store MC(i) values in an array M which we fill for increasing i.

④ Write the DP alg.

```
MaxCandy(array C of candy amnts):
    M[1] = C[1]
    M[2] = max(C[1], C[2])
    for i in 3 to n:
        M[i] = max(C[i]+M[i-2], M[i-1])
```