moony 05

Goal :

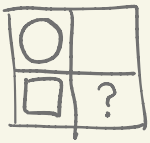
$1+2+3+\ldots+n$

$= \frac{n}{2}(n+1)$

$= \frac{n^2}{2} + \frac{n}{2}$

$= O(n^2)$

moony_1 : (?)

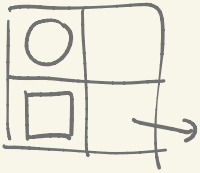


Wishful thinking

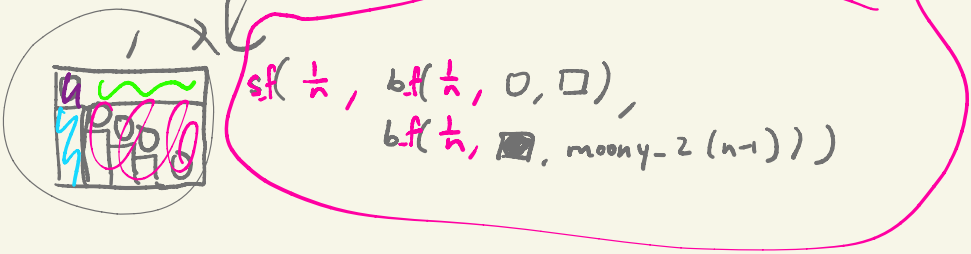
↳ base case

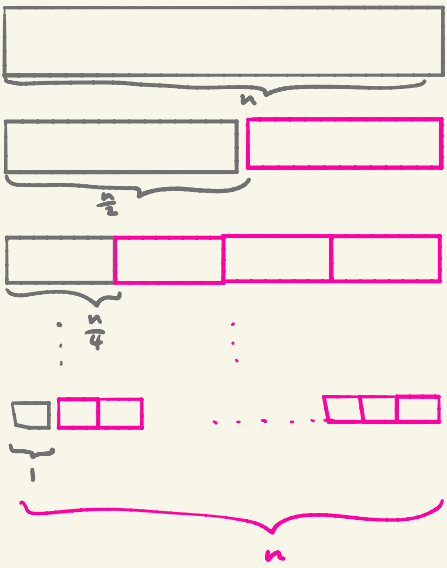
↳ Inductive step:
$f(n-1) \rightarrow f(n)$

moony_2 : (n)



moony_3 :




n
n/2
n/4
1
n

```
fn cone(n, rune) {
    fn cone_helper(n, rune, max) {
        ↳ n === 1
            ? rune
            : overlay_f( 1/n, 1 - (n-1)/max
                         scale(1/n, rune),
                         cone_helper(n-1, rune, max) );
    }
    ↳ cone_helper(n, rune, n);
}
```

$\frac{1}{4}$ $\frac{1}{3}$ $\frac{1}{2}$ $\frac{1}{1}$

↳ $\frac{1}{4}$ $\frac{2}{4}$ $\frac{3}{4}$ $\frac{4}{4}$

f(n)

↓ beside_f, ...

f(n-1)

① Base case (n = 1)

② Inductive step

cone(n) {

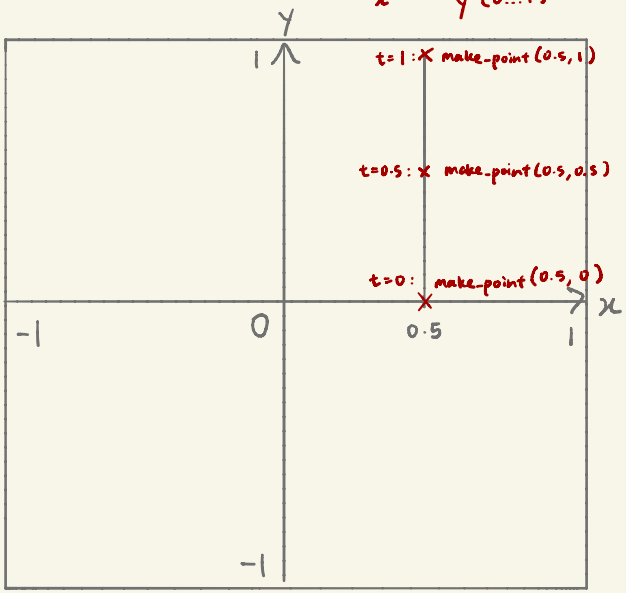
↳ n === 1

? 0

: cone(n-1)

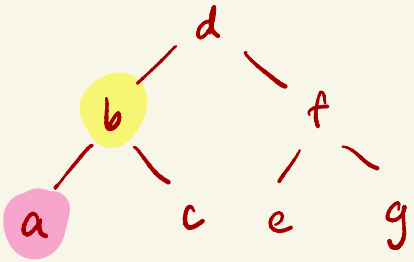
t ⇒ make_point (0.5, t)

x ↓ y (0...1) ↓



p(b, p(b, p(b, p(b, p(b, 0)))))

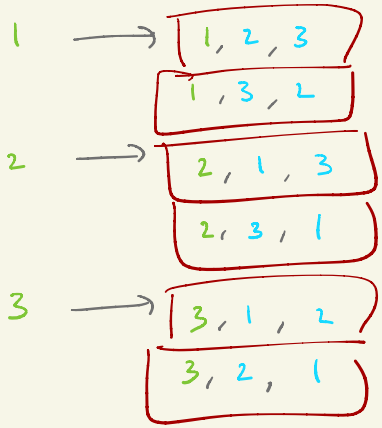
T(n) = nT



permutations (lst)

list (1, 2, 3):

① Wishful thinking. For each element (↗ x), assume we have a solution for the list minus that element → remove (x, lst)



② Wishful thinking:

```
permutations ( remove (x, lst) )
```

③ Map list in wishful thinking solution

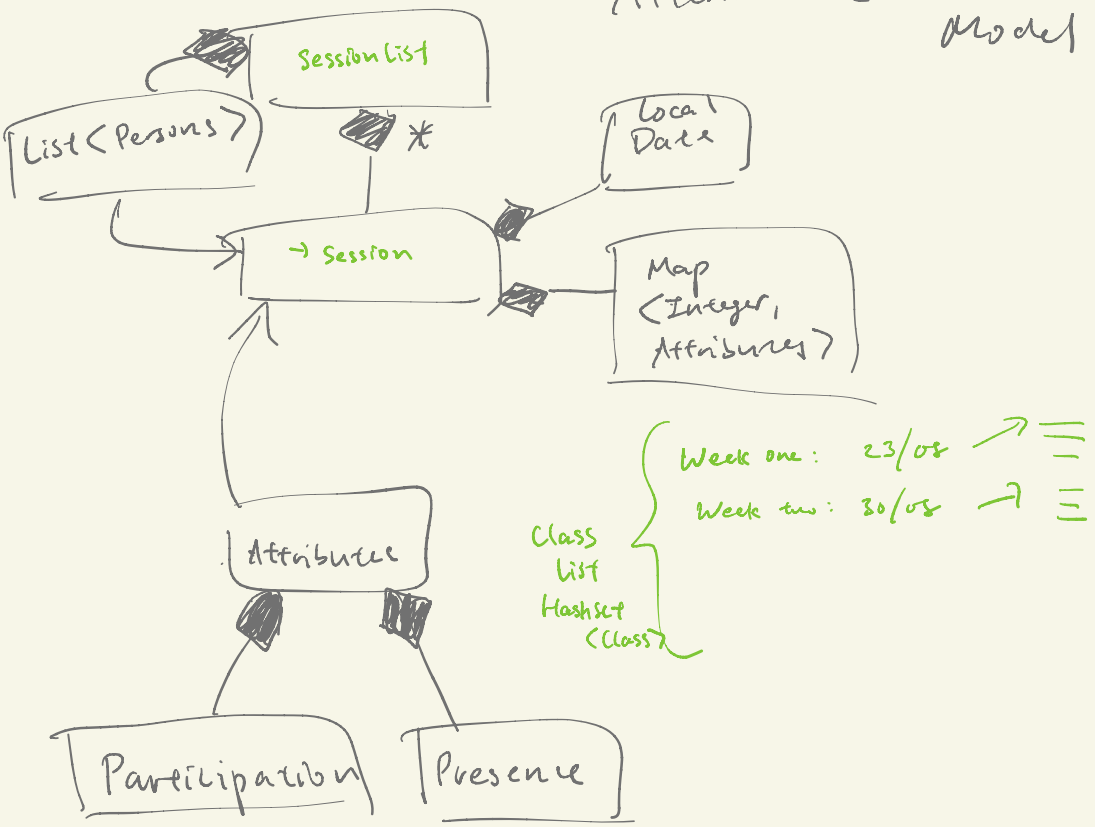
```
map ( p ⇒ pair (x, p),
      permutations ( remove (x, lst) ) )
```

④ Map element in lst

```
map ( x ⇒ map ( p ⇒ pair (x, p),
                permutations (remove (x, lst)) ),
      lst )
```

⑤ Final step: combine solutions using accumulate (append, null, ...)

Attendance Model



chain(twice, 3)

→ twice(twice)(twice)

→ four_times(twice)

→ four_times(ff)

→ three_times(ffff)

→ two_times(ff...f) (32 f's)

→ ff...f (16 f's) = $2^{2^4} = 32$

$t(b, n-1) : \Theta(b^{n-1})$

$t(b, n-2) : \Theta(b^{n-2}) < \Theta(b^{n-1})$

$t(b, n-3) : \Theta(b^{n-3}) < \Theta(b^{n-1})$

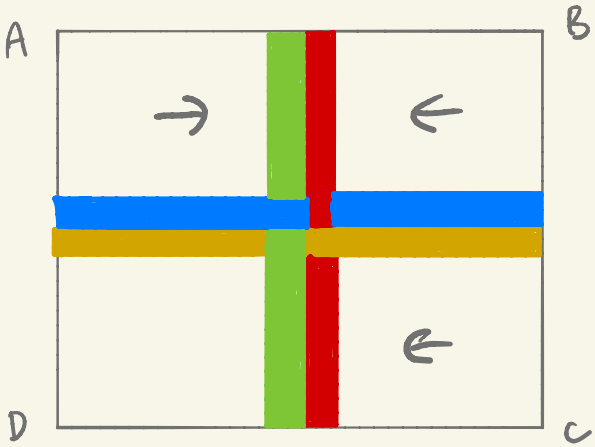
⋮

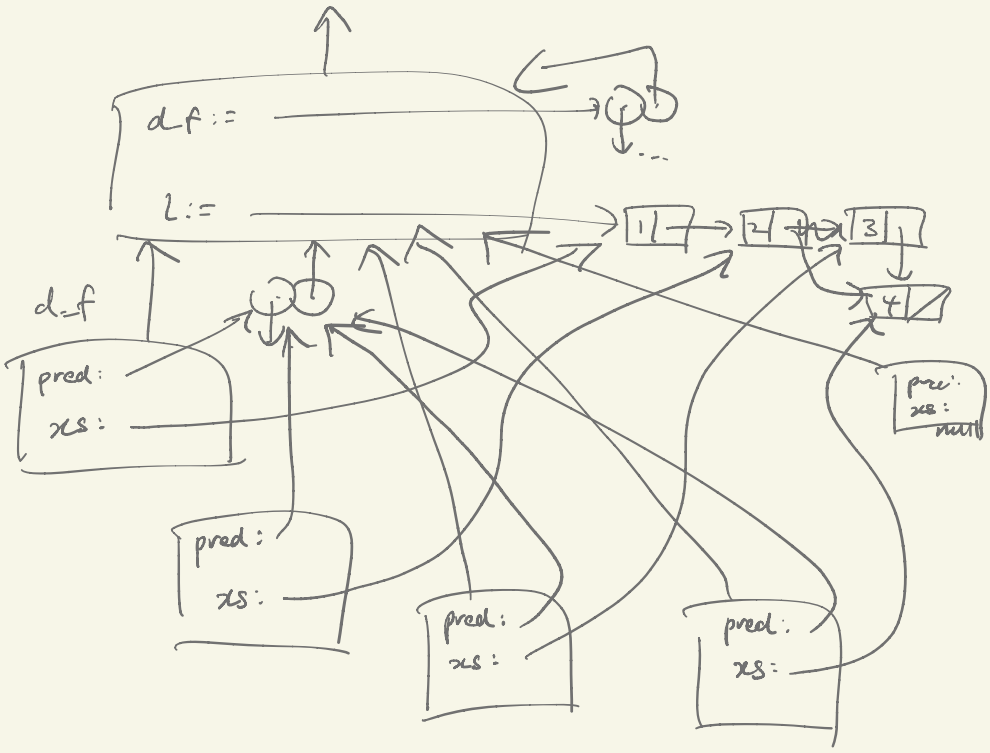
$t(b, 1) : \Theta(1) < \Theta(b^{n-1})$

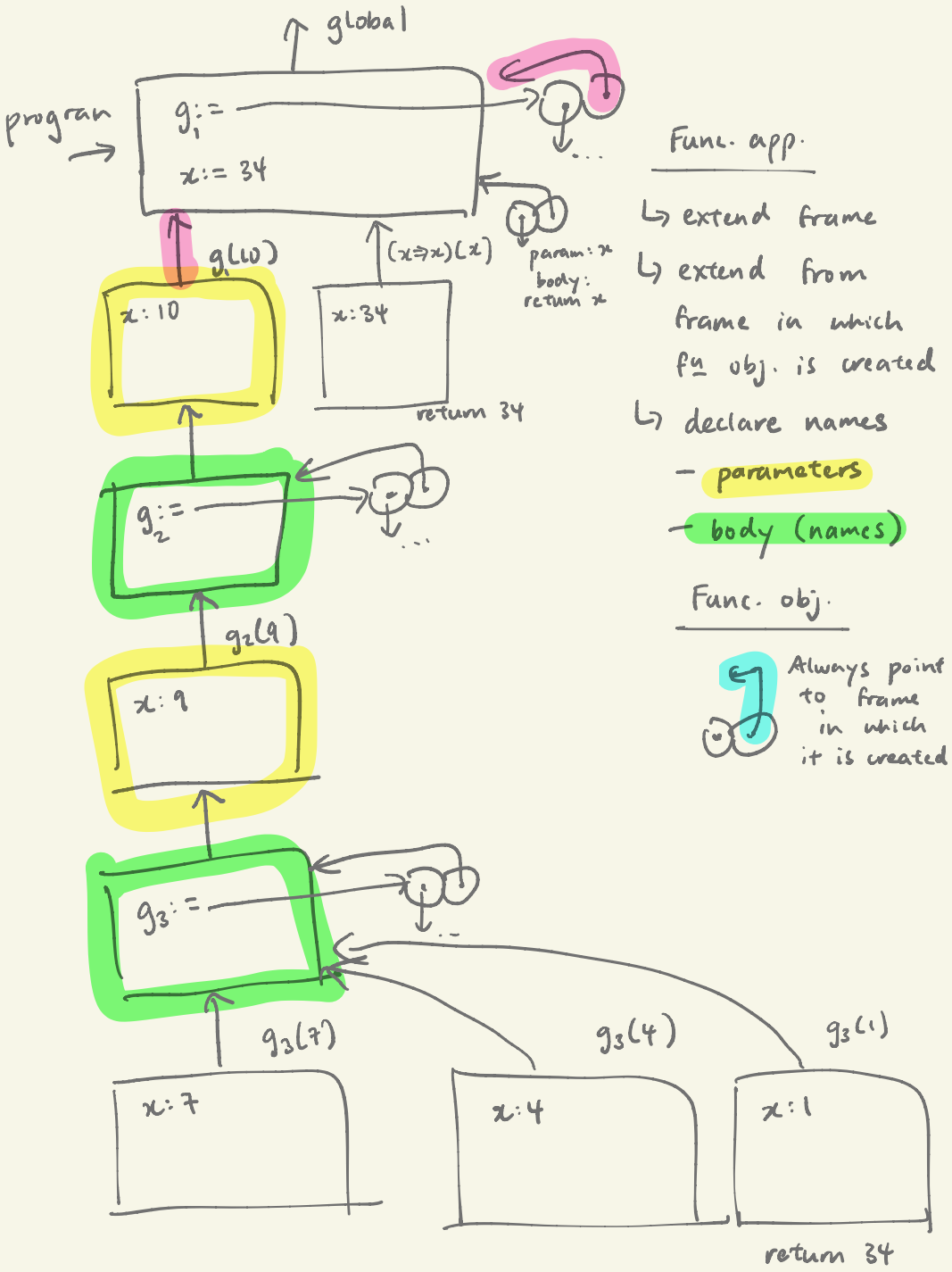
} n

- IIII IIII
  II IIII

$2^7 - 1$


A
B
D
C


d_f :=
l :=
1
2
3
4
d_f
pred :
xs :
pred :
xs :
pred :
xs :
pred :
xs :
pre :
xs :
null



Func. app.

↳ extend frame

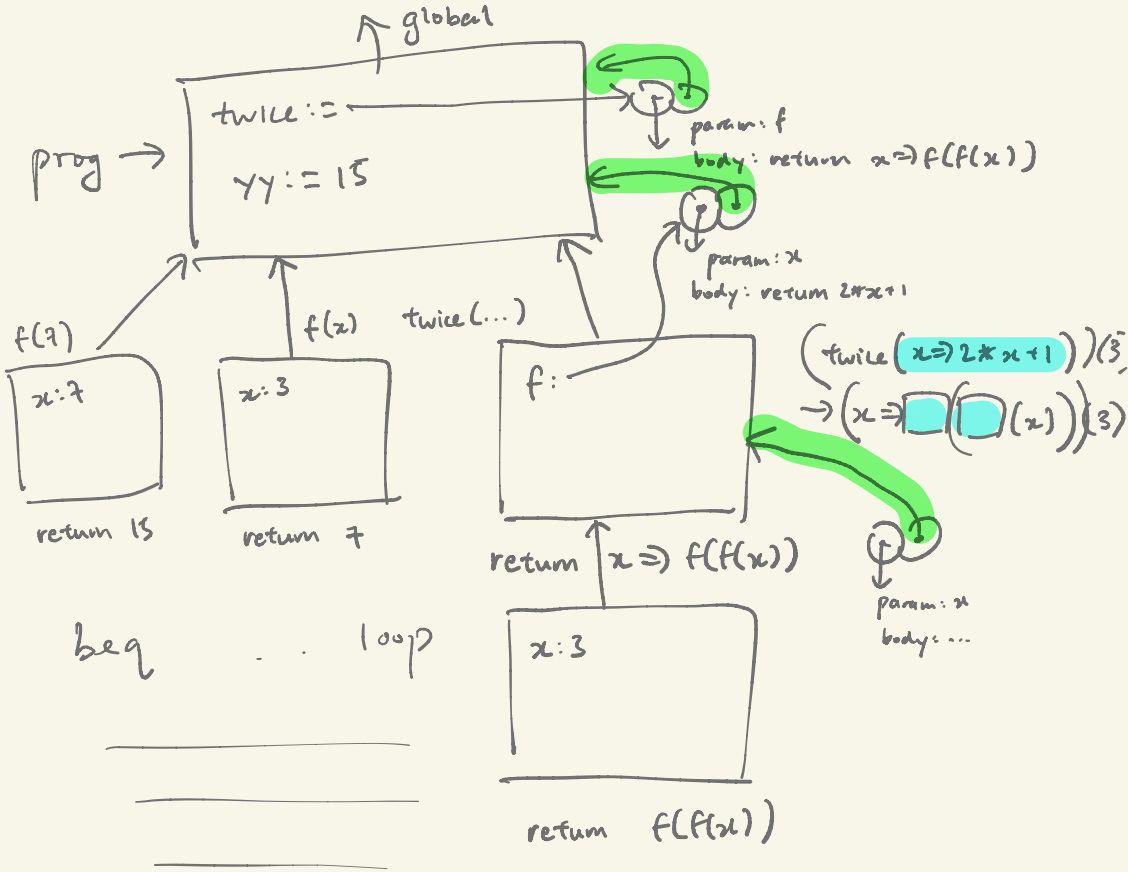
↳ extend from frame in which fn obj. is created

↳ declare names

- parameters
- body (names)

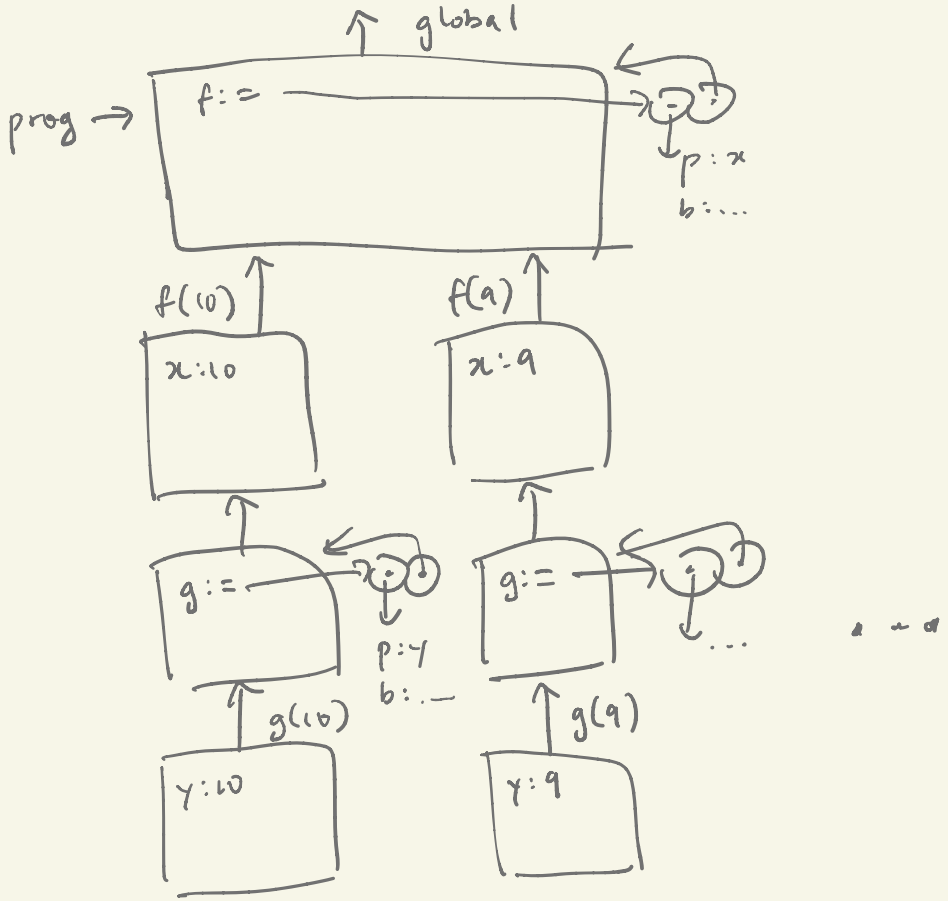
Func. obj.

Always point to frame in which it is created



beq . . loop

loop:


global
prog
f:=
p:x
b:...
f(10)
x:10
f(9)
x:9
g:=
p:y
b:...
g(10)
y:10
g:=
...
g(9)
y:9