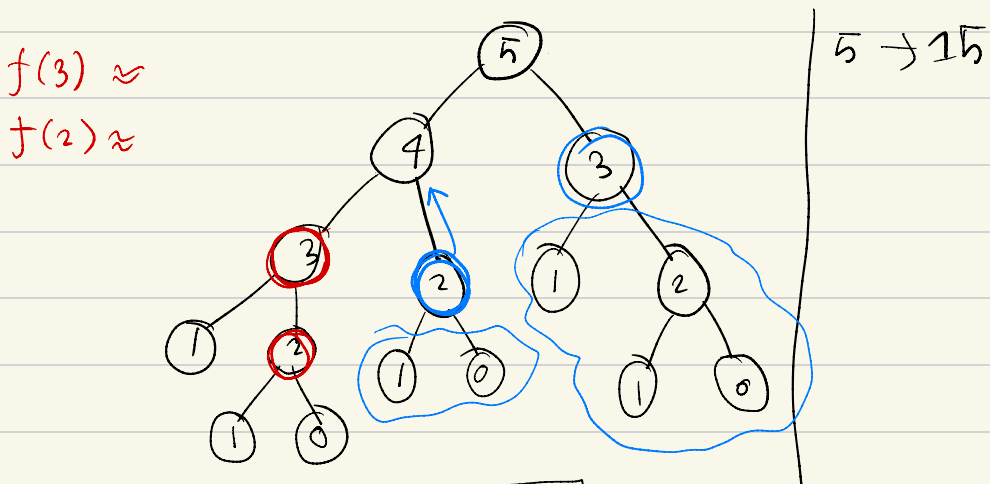
Dynamic Programming:

https://atcoder.jp/contests/dp/tasks

$f(n) = f(n-1) + f(n-2)$, $f(0) = 0$, $f(1) = 1$

```
f(n):
    if n ≤ 1: return n
    else return f(n-1) + f(n-2)
```



$\rightarrow T(n) = T(n-1) + T(n-2) + 1$

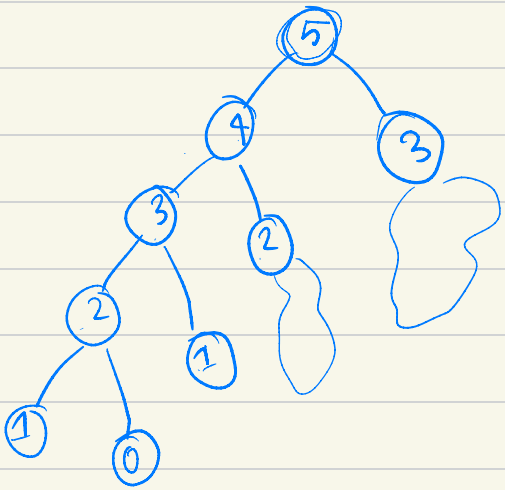
$= T(n-1) + T(n-1) + 1$

$= 2T(n-1) + 1$

$= (2 \cdot 2 \cdot T(n-2)) + (1 + 1 + 1 \cdots)$

$= 2^n + n$

$O(2^n) \rightarrow$



$\approx 2 \cdot n$

$O(n)$

→ linear.

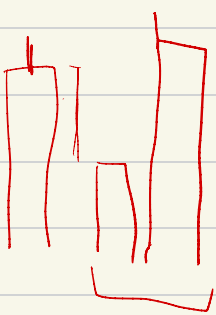
→ $f[0] = 0,\ f[1] = 1$

$f[i] = -1$

```
call(n):
    if n <= 1: return n
    if f[n] != -1: return f[n]
    return f[n] = call(n-1) + call(n-2)
```

Problem 1: Jumping frogs:

$h_i \to h_{i+1}$ / h_{i+2} | if you jump from h_i to h_j then cost is $|h_i - h_j|$



- What is the base case?
- What is the recursion?

$f(n)$

$f(1) = 0 \to$

$f(i)$

$\min(f(i-1) + |h_i - h_{i-1}|,$
$f(i-2) + |h_i - h_{i-2}|)$

1 ... i-1

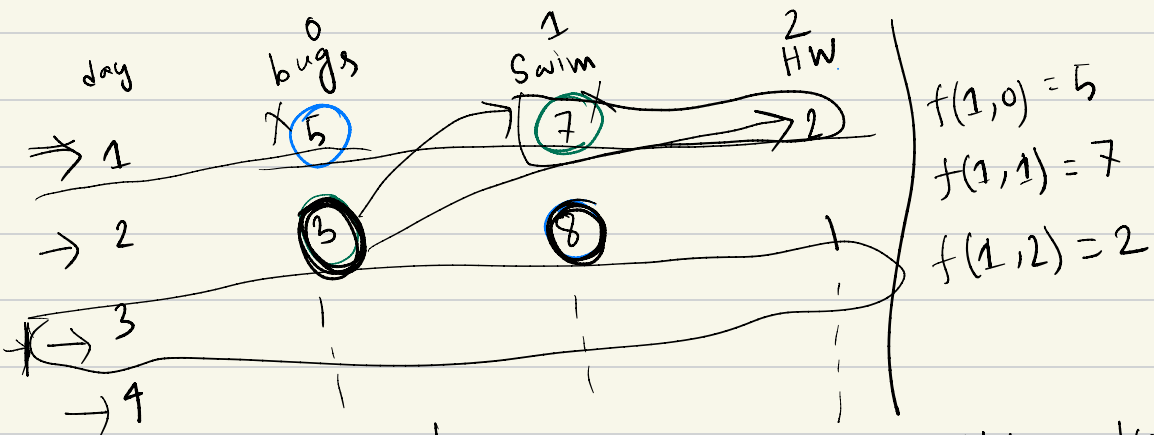
(n, m, k) → (n', m', k')

dp[ ][ ][ ]

problem 3: Vacation

n days of vacation:

H → c_1	c_2	c_3		
S → a_1	a_2	a_3	- - - -	
b → b_1	b_2	b_3		
1	2	3	4	5



f(3, K)

f(2,0) = inp(2,0) + max(f(1,1), f(1,2))

f(2,1) = ... + f(1,0), f(1,2)

f(2,3) = ...

max (f(n,0), f(n,1), f(n,2))

call (n, i):

n == 1 : dp[n][i] = ~~~

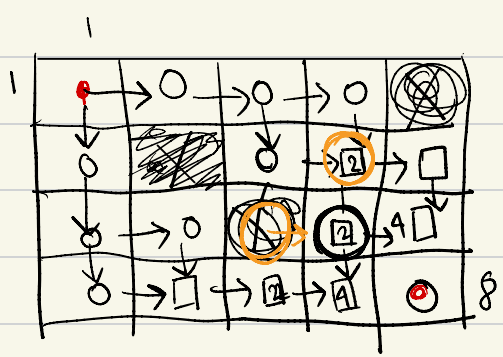
if dp[n][i] ≠ -1 :

for j → 0, 2

j == i :

return ~ ~ ~ ~

Problem : Grid 1



(1, 1) → 1

(1, 2) → 1

dp(1,1) = 1

$$dp(i,j) = dp(i-1,j) + dp(i,j-1)$$