S. KIMMEL

Recurrence Relations

Algorithm with loops → use summation/graph analysis to analyze runtime

Recursive algorithm → use recurrence relations

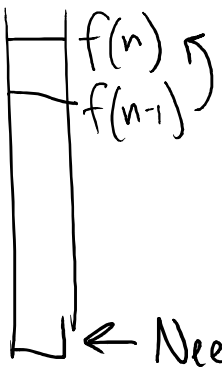
Recurrence: a function that calls itself

recurrence

$f: \mathbb{N} \to \mathbb{N}$

base case

$$f(n) = 2f(n-1) + 1 \quad ; \quad f(1) = 1$$



$f(1) = 1$

$f(2) =$ (use recurrence)

$= 2f(1) + 1$

$=$ (use base case)

$= 2 + 1 = 3$

$f(3) = 2 \cdot f(2) + 1$

$= 2 \cdot 3 + 1 = 7$

$\vdots$

S.KIMMEL

Q: What is a recurrence relation for the runtime of this algorithm?

```
Thing(n)
if n=1 or n=2: return n
for i = 1 to n^2
    Print: "meh"
return Thing(n-1) + Thing(n-2)
```

A) $T(1) = O(1)$; $T(n) = T(n-1) * T(n-2) + O(n^2)$

B) $T(1) = O(n)$; $T(2) = O(1)$; $T(n) = T(n-1) * T(n-2) + O(n^2)$

C) $T(1) = O(1)$; $T(n) = T(n-1) + T(n-2) + O(n^2)$

D) $T(1) = O(1)$; $T(2) = O(1)$; $T(n) = T(n-1) + T(n-2) + O(n^2)$

S. KIMMEL

Q: What is a recurrence relation for the runtime of this algorithm?

```
Thing(n)
if n=1 or n=2: return n          ← O(1)
for i = 1 to n^2                 ] ← O(n^2)
    Print: "meh"
return Thing(n-1) * Thing(n-2)   ← T(n-1) + T(n-2) + O(1)
```

multiplication, return

A) $T(1) = O(1)$; $T(n) = T(n-1) * T(n-2) + O(n^2)$

B) $T(1) = O(n)$; $T(n) = T(n-1) * T(n-2) + O(n^2)$

C) $T(1) = O(1)$; $T(n) = T(n-1) + T(n-2) + O(n^2)$

D) $T(1) = O(1)$; $T(2) = O(1)$; $T(n) = T(n-1) + T(n-2) + O(n^2)$

need 2 base cases because otherwise fall off the bottom of ladder

If only one base case

- $T(1) = O(1)$
- $T(2) = T(1) + T(0) + O(2^2)$

Fall off bottom of ladder