Time Complexity and Insertion Sort

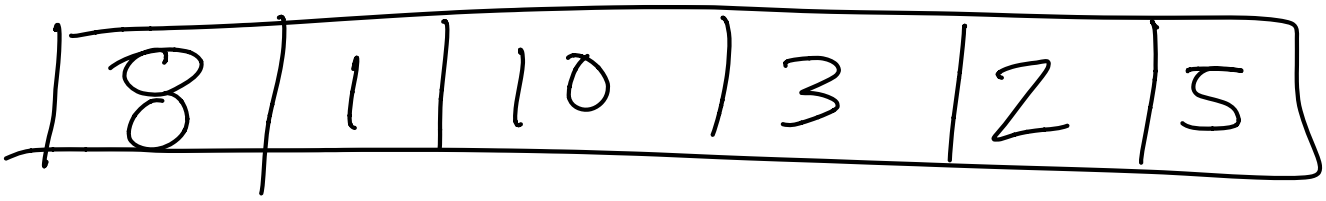
Insertion sort

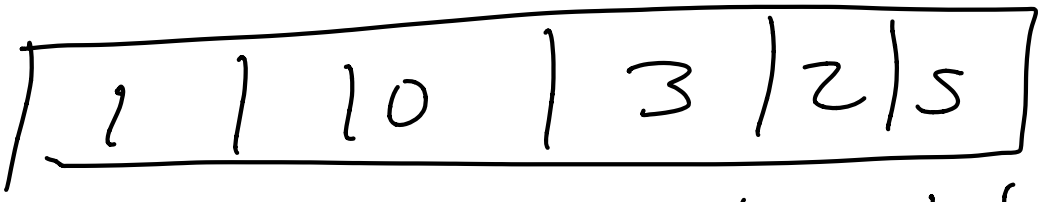
You have a list you want to sort the list.

1. Remove the first element jot the list

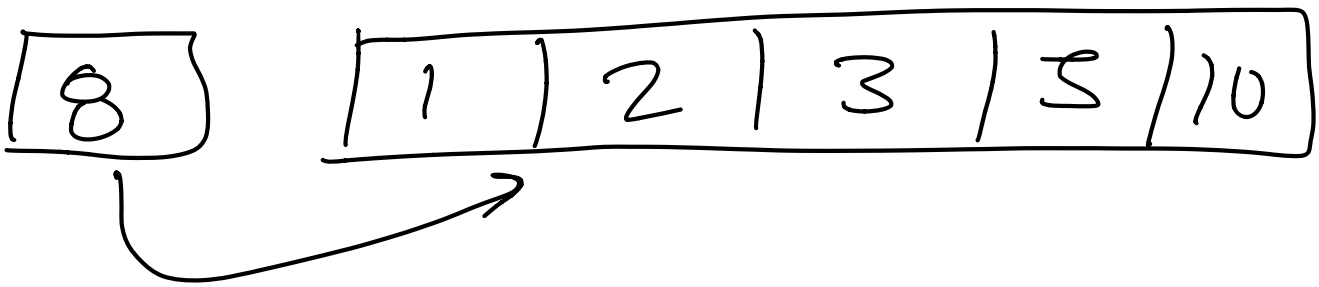
2. You recursively sort the remaining elements.

3. You insert the first back in to list.

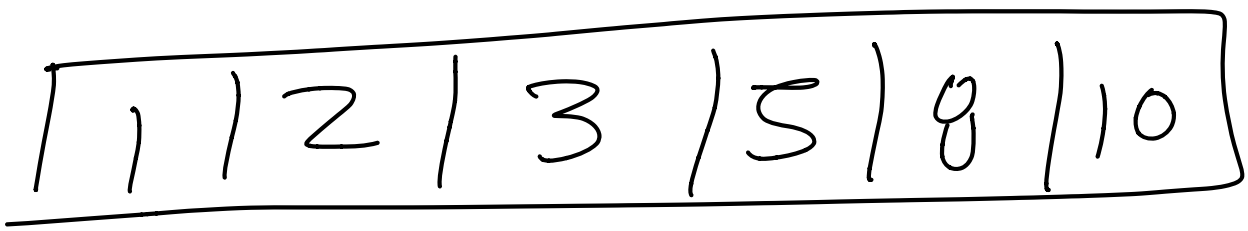




recursion will sort it



Insert 8 in to the list in the right position



```
(: insertion-sort : (Listof Integer)
                    -> (Listof Integer))
(define (insertion-sort my-list)
  (cond
    [ (empty? my-list) empty]

    [else (insert  (first my-list)
          (insertion-sort  (rest my-list)))]
                    )
```

Recursively sorted list

Insert function

input 1 : Integer

input 2 : Listof Integer.
(Sorted)

if input 1 $\leq$ (first input 2)

input 1 → 3
input 2 → [5, 6, 10, 13]

[3, 5, 6, 10, 13]

(cons input1 input 2)

Else

input 1 > (first input2)

input 1 → 11
input 2 → [5, 6, 10, 13]

Remove the first element from

input 2.

Insert 11 → [6, 10, 13]

↓

[6, 10, 11, 13]

Put back the first element.

5 → [6, 10, 11, 13]

[5, 6, 10, 11, 13]

(cons (first input2)

(insert input1 (rest input2)))

Base case

what if input2 is empty2.

(list input1)

```
(: insert : Integer (Listof Integer)
   → (Listof Integer))
```

input1 input2

```
(define (insert x mylist)
  (cond [(empty? mylist) (list x)]
        [else (if (<= x (first mylist))
                  (cons x my-list)
                  (cons (first mylist)
                        (insert x (rest my-list))))]))
```

1 op ← 1 op ← 1 op ← 1 op

1 op ← 1 op

1 op + 1 op

1 op

Time complexity of insert. $S(n-1)$

Time complexity of insert by $S(n)$

Step 0:

insert < x integer

mylist List of int

↓ size of this list as your input size: n.

Step 1: Write down the recurrence.

$$S(n) = 9 + S(n-1)$$

$$S(n) = C + S(n-1)$$

$$S(0) = 3$$

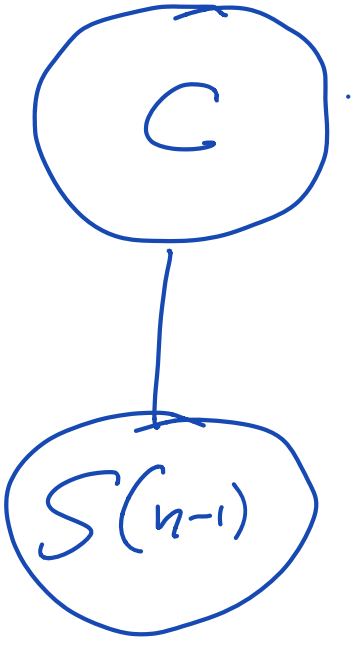
Recurrence Relation will be.

$$S(n) = C + S(n-1)$$
$$S(0) = C$$

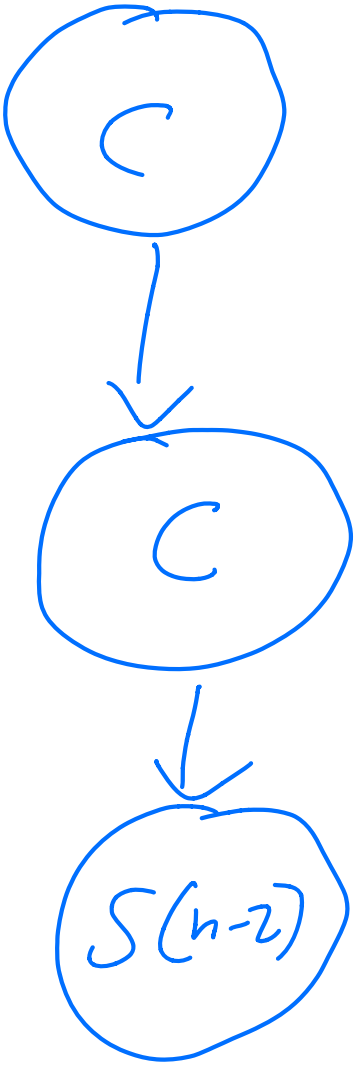
Step 2: Solve the Recurrence.

Tree method

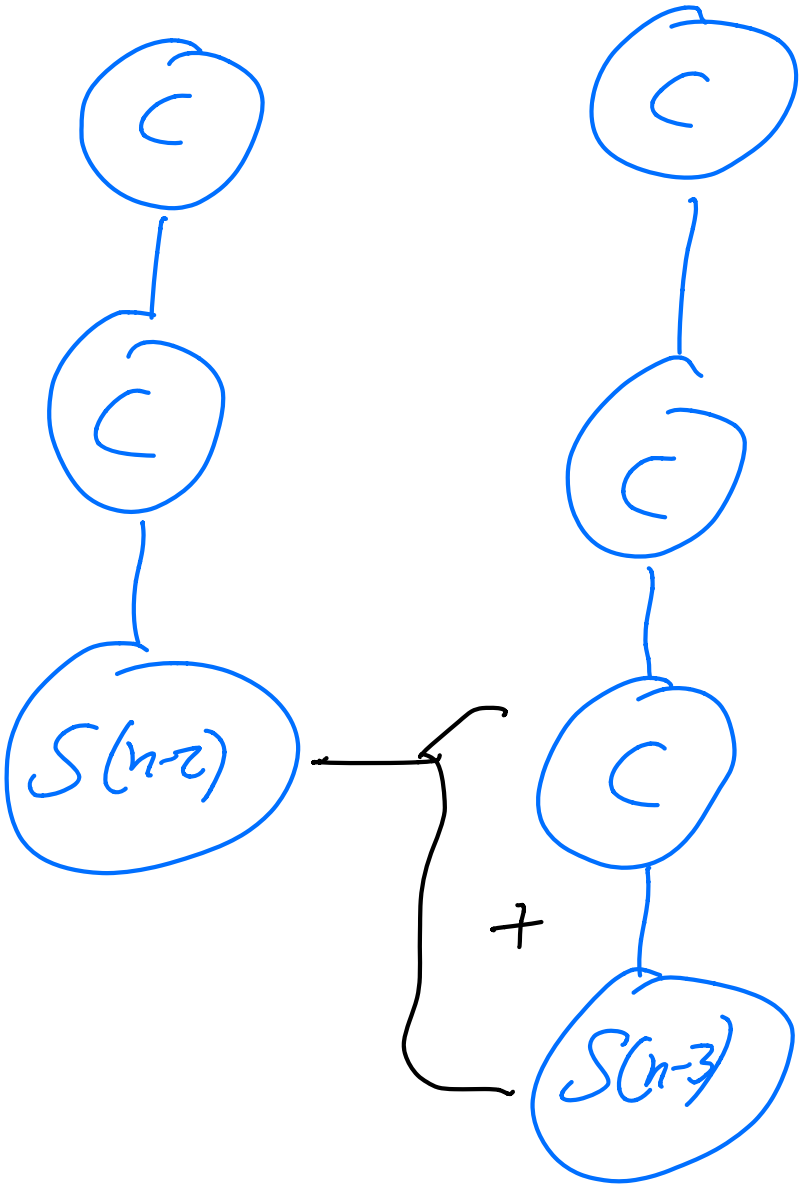
$$S(n) = C + S(n-1)$$

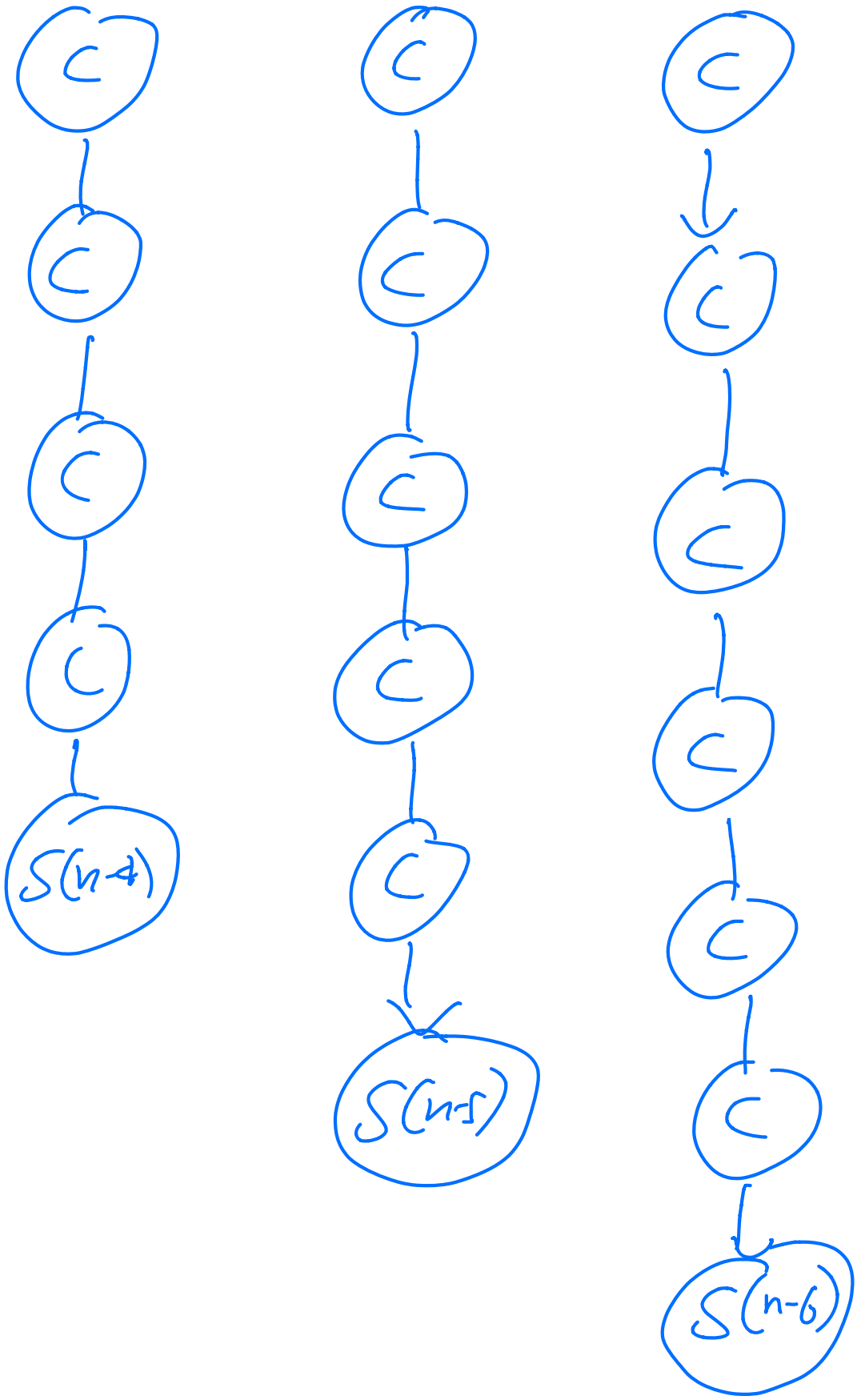


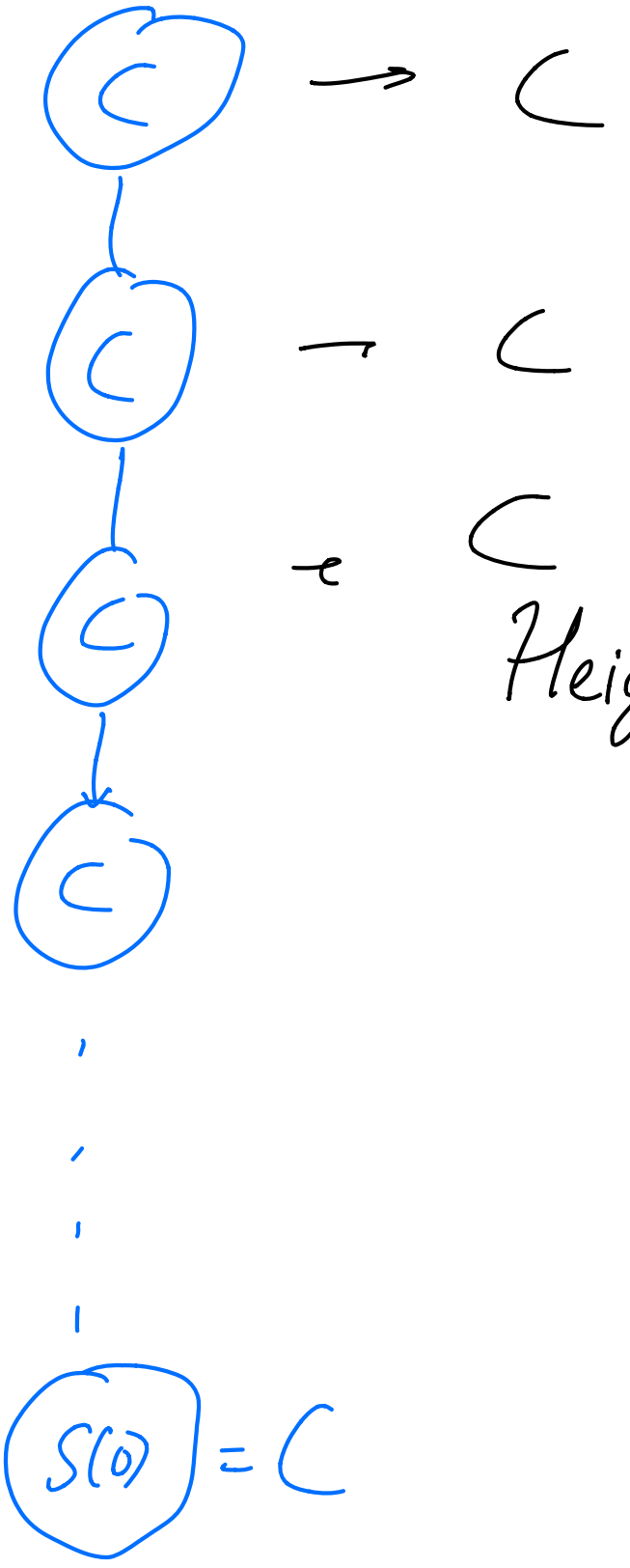
$S(n-1) = C + S(n-2)$



$S(n-2) = C + S(n-3)$


C
C
C
C
C
S(n-2)
+
S(n-3)


C
C
C
C
S(n-4)
C
C
C
C
C
S(n-5)
C
C
C
C
C
C
C
S(n-6)



Height of this = n+1

$$\frac{}{(n+1)\,C}$$

$$P(n) = (n+1)\,C$$

$$= Cn + C$$
$$= O(n)$$

$$S(n) = O(n)$$

```
(: insertion-sort : (Listof Integer)
                    → (Listof Integer))
(define (insertion-sort my-list)
  (cond [(empty? my-list) empty]
        [else (insert (first my-list)
                      (insertion-sort (rest my-list)))]))
```

1 op ← 1 op, 2 op, 1 op ← 1 op

Recursively sorted list

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

Let the time complexity of insersion sort be $T(n)$

n → size of our list.

Step 1: Write down the recurrence

$$T(n) = 5 + T(n-1) + S(n-1)$$

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

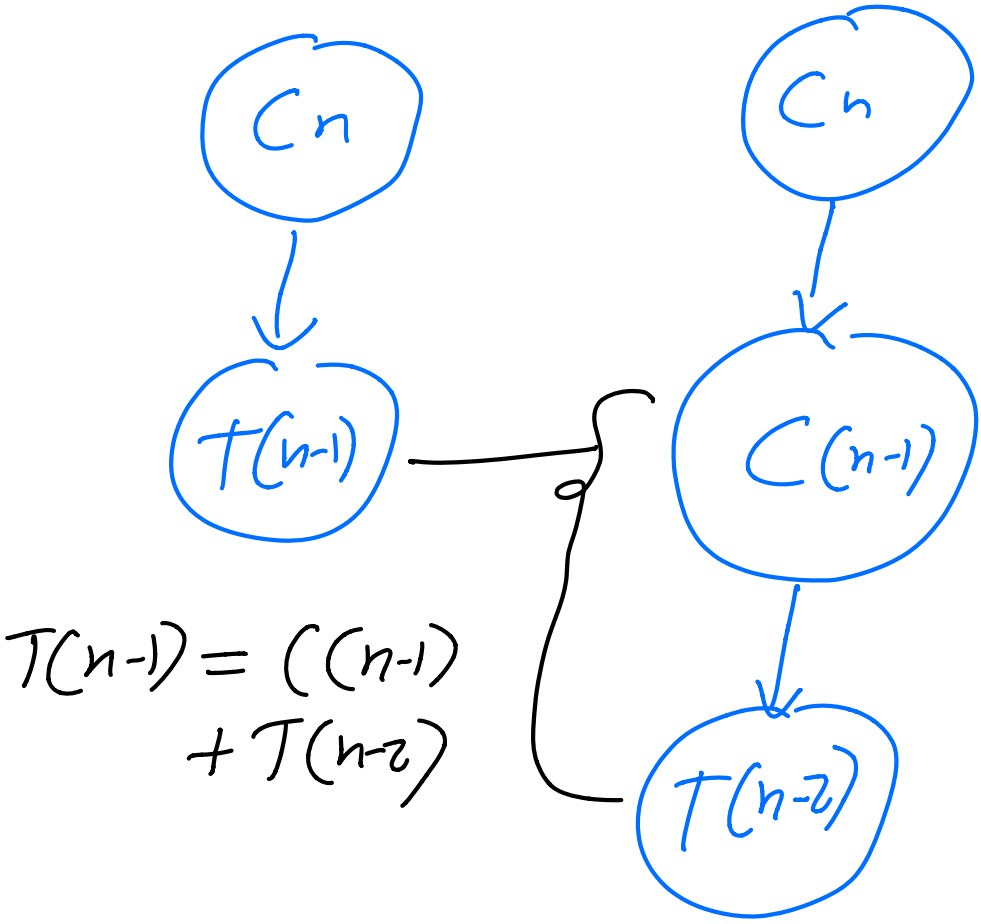
$$T(0) = C$$

$$S(n) = O(n)$$

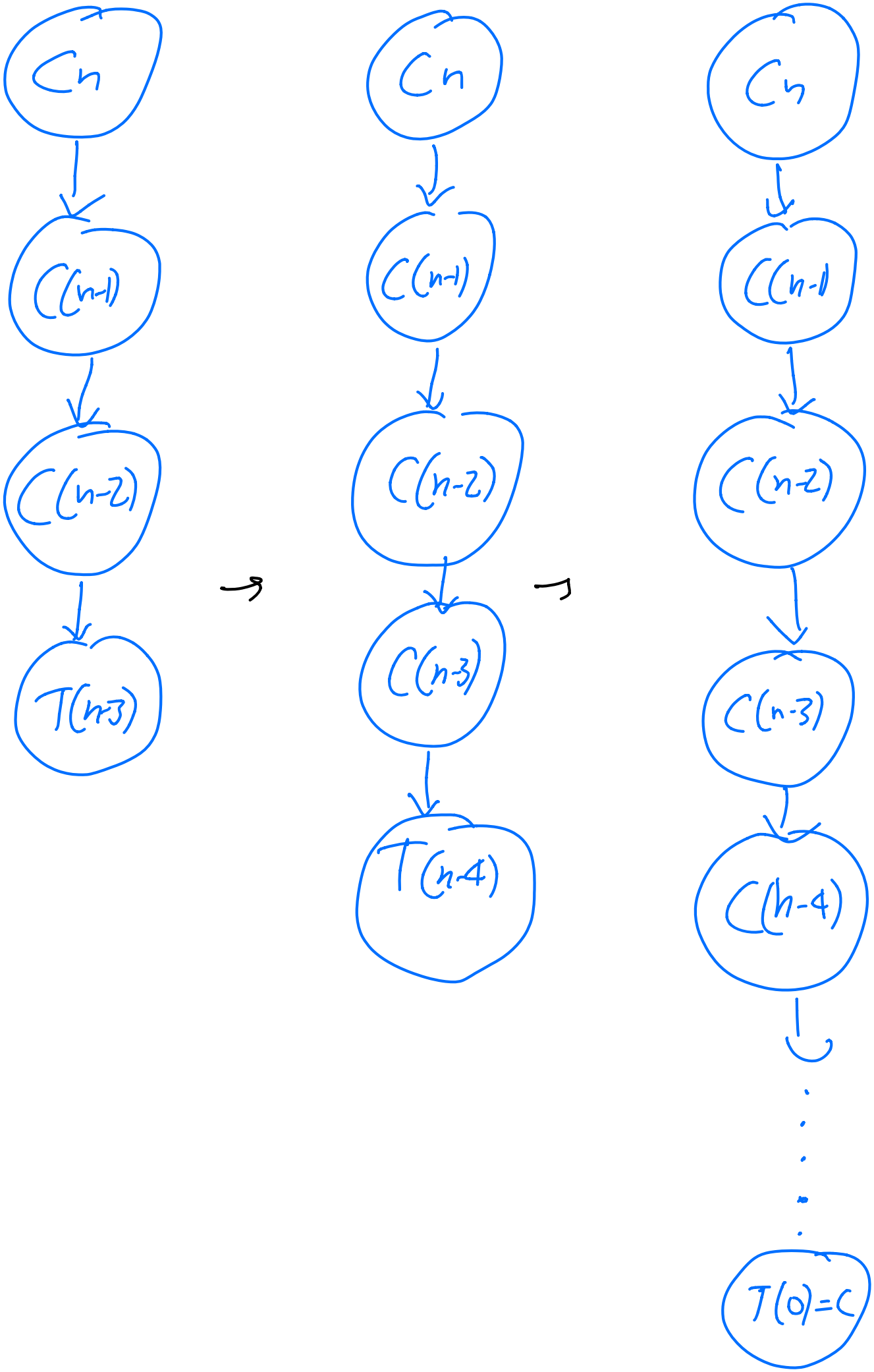
$$S(n-1) = C(n-1)$$

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

Step 2 : Solve this recurrence



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


Cn
C(n-1)
C(n-2)
T(n-3)
Cn
C(n-1)
C(n-2)
C(n-3)
T(n-4)
Cn
C(n-1)
C(n-2)
C(n-3)
C(n-4)
T(0)=C

$$T(n) = C_n + C_{(n-1)} + C_{(n-2)} + C_{(n-3)} + C_{(2)} + C.$$

$$= C(n + (n-1) + (n-2) + (n-3) + \ldots 1).$$

$$= C \times (\text{The sum of first } n \text{ natural numbers})$$

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

$$C\frac{n^2}{2} + C\frac{n}{2} \in O(n^2)$$

$$T(n) = O(n^2)$$

Time complexity of click-piece

```
(: click-piece : Piece Checkers -> Checkers)
(define (click-piece p game)
 (local [
  (: new-list : (Listof Piece))
  (define new-list (filter
    (lambda [p2 : Piece] (not (piece=? p2 p)))) (Checkers-pieces game)))
  ]
(Checkers new-list (Checkers-turn game)
  p)))
```

Define this function

Time complexity : $T(n)$

We are creating a new list.

What is the input size n.

n = size of the List ↓ Checker-pieces

Time-complexity of filter
= O(Time-complexity of our lambda * n)

piece=? Takes constant time.
(Fixed number of (comparisons) - operation.

Time of lambda = It Time-complexity of piece=? = Constant

Time - complexity of filter
= $O(\text{constant} * n) = O(n)$

Time complexity of creating new list is $O(n)$.

Time complexity of creating a creating a new checkers will be size of new-list + 1 + 1
= $n + 1 = O(n)$

Time complexity of click-piece
= $O(n) + O(n)$
= $cn + cn = 2cn$

$= O(n)$

So, time complexity of click-pile
$= O(n)$.