Recursion

1. Identify if you can breakdown problems into simpler problems
2. Write the recurrence relation if needed
3. Draw the recursion tree
4. About the tree:
   - See flow of funcⁿs
   - Identify and follow left tree calls & right tree calls
   - Draw tree till recursion not clear ! imp
   - Use a debugger
5. See how values are returned at each step, what type of values, where the function call will come out, In the end, you will come out of main function.

For reference

```
s=0
e=len(nums)-1
while s<e:
  mid=s+(e-s)//2
  :
```

Binary Search using recursion:

```
① s→0, e→ len(nums)-1
② def binarySearch(s,e):
       mid= s+ (e-s)//2
        if s>=end:
          return -1   [Base case]*
       if target == nums[mid]
         ~~binarySearch(0, mid-1)~~
          return mid
       elif target < nums[mid]
     return binarySearch(s, mid-1)
       else:
     return binarySearch(mid+1, e)

~~return -1~~
```

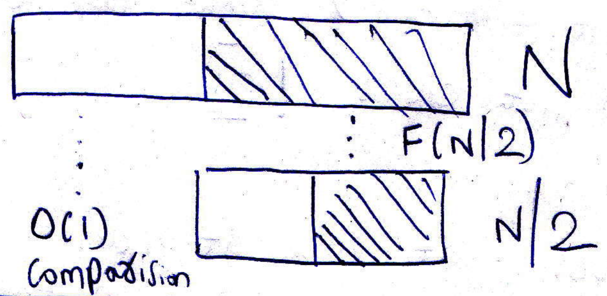
⊛ Variables in recursion

i] Arguments (Variables that need to be passed in future calls go in arguments)

ii] Return type

iii] Body of funcⁿ

Recurrence relation:



$$\boxed{F(N) = \underbrace{O(1)}_{\text{Comparision}} + \underbrace{F(N/2)}_{\text{Dividing}}}$$

⊛ Types of recurrence relations :3

① linear recurrence relation $\longrightarrow$ eg: Fibo Fibo (* Dynamic Programming Says Hello ;))

② Divide & Conquer rr $\rightarrow$ Eg: B.S

ro Says:
Do not overthink!

Bro Says:
Make sure to return a function call if the primary function has return type