Fibonicci Series

1	1	2	3	5	8	13
0	1	2	3	4	5	6

$$fib(n) = \underbrace{fib(n-1)}_{\text{last}} + \underbrace{fib(n-2)}_{\text{second last}}$$

PMI - Extended Form

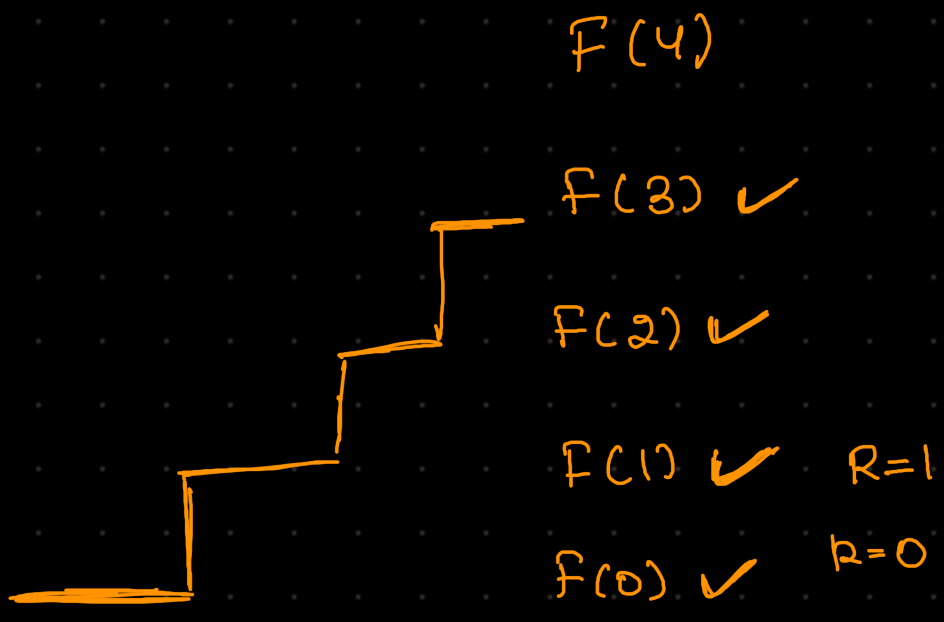
1. Proof for base case F(0) F(1)
2. Assume for F(i) equal to true

where your i can be from

$$0 \leq i \leq R$$ F(R)

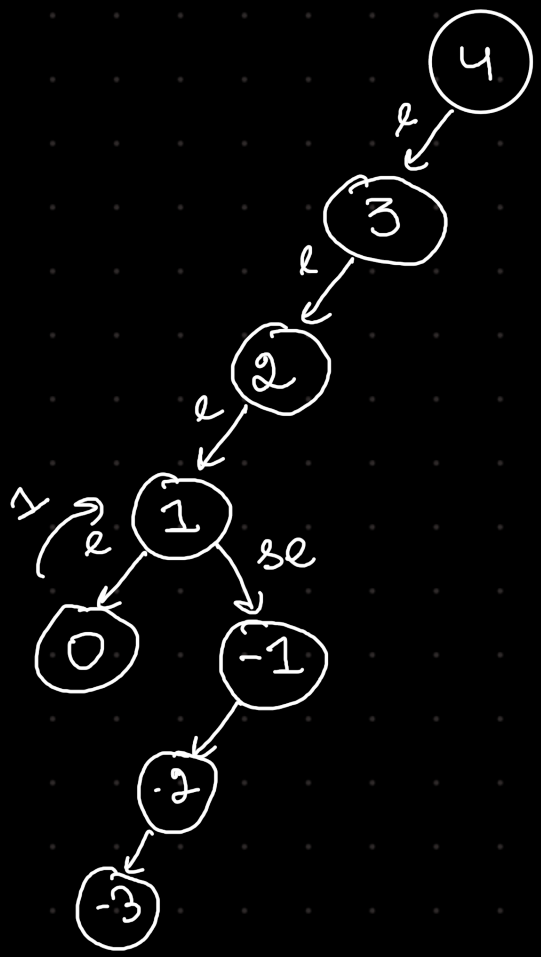
so that means we can assume for

F(R) true
F(R-1) true
} ⇒ all assume true for less than n

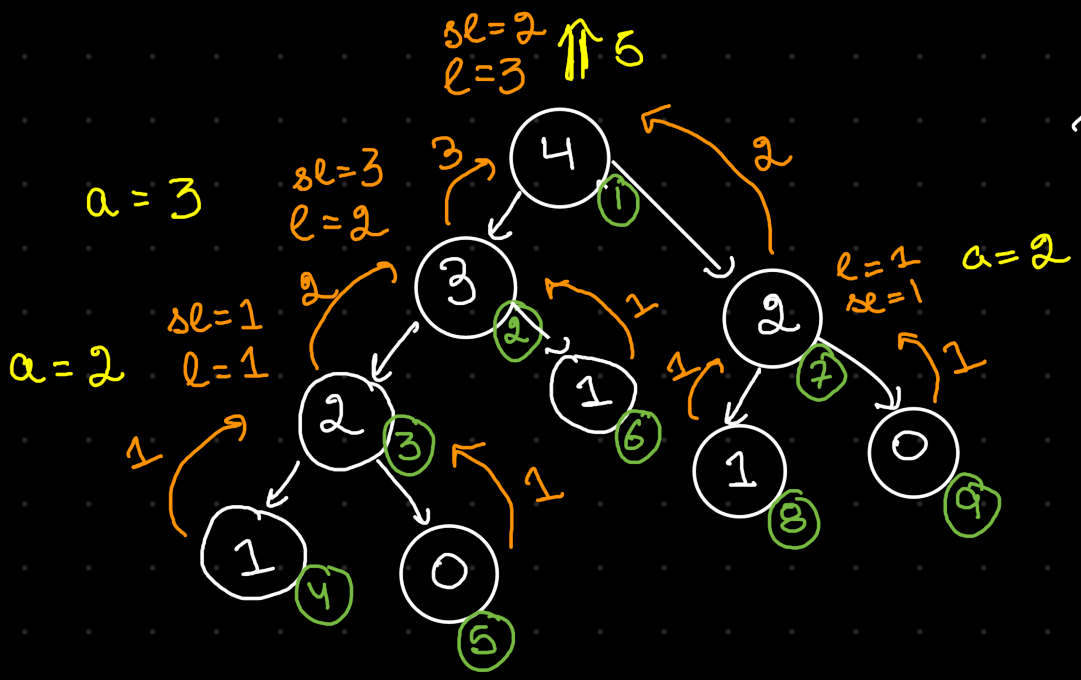


Last F(n-1) to be true

second last F(n-2) to be true


4
3
2
1
0
-1
-2
-3

Fibonicci Series : Recursion Tree



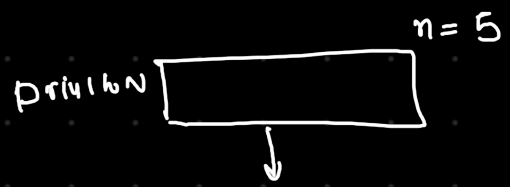

```python
def fibonicci(n):
    # print(n)
    if(n==0):
        return 1
    if(n==1):
        return 1

    last = fibonicci(n-1)
    secondLast = fibonicci(n-2)

    ans = last + secondLast

    return ans
```

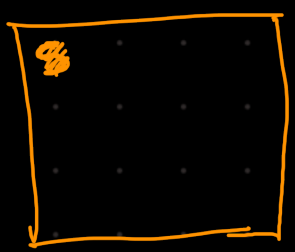
```
The function call is 1 for 4
The function call is 2 for 3
The function call is 3 for 2
The function call is 4 for 1
The function call is 5 for 0
The function call is 6 for 1
The function call is 7 for 2
The function call is 8 for 1
The function call is 9 for 0
```

Q: Write a program for a given number n.

1. Print 1 to N.
2. Print N to 1

Print 1 to N :-





Print N to 1

1. Base Case
2. Recursive call
3. Our work

Head

Tail

Assignment

1. Sum of digit of a number

2. Power of a number (base, exp)
   3 2

Head vs Tail Recursion

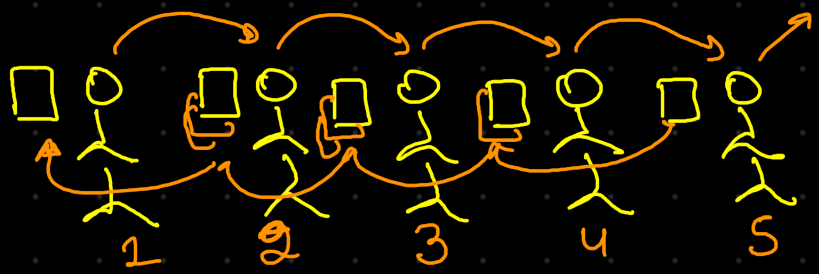
Head

We make recursive call at the beginning of our fn implementation.

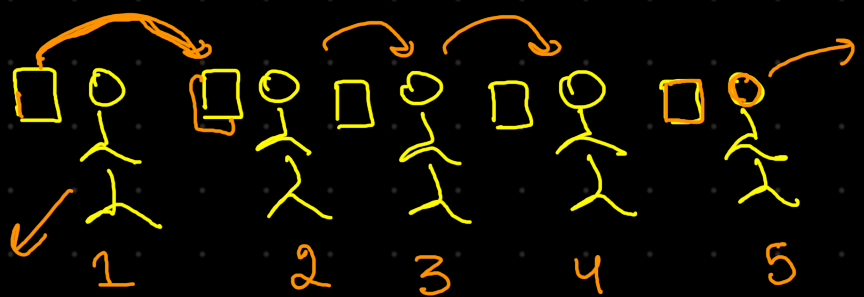
Tail

When we make recursive call at the end of our implementation

head



tail



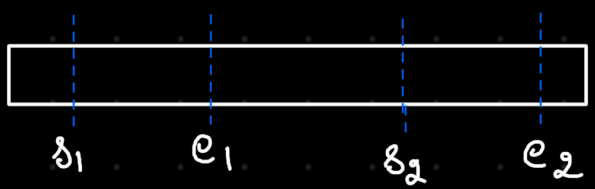
Recursion with Arrays/List

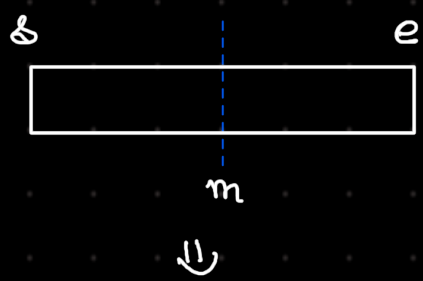
Whenever in arrays, we are dealing with a problem that can be broken & individually be applied to left over part, we can use recursion there.

Below are major ways we break our array/list.

1. First or last element removed

2. Using start and end index (eg. binary search)





divide from midde

3. Copy part into a new array and send to further calls.

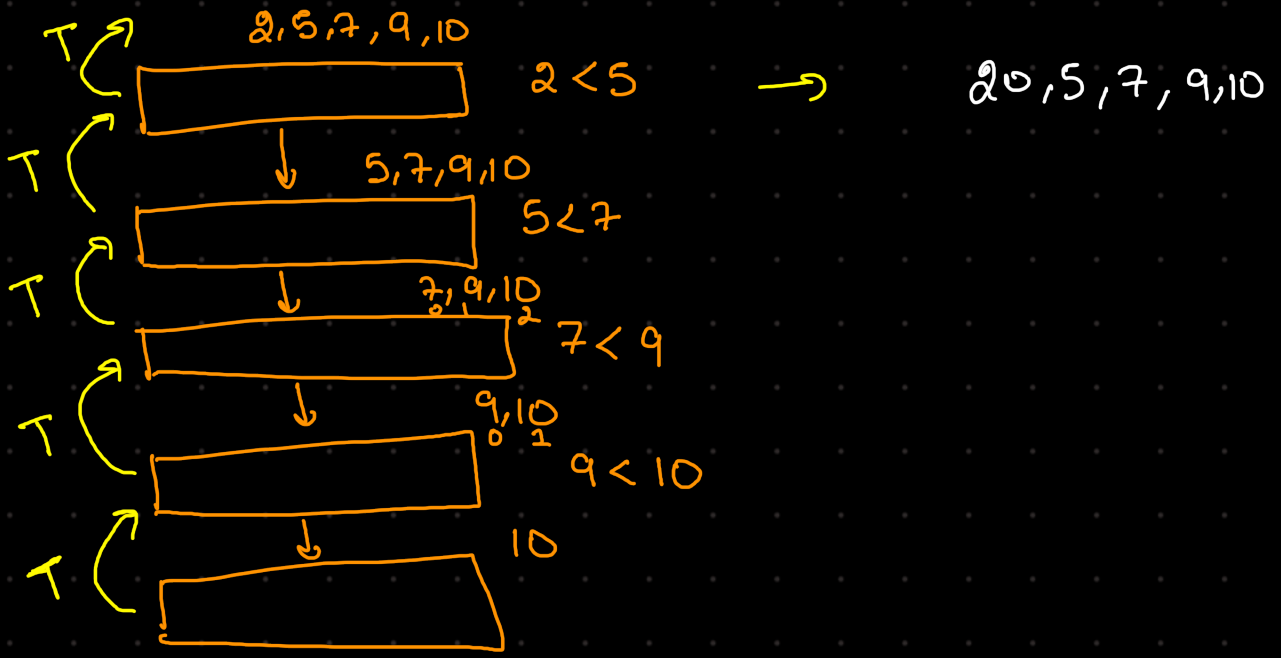
Q: Write a Program to check if array is sorted ?

recurrence relation recursion tree

head / tail recursion.

1. Base case : size == 1 or size == 0

2. Recursion call



check sorted (a1) = (a1[0] < a1[1]) and check sorted (a1[1:])

Q: Sum of an array using Recursion.

Please specify recursion relation & draw recursion tree as well.

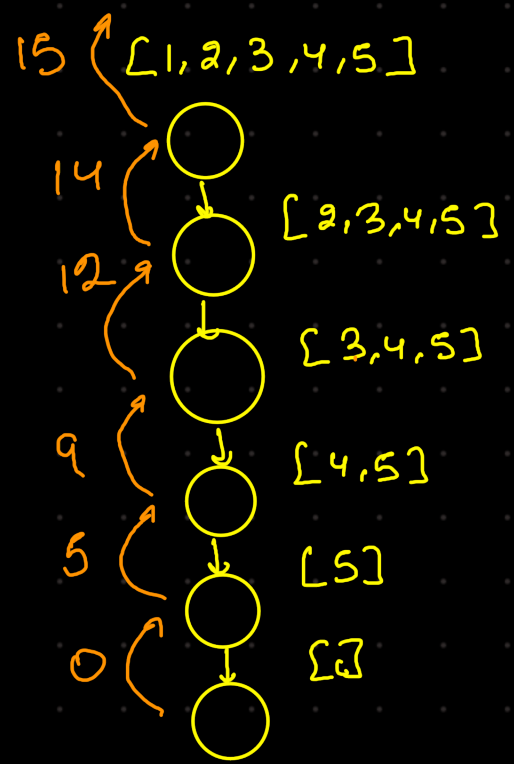
Give head and tail recursion solution.

1. Base Case size = 1 A[0]

   size = 0 0

2. sum (A | A[1:]))

3. A[0] + sum of 11



So, above question showed us that either we can do our work first or after recursive call.

1] Base Case
2] Recursive call / small answer
3] Our calculation

Q: Given an array and an element, find

a) first index of element.

b) last index of element.

[3, 2, 5, 2, 8, 2, 1], 2 -1 if not found
 0 1 2 3 4 5 6

first : 1
last : 5

first index :- 1] Base case

2] if al[0] == x
return 0

3] recursive call

Q: Write a program to find all index of a given element.

Result have to be as below :-

1] Print the indexes

2] Update the answer in list provided

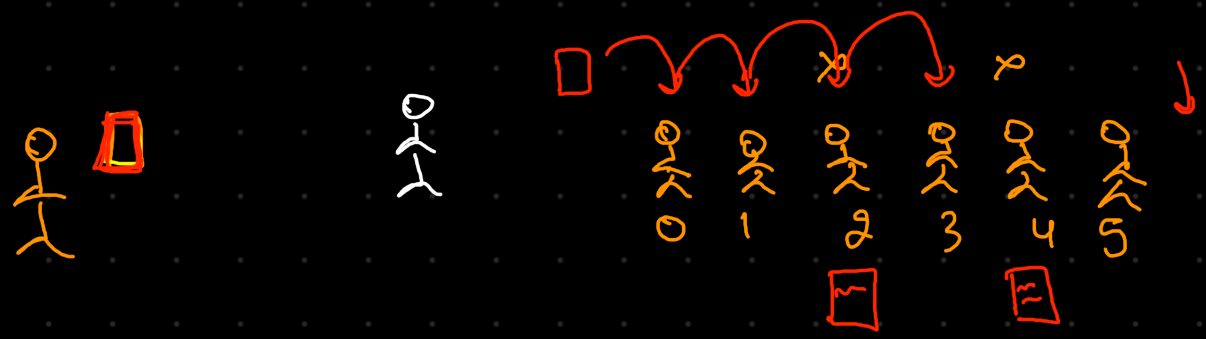
3] Update indices in the global list

4] Return answer as a new list.

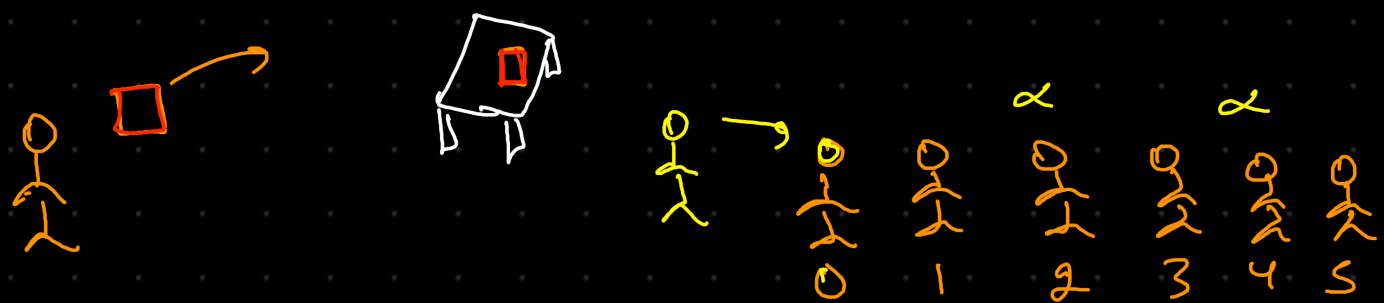
1] [3, 2, 5, 2, 8, 2, 1], 2

1
3
5

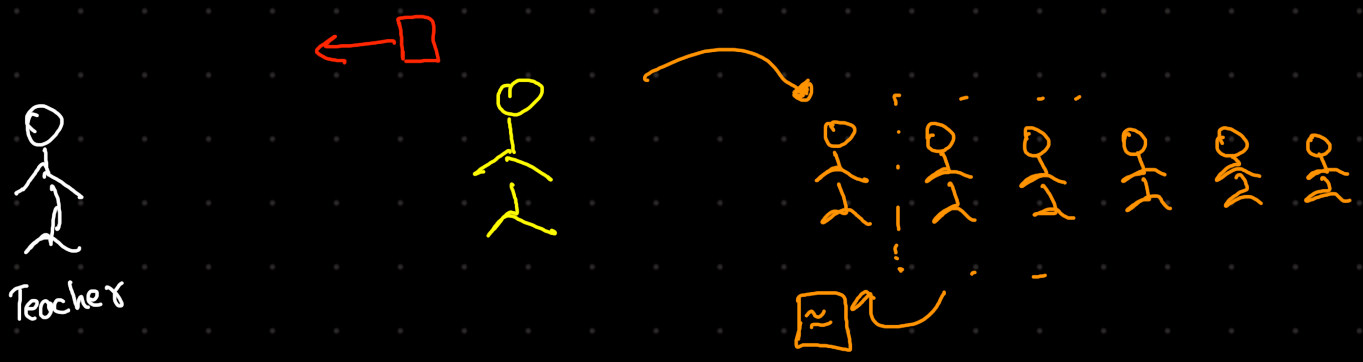
Update all indices in a provided list



Update indices in a global list



Return a list as an answer



1. Base case