Sat/Sunday → 9-11/1 → 9-12

Wednesday Night → Doubt solving

New topic → Mobile → Additional sessions

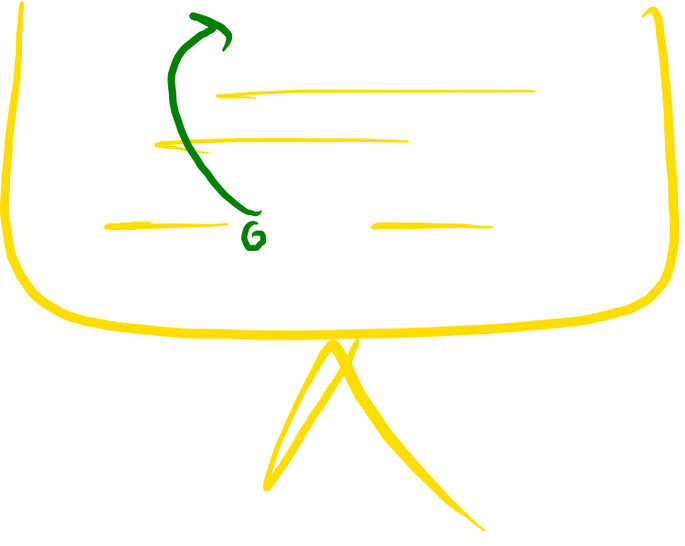
Class → Questions → Discuss

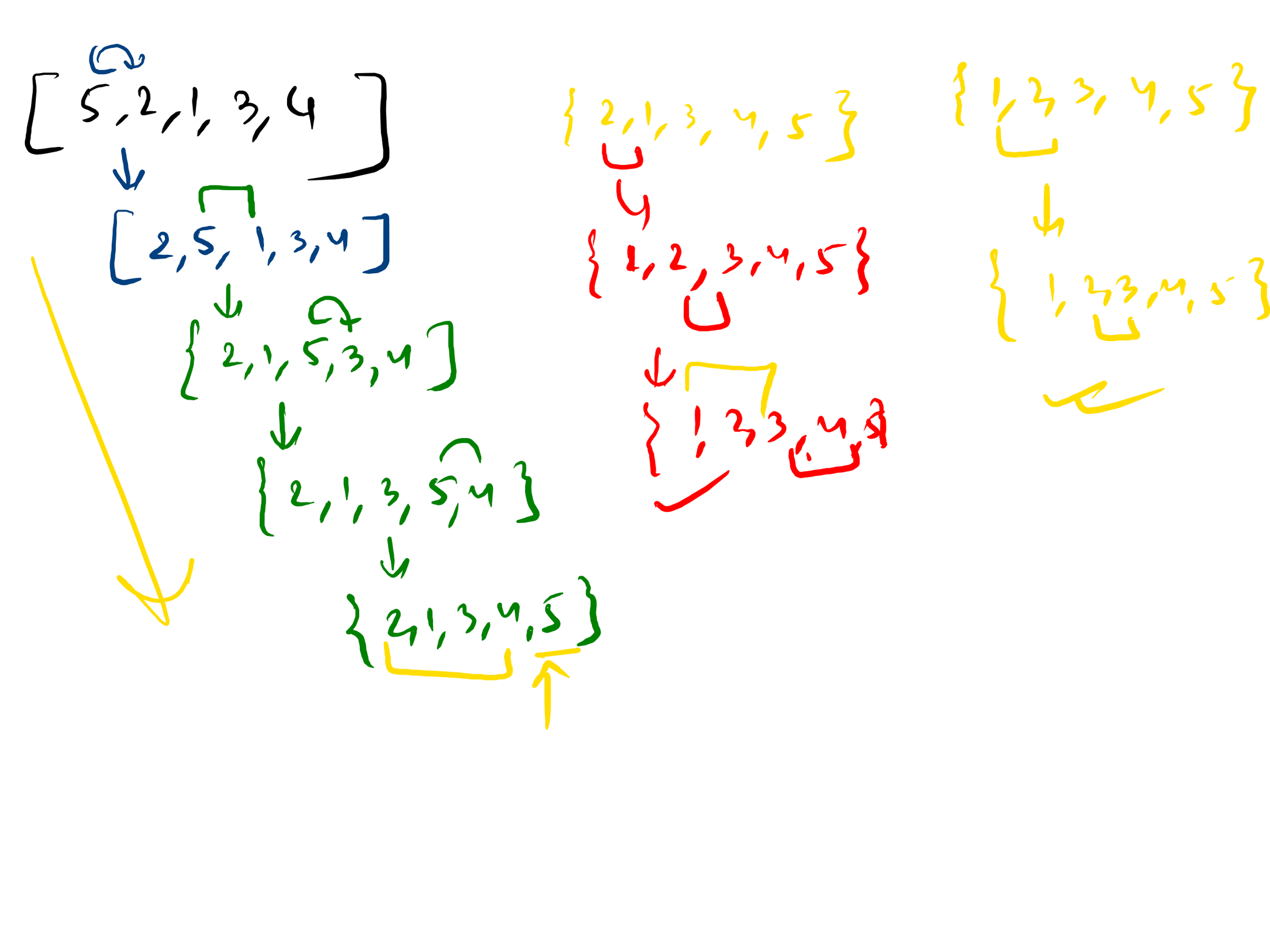
Sorting → Arrange elements in proper logical order

Data → ① storage
Data → ② computation → search

3 Sorting Algorithms :-

1. Bubble Sort




[5, 2, 1, 3, 4]
[2, 5, 1, 3, 4]
{ 2, 1, 5, 3, 4]
{ 2, 1, 3, 5, 4 }
{ 2, 1, 3, 4, 5 }
{ 2, 1, 3, 4, 5 }
{ 1, 2, 3, 4, 5 }
{ 1, 3, 3, 4 }
{ 1, 3, 3, 4, 5 }
{ 1, 3, 4, 5 }

Worst sorting algorithm

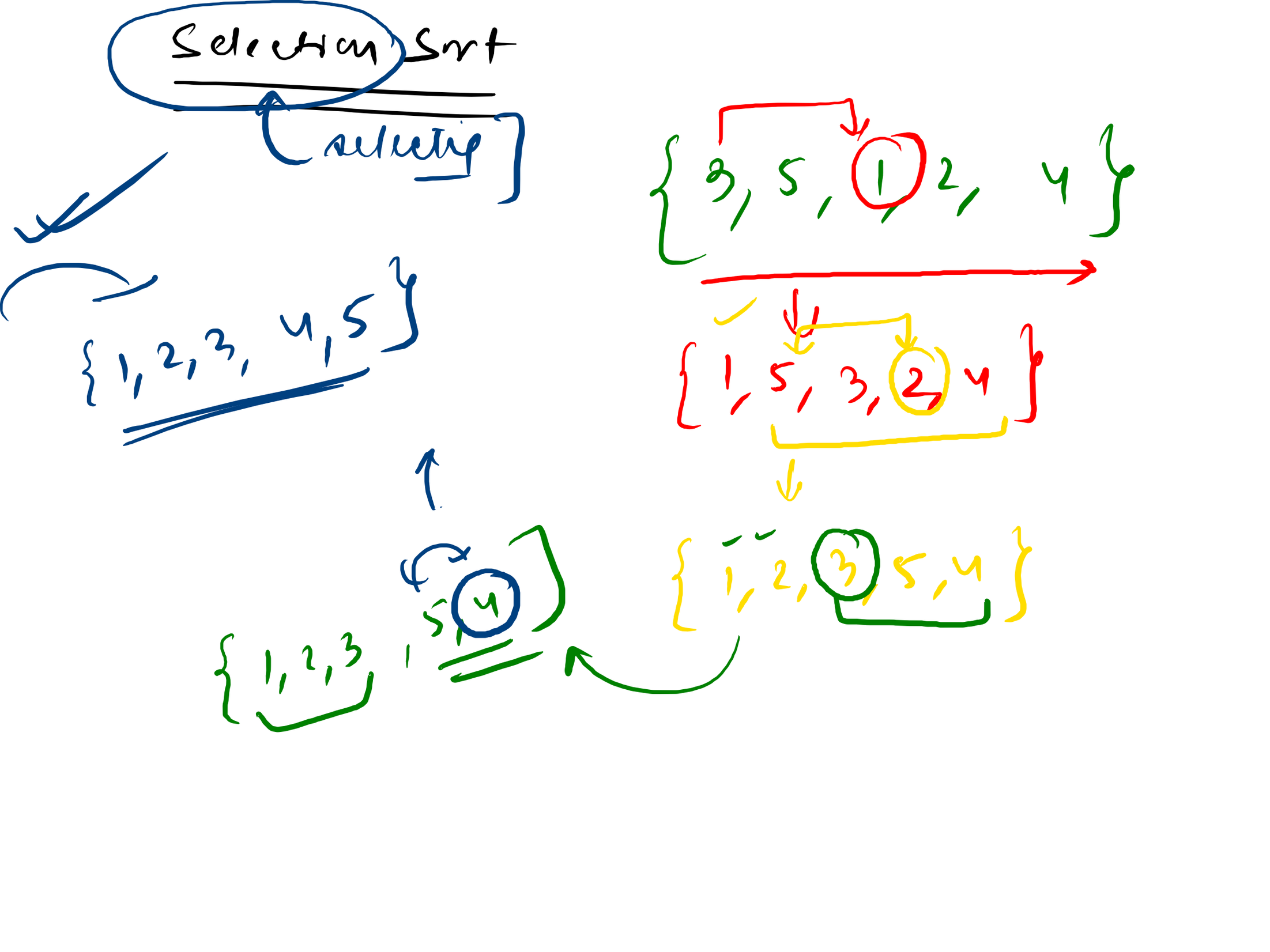
```
def bubbleSort(arr):
  for i in range(len(arr)-1, 0 , -1):
    for j in range(i):
      if(arr[j]>arr[j+1]):
        arr[j],arr[j+1] = arr[j+1],arr[j]
```

stable

$TC = O(n^2)$

comparisons $\approx n^2$

swaps $\approx n^2$


Selection Sort
selecting
{3, 5, 1, 2, 4}
{1, 2, 3, 4, 5}
{1, 5, 3, 2, 4}
{1, 2, 3, 5, 4}
{1, 2, 3, 5, 4}

```python
#Selection Sorting
def selectionSort(arr):
  for i in range(len(arr)):
    min =i
    for j in range(i+1,len(arr)):
      if arr[j] <arr[min]:
        min =j
    #Find the smallest values
    arr[i],arr[min] = arr[min],arr[i]
```

→ select the smallest

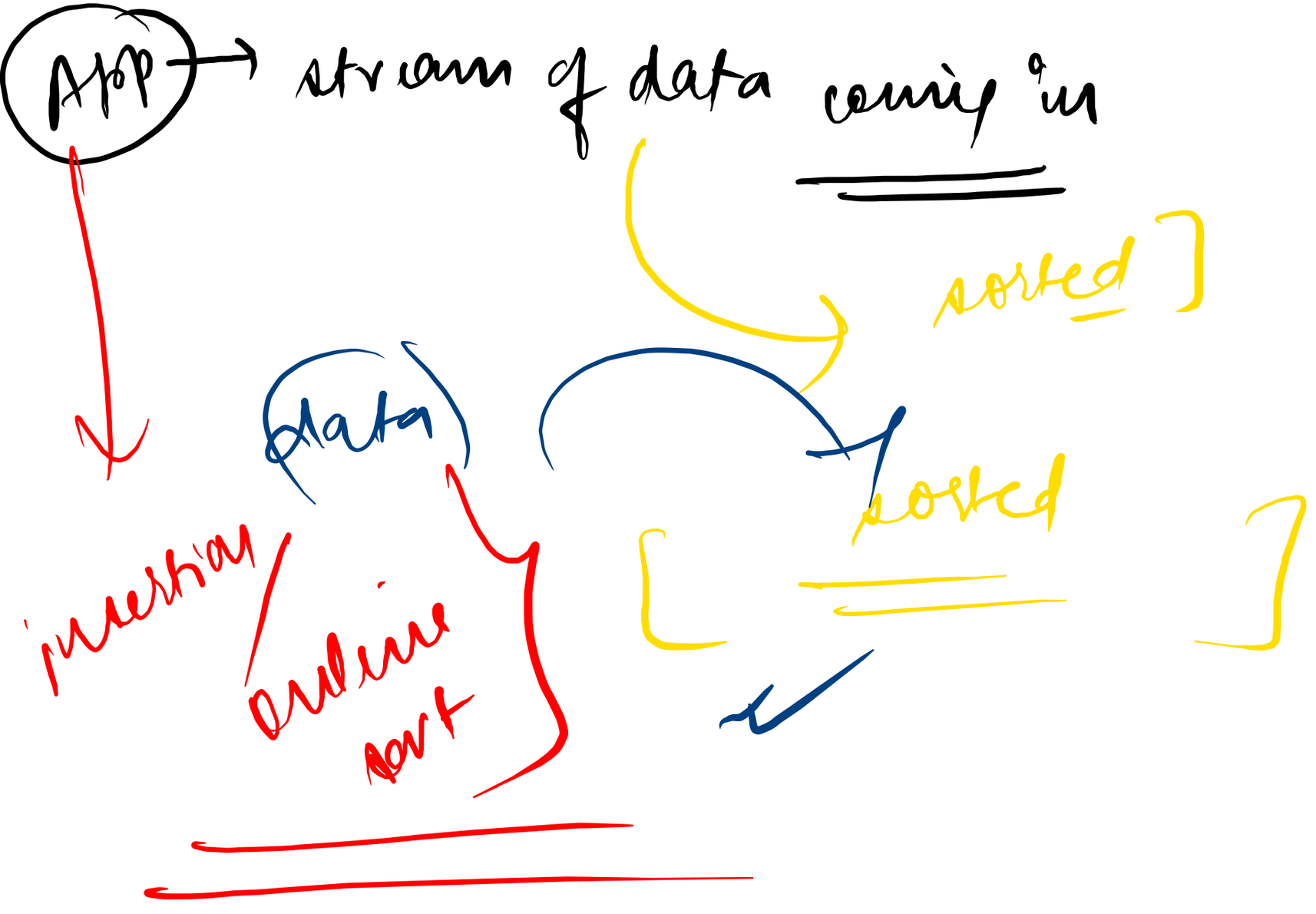
→ put the smallest at i'th index

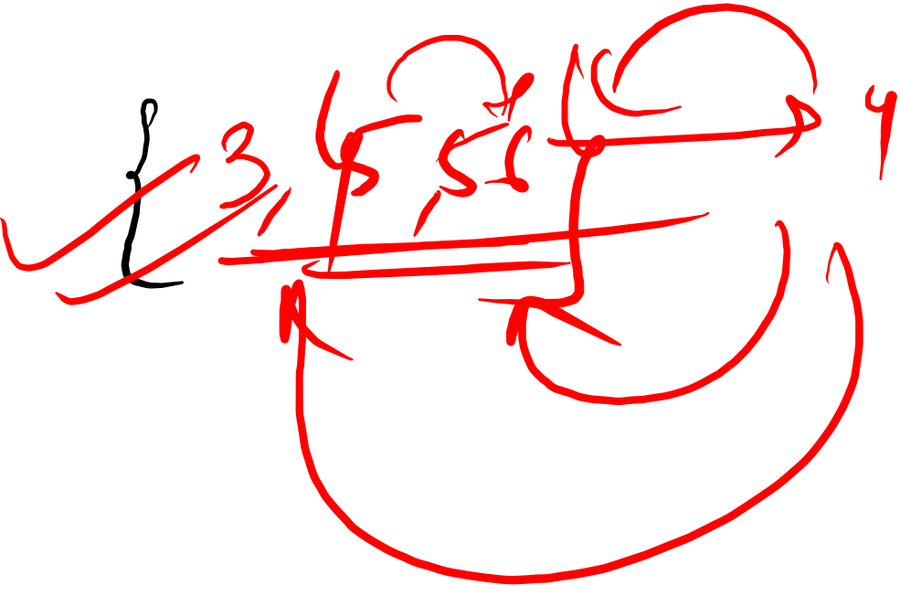
TC = O(n²)

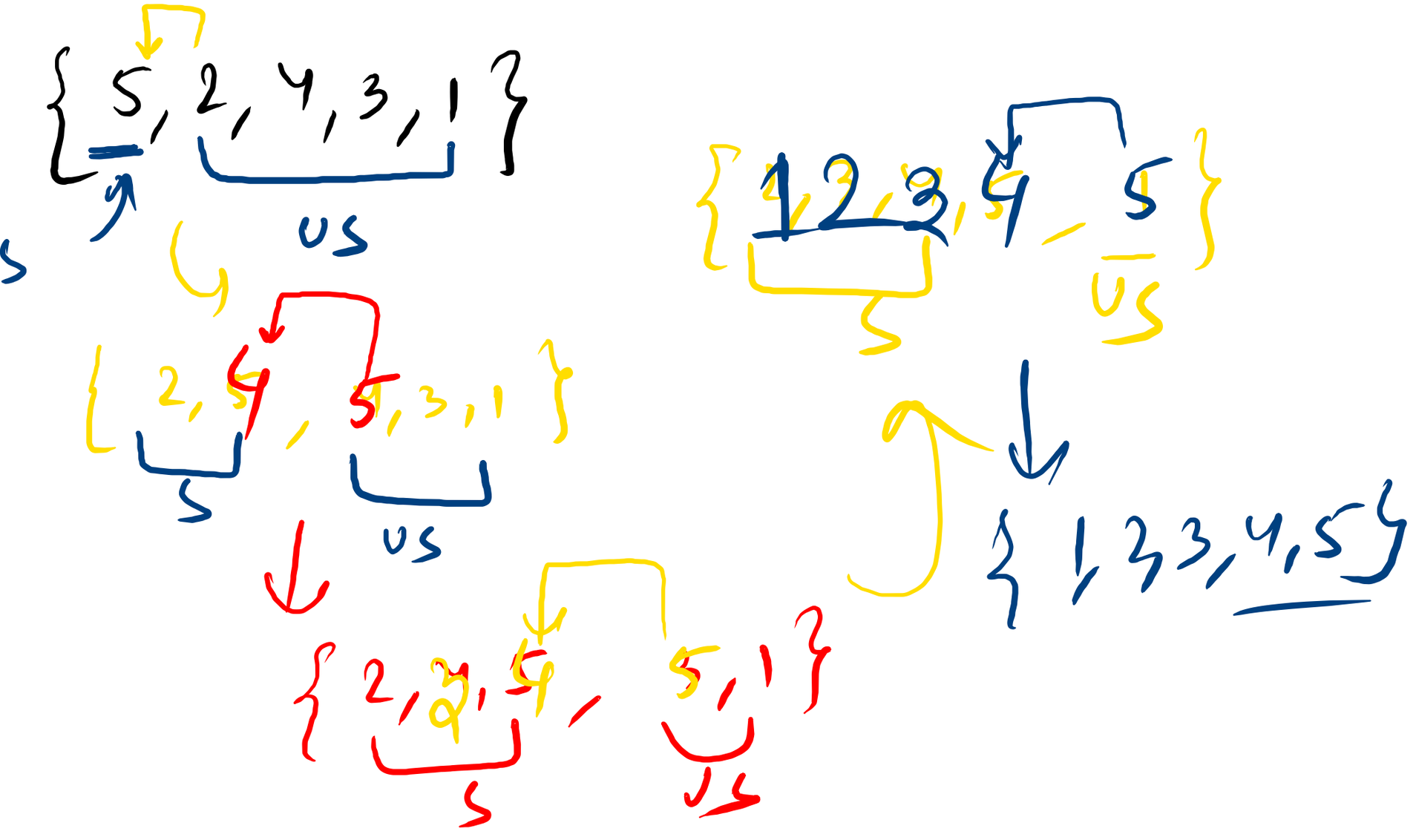
comparisons ≈ n²

swaps ≈ n

unstable


App
stream of data coming in
sorted
data
inserting
ordering
sort
sorted


3
5
5
4


{ 5, 2, 4, 3, 1 }
us
{ 2, 4, 5, 3, 1 }
s
us
{ 2, 3, 4, 5, 1 }
s
us
{ 1, 2, 3, 4, 5 }
s
us
{ 1, 3, 4, 5 }

```python
def insertion_sort(arr):
    for i in range(1,len(arr)):
        v= arr[i]
        j=i
        while(j>=1 and arr[j-1]>v):
            arr[j]=arr[j-1]
            j-=1
        arr[j]=v
```

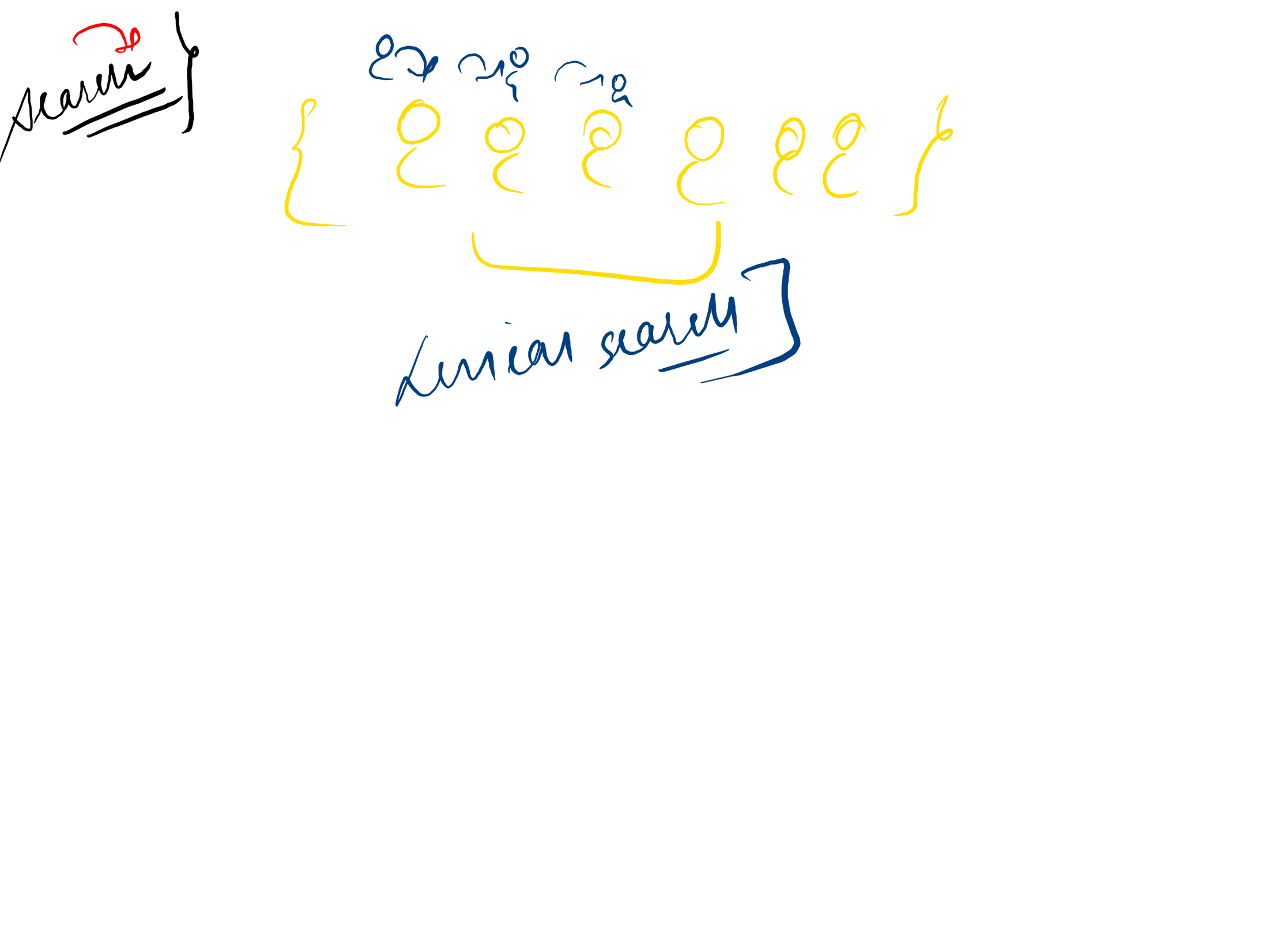
$TC = O(n^2)$

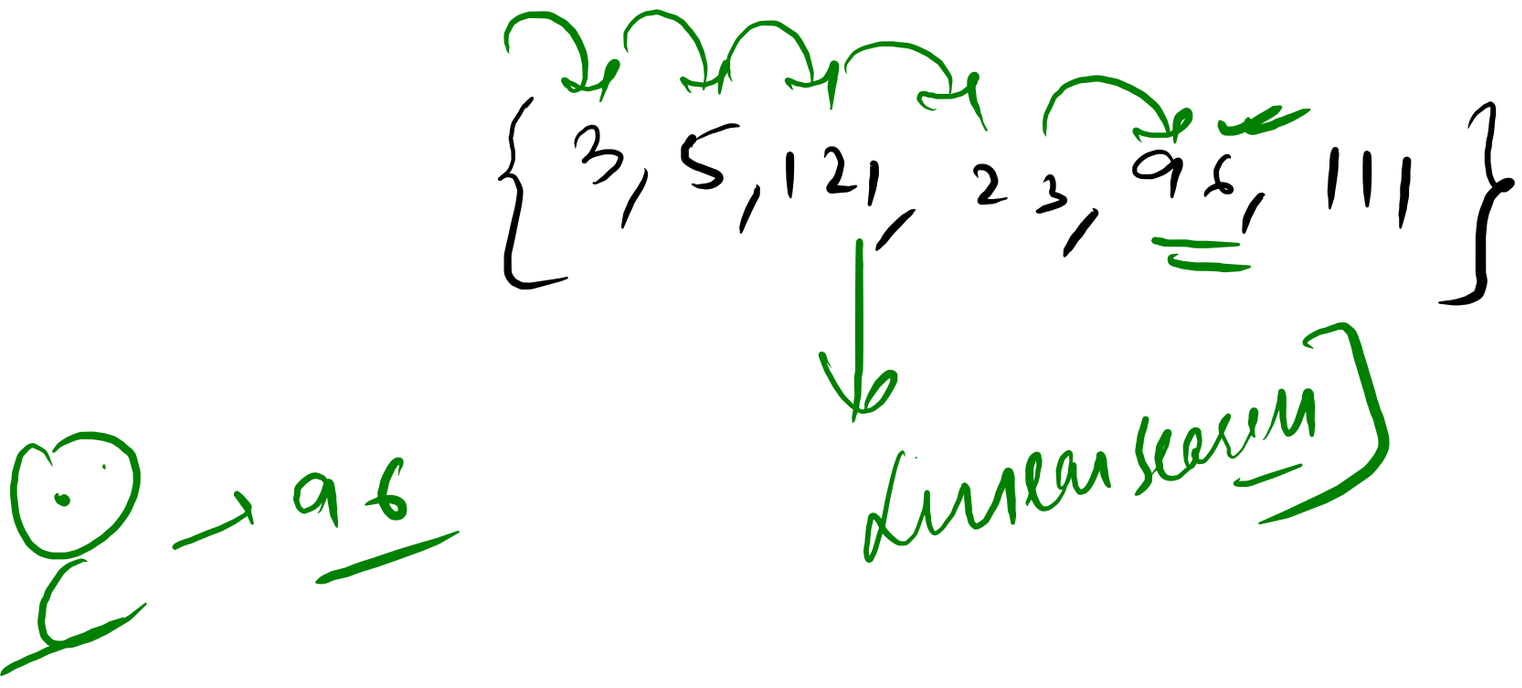
comparisons $\simeq n^2$

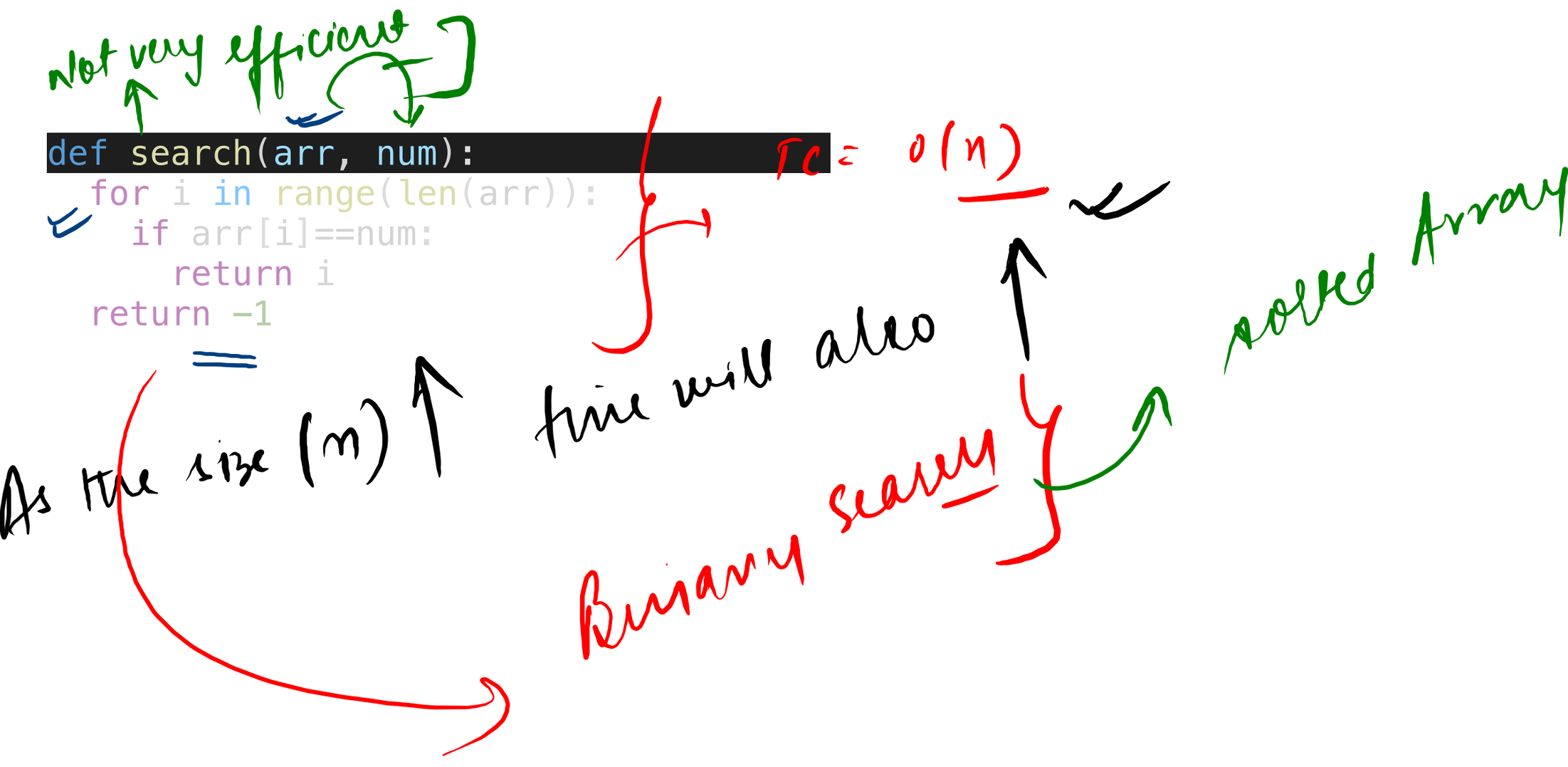
swaps $\simeq n^2$

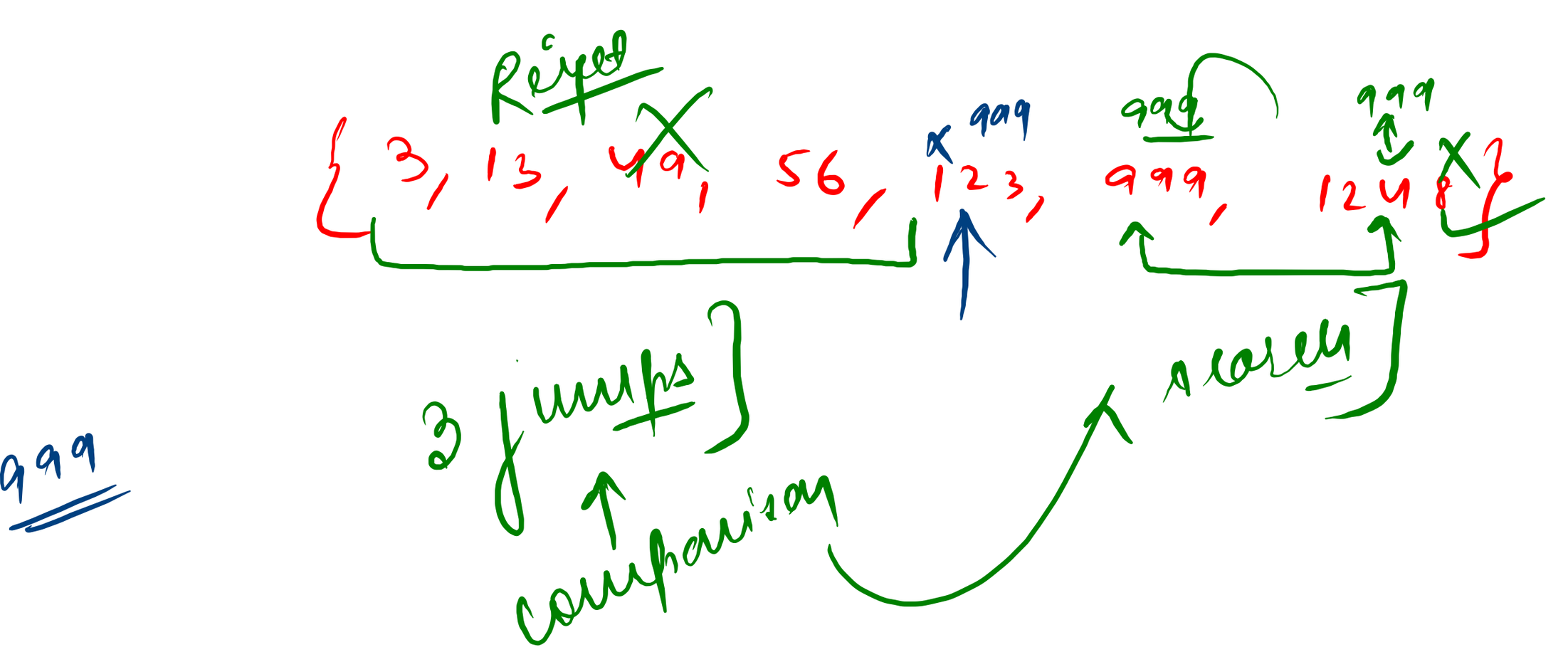
outer loop

↳ insert data in all already sorted collection


Lexical scope


{3, 5, 121, 23, 96, 111}
Linear search


not very efficient
def search(arr, num):
for i in range(len(arr)):
if arr[i]==num:
return i
return -1
TC = O(n)
As the size (n) ↑ time will also ↑
Binary search
sorted Array


Reject
3, 13, 49, 56, 123, 999, 1248
999
999
999
3 jumps
comparison
return

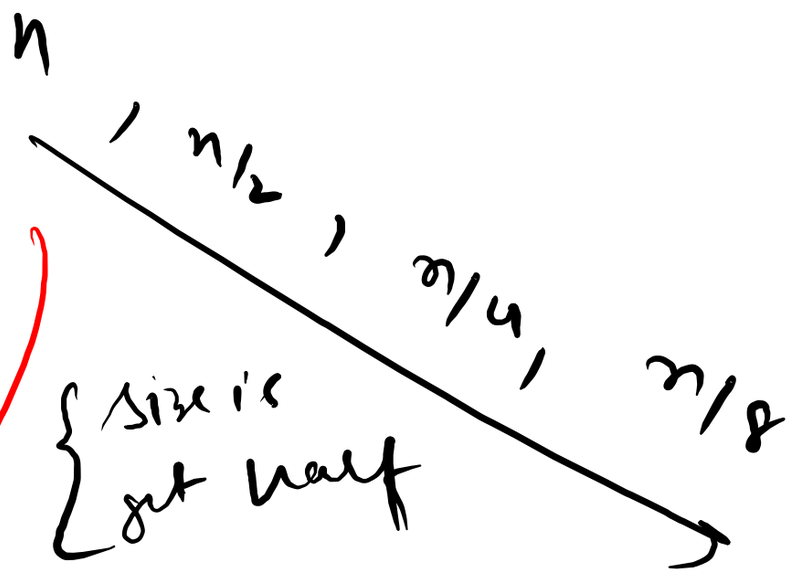
999

```
def binarySearch(arr, num):
  left = 0
  right = len(arr) -1


  while(left <=right):
    mid = int(left + (right-left)/2)
    if arr[mid] == num:
      return mid
    elif arr[mid] > num :
      right = mid-1
    else :
      left = mid+1
  return -1
```

(left+right)/2

= Middle

TC =



Tc : O(log n)

Binary search | Linear search

Graph

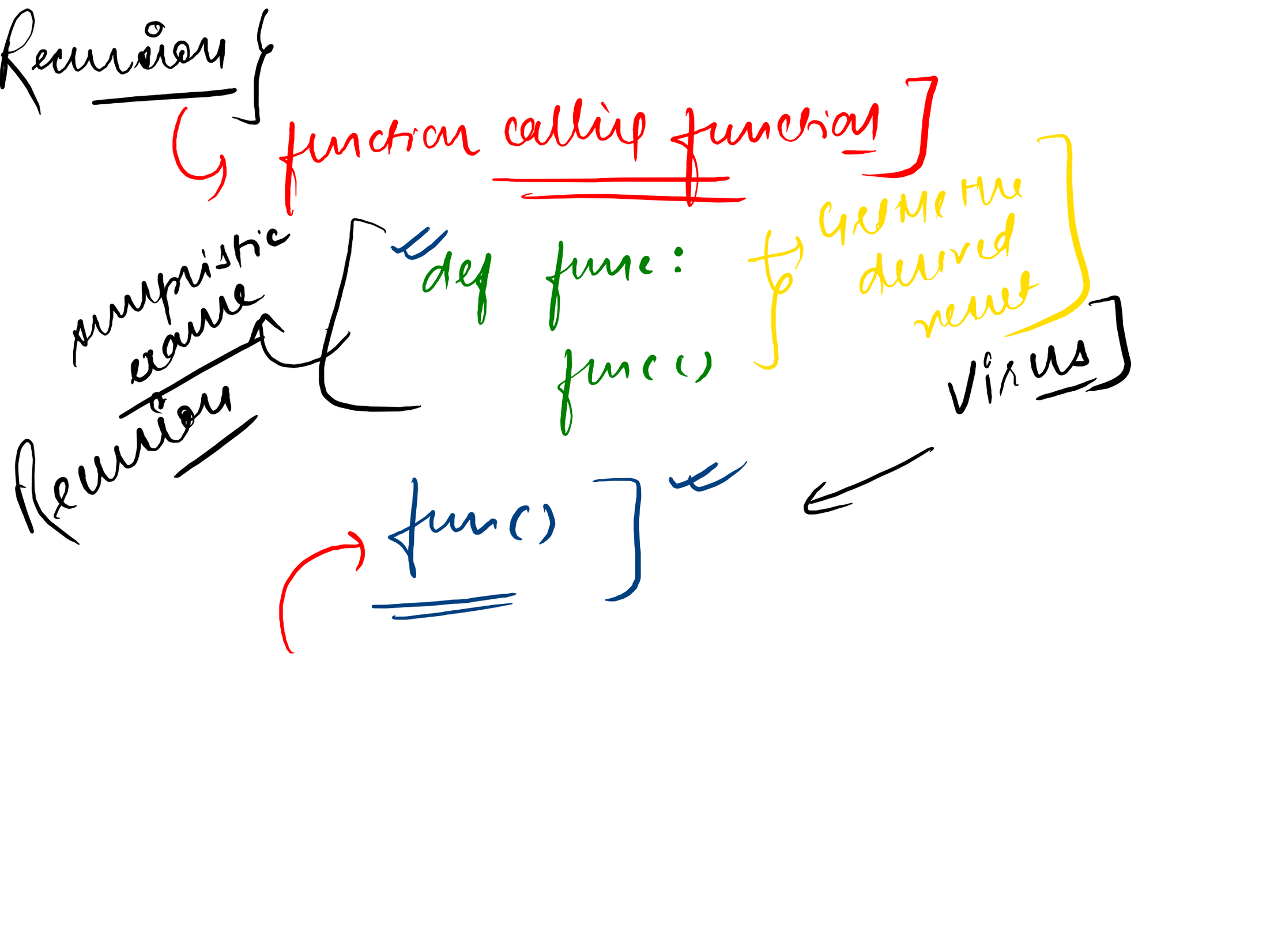
Tree

DP

complex problems

90%

100%


Recursion
function calling function
simplistic example
Recursion
def func:
func()
Genuine derived recursion
virus
func()

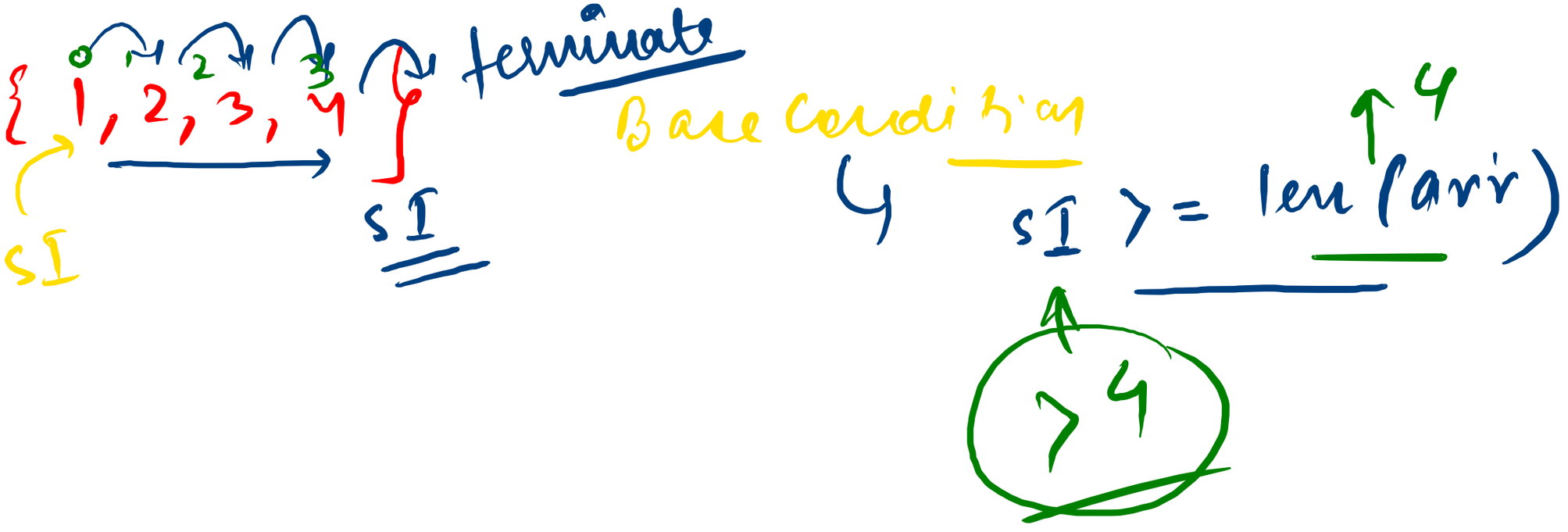
Prerequisite

Recursion

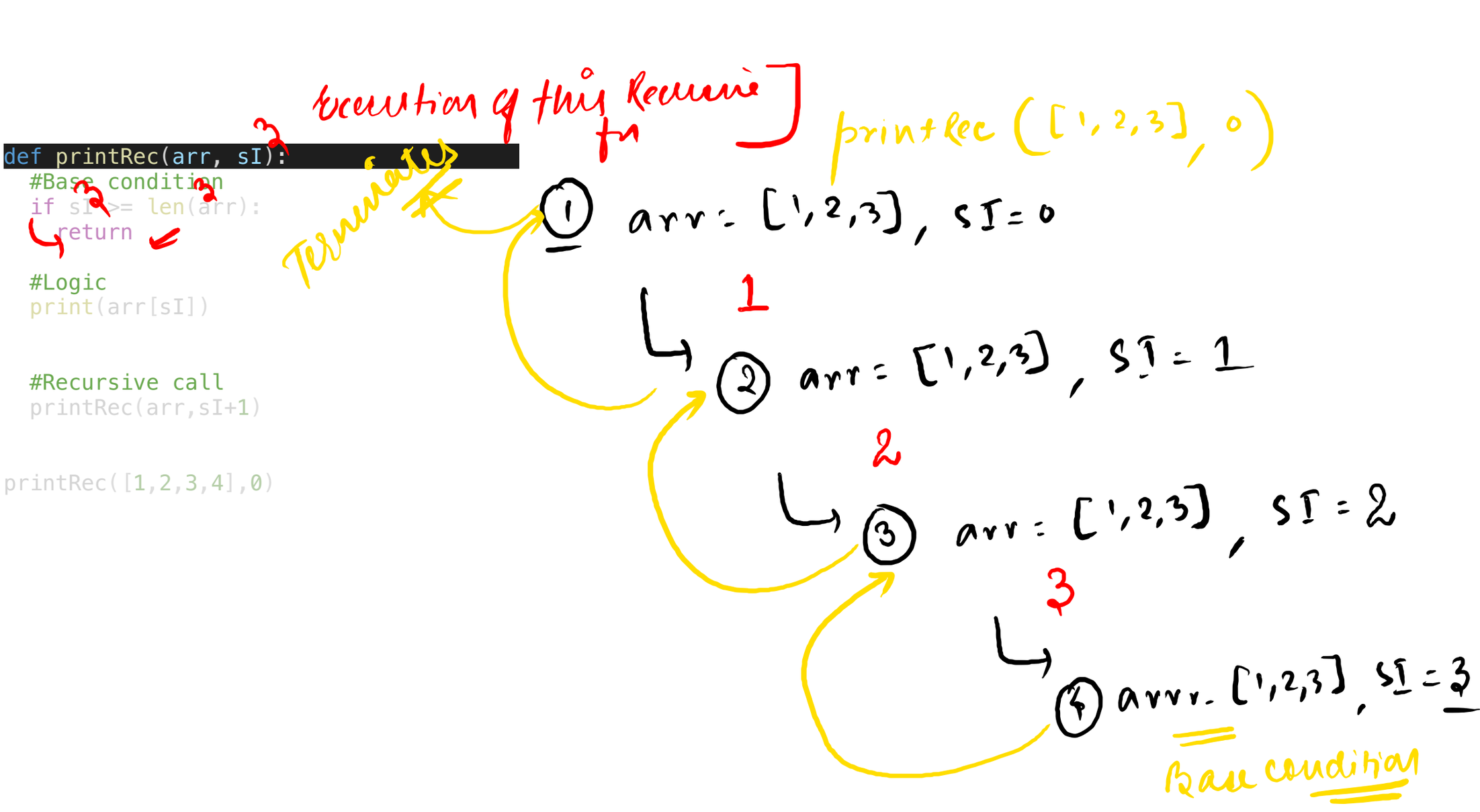
1. Base condition / Termination condition
2. Logic
3. Call the function again (recursive call)

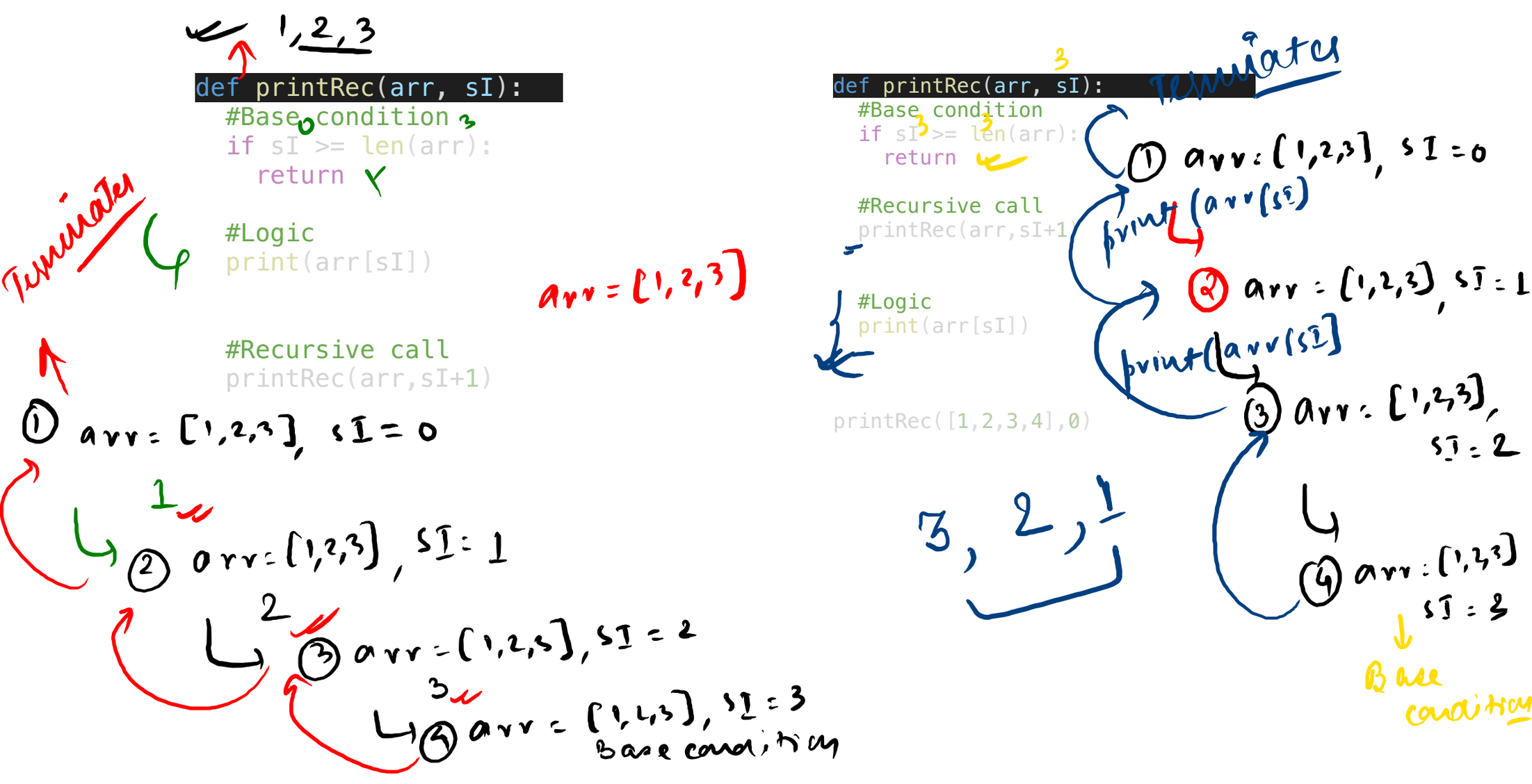
{1, 2, 3, 4} → 1, 2, 3, 4 Recursion

→ def printRec(arr, sI)

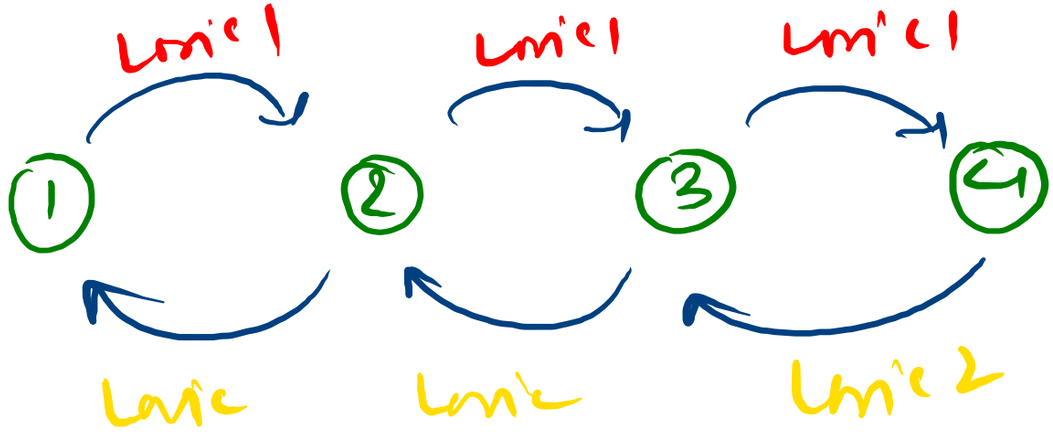

terminate
Base Condition
{ 1, 2, 3, 4
sI
sI
sI >= len(arr)
4
4
>4

```python
def printRec(arr, sI):
  #Base condition
  if sI >= len(arr):
    return

  #Logic
  print(arr[sI])


  #Recursive call
  printRec(arr,sI+1)


printRec([1,2,3,4],0)
```




1,2,3
def printRec(arr, sI):
#Base condition
if sI >= len(arr):
return
#Logic
print(arr[sI])
#Recursive call
printRec(arr,sI+1)
Terminates
arr = [1,2,3]
① arr = [1,2,3], sI = 0
1
② arr = [1,2,3], sI = 1
2
③ arr = [1,2,3], sI = 2
3
④ arr = [1,2,3], sI = 3
Base condition
def printRec(arr, sI):
#Base condition
if sI >= len(arr):
return
#Recursive call
printRec(arr,sI+1)
#Logic
print(arr[sI])
printRec([1,2,3,4],0)
Terminates
① arr = [1,2,3], sI = 0
print(arr[sI])
② arr = [1,2,3], sI = 1
print(arr[sI])
③ arr = [1,2,3], sI = 2
④ arr = [1,2,3], sI = 3
Base condition
3, 2, 1

Recursion



(RF)

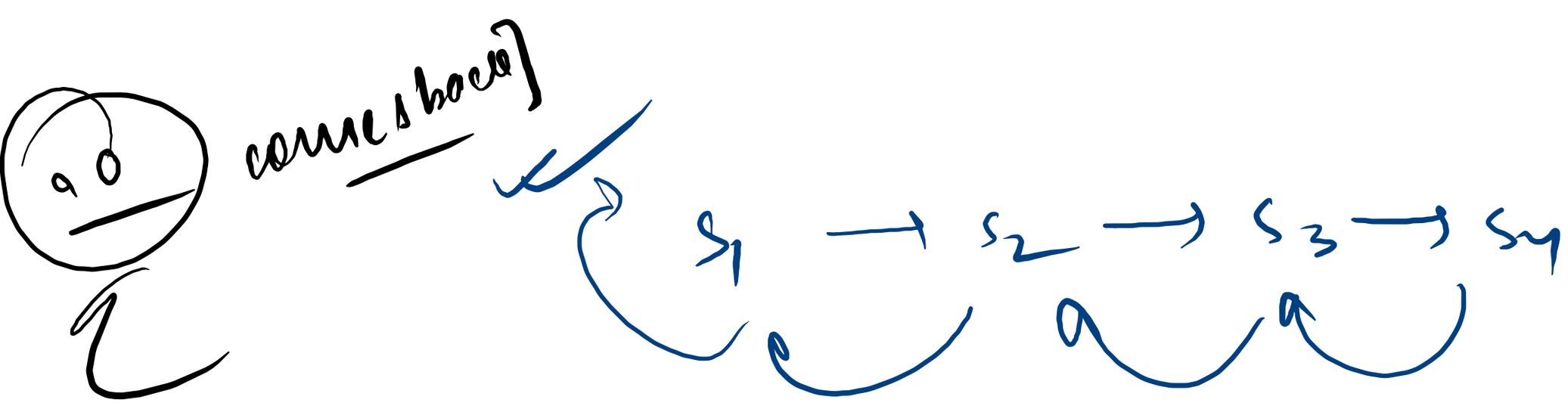
→ Base Condition

→ Logic 1

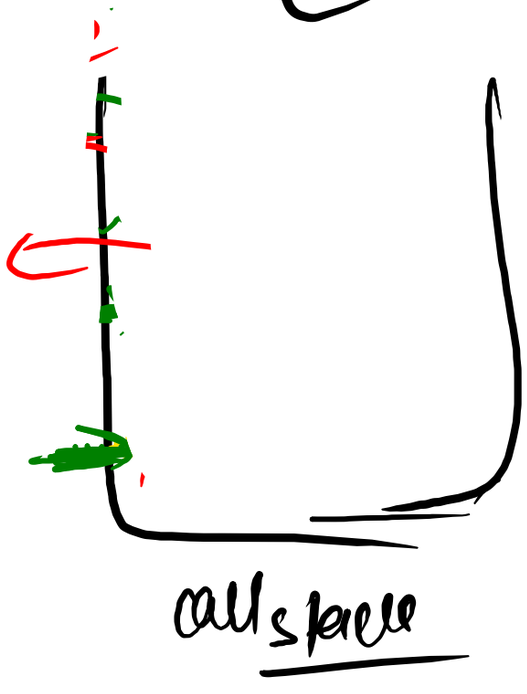
→ Recursive call

Recursive call

Logic 2


come back
s_1
s_2
s_3
s_4

```python
def printRec(arr, sI):
    #Base condition
    if sI >= len(arr):
        return

    #Logic
    print(arr[sI])


    #Recursive call
    printRec(arr,sI+1)
```



→ printRec([1,2,3], 0)

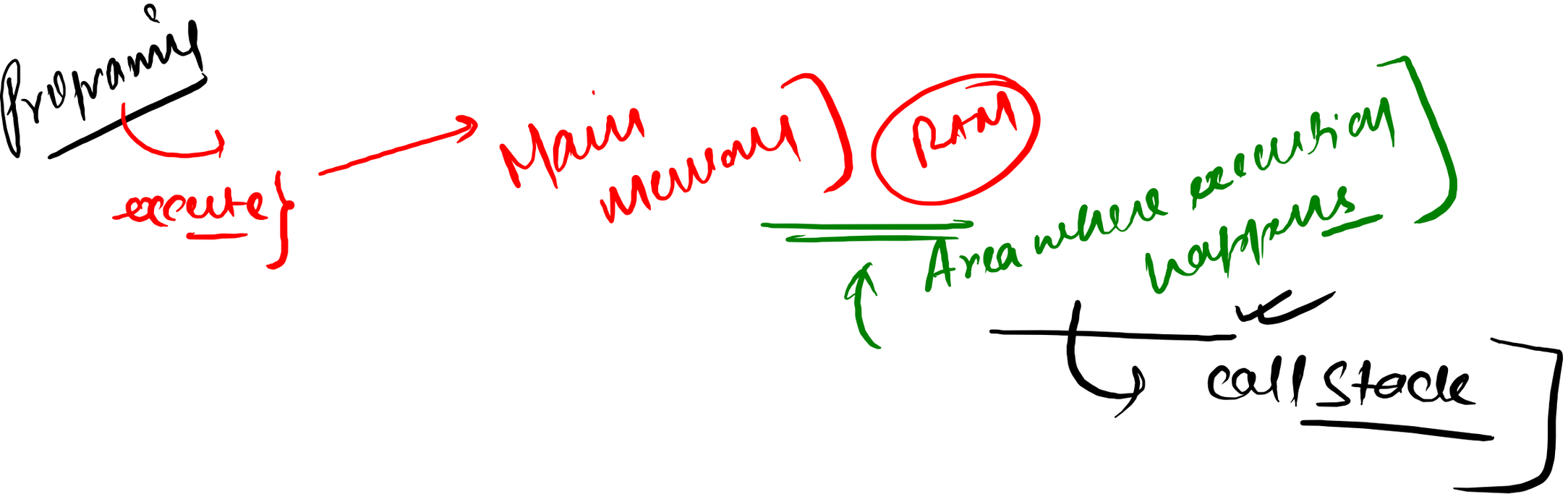
① arr = [1,2,3], sI = 0

② arr [1,2,3], sI = 1

③ arr [1,2,3], sI = 2

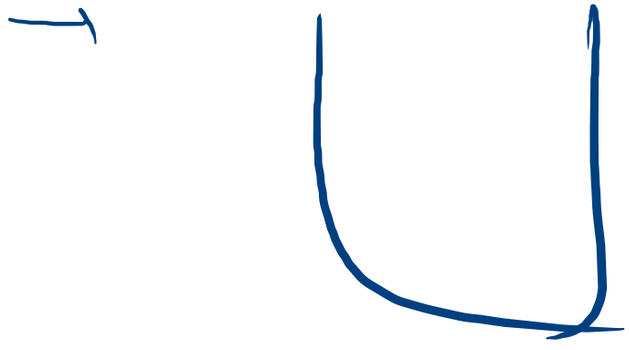
④ arr [1,2,3], sI = 3

1, 2, 3


Program
execute
Main memory
RAM
Area where execution happens
call Stack

Code

f1
f2
f3
f4



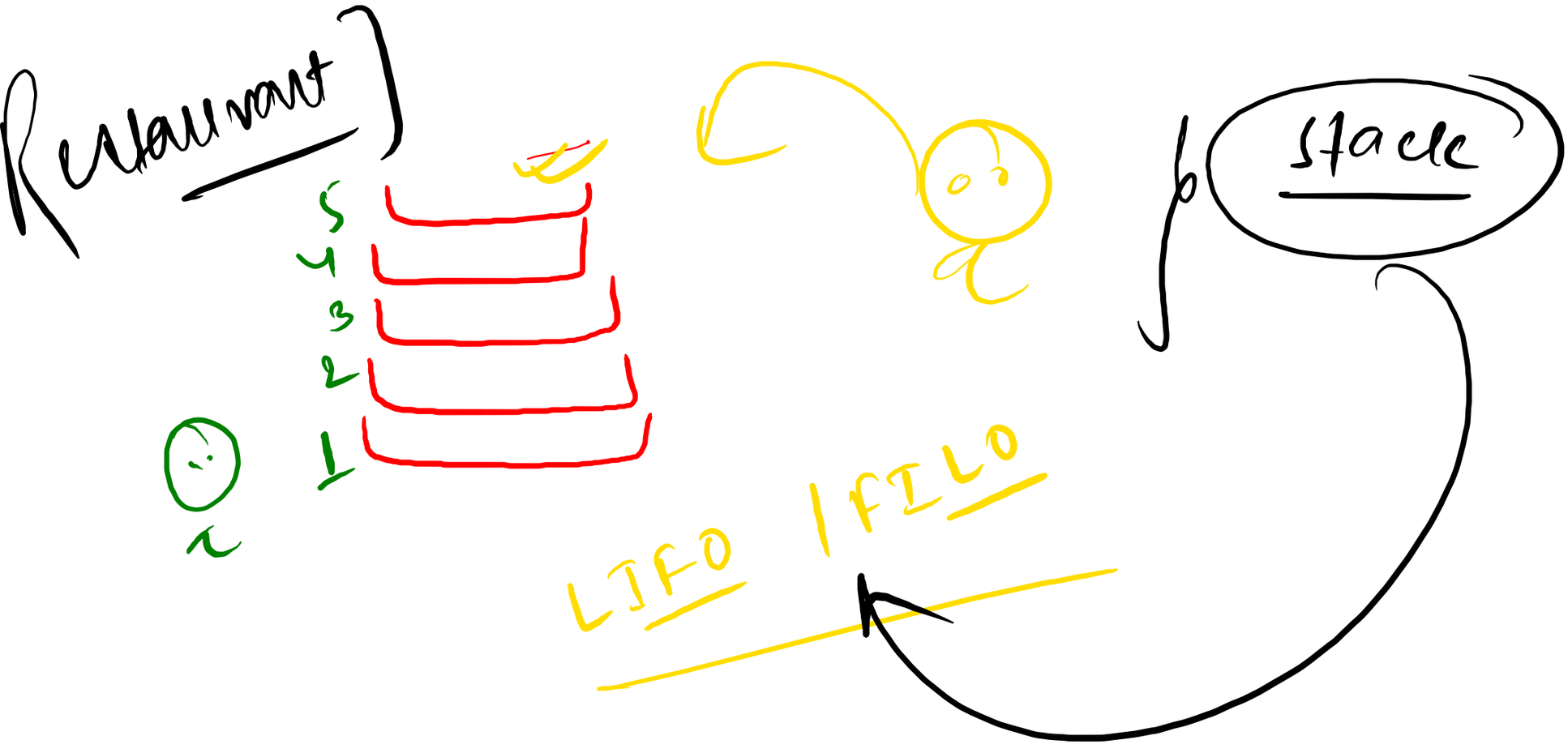
call stack

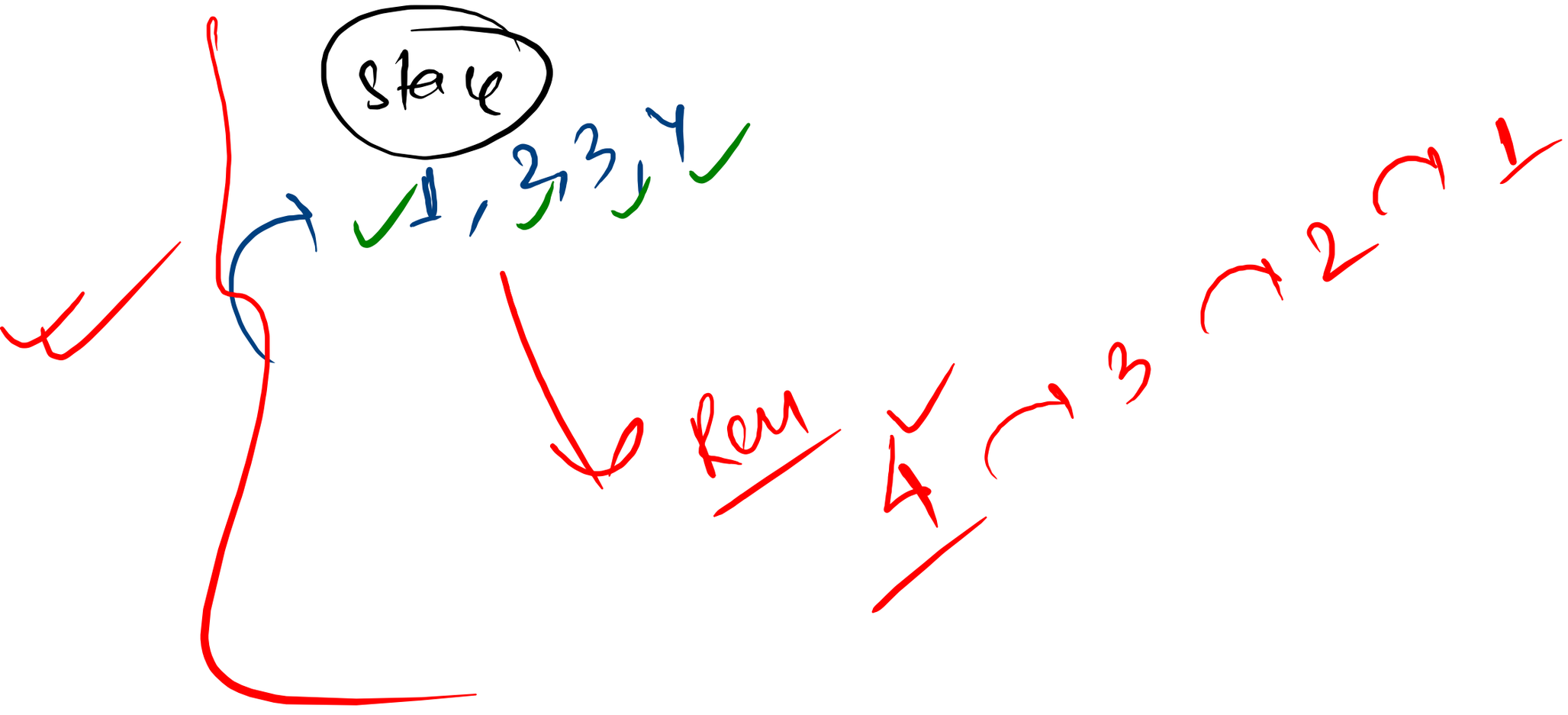
↓

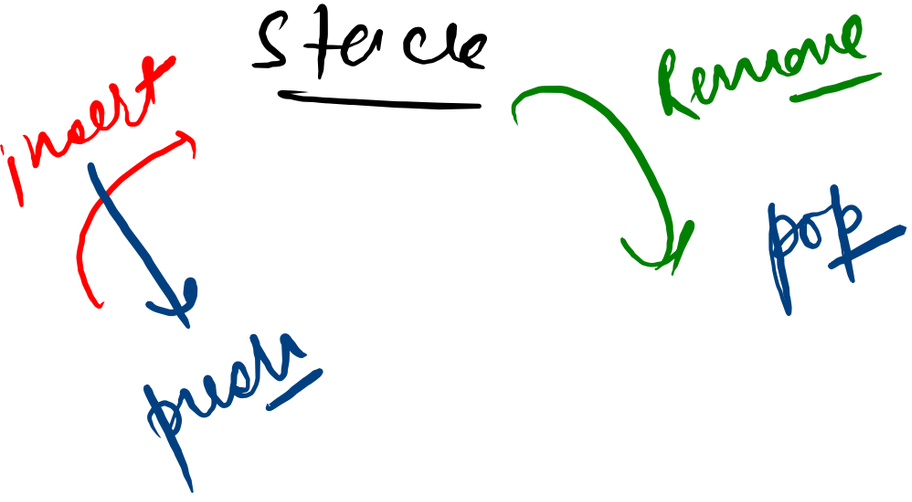
stack kind data structure

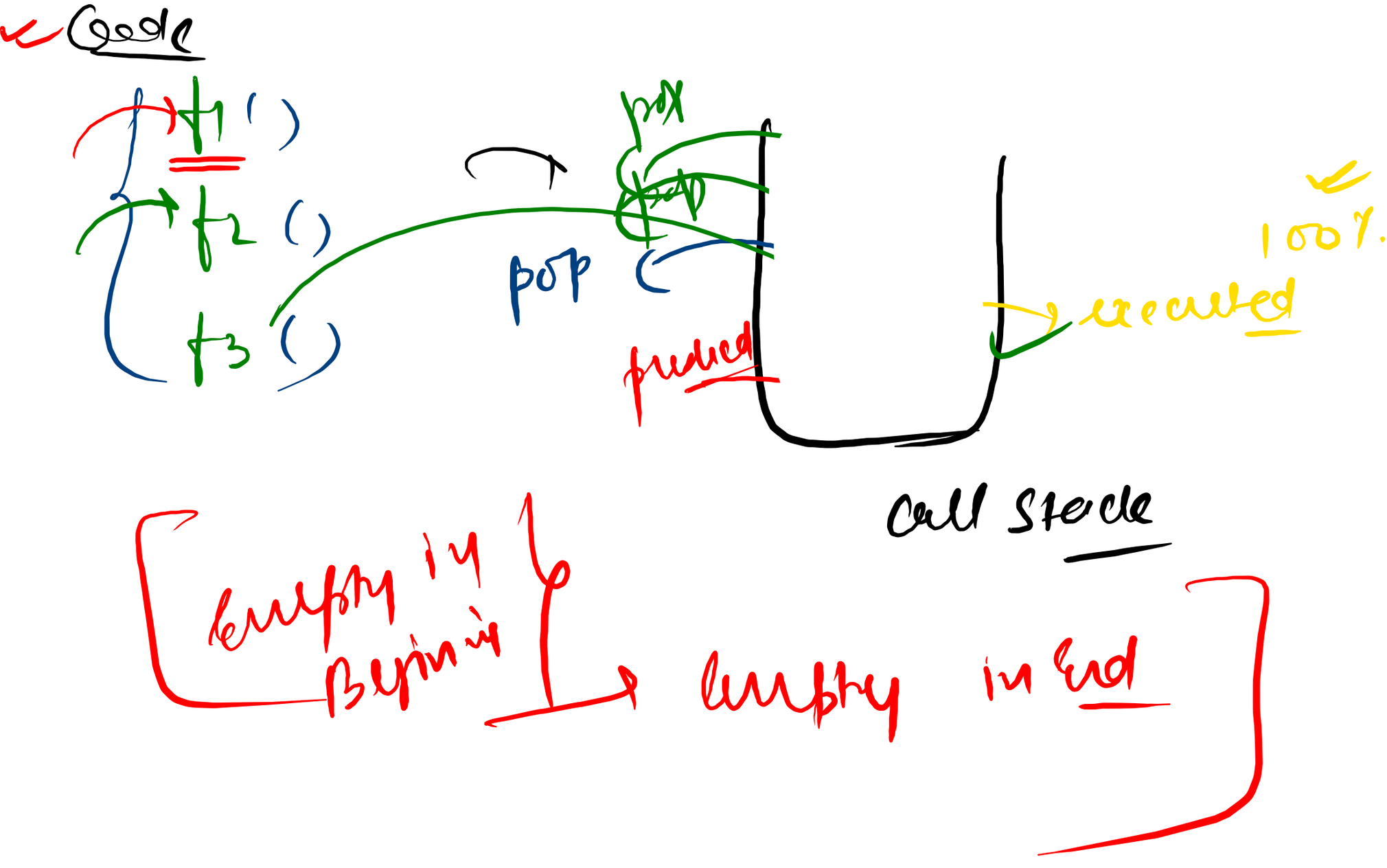
↓

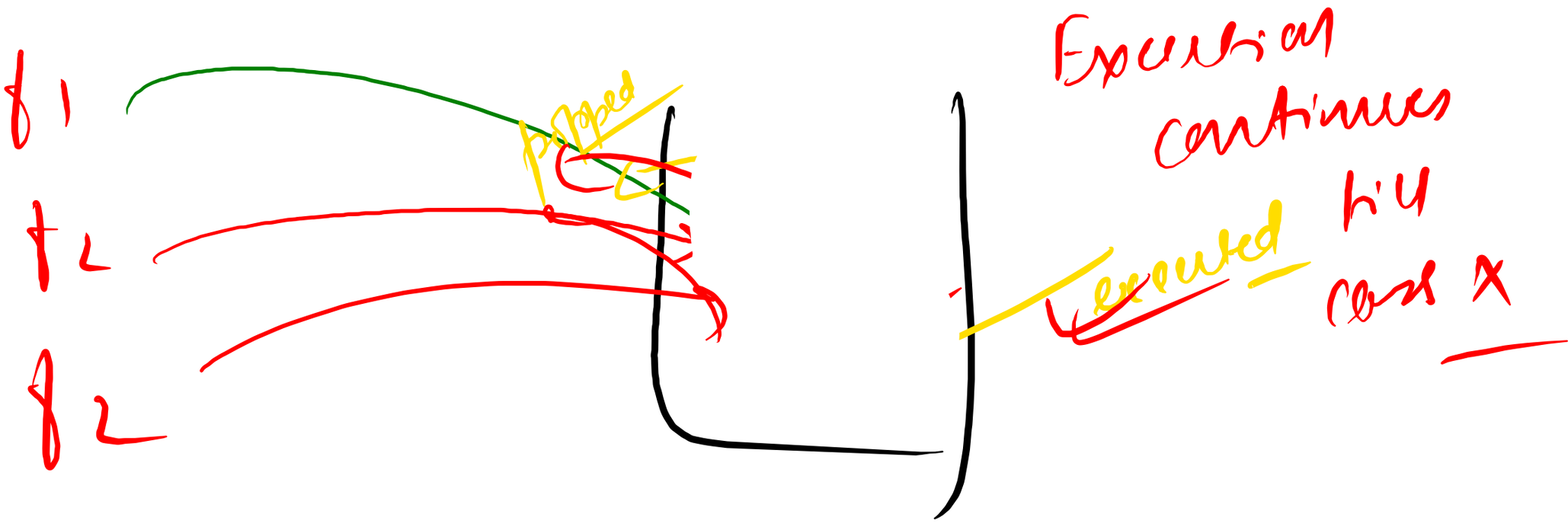
FILO / LIFO


Restaurant
5
4
3
2
1
Stack
LIFO
FILO


Stack
1, 2, 3, 4
Ret
4
3
2
1


Stack
insert
push
Remove
pop

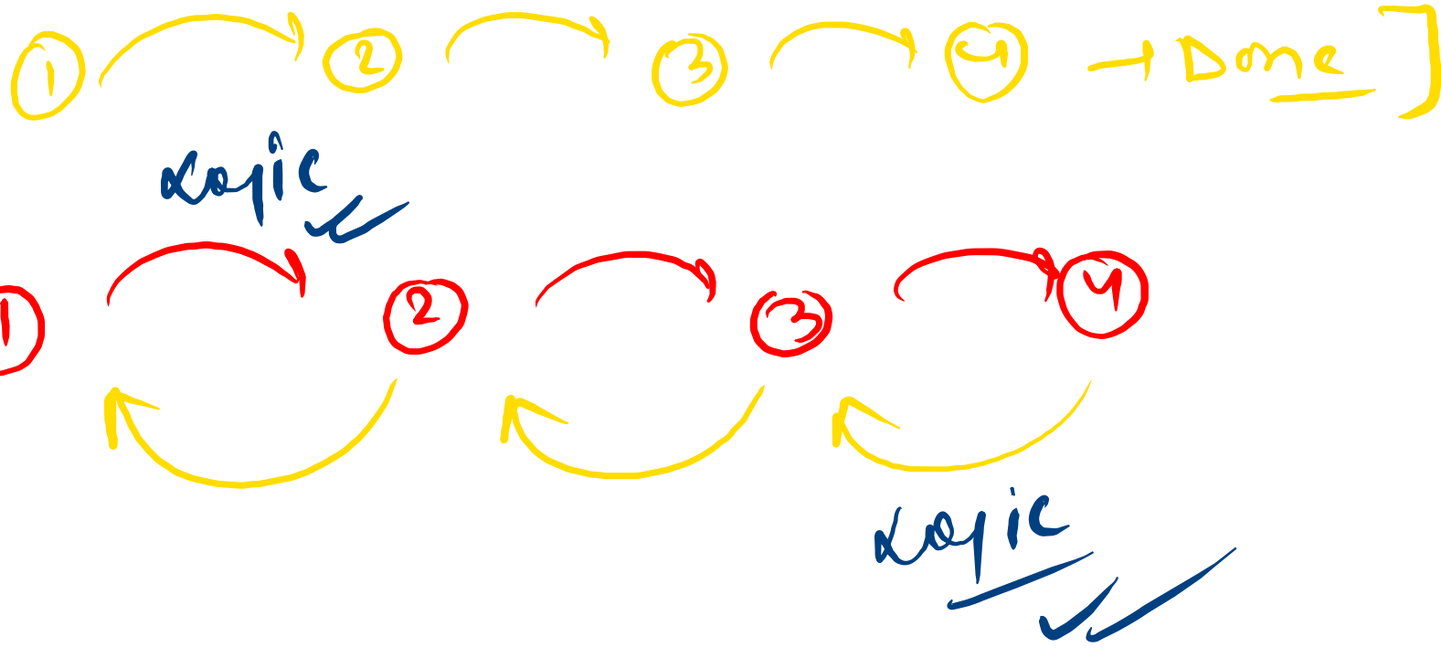

Code
f1 ()
f2 ()
f3 ()
pop
pop
pushed
100%.
executed
Call Stack
empty in
Begin
empty in End


f1
f2
f2
Dropped
Execution continues
Executed by cpu x

for Loop / while

↓

Recursion



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

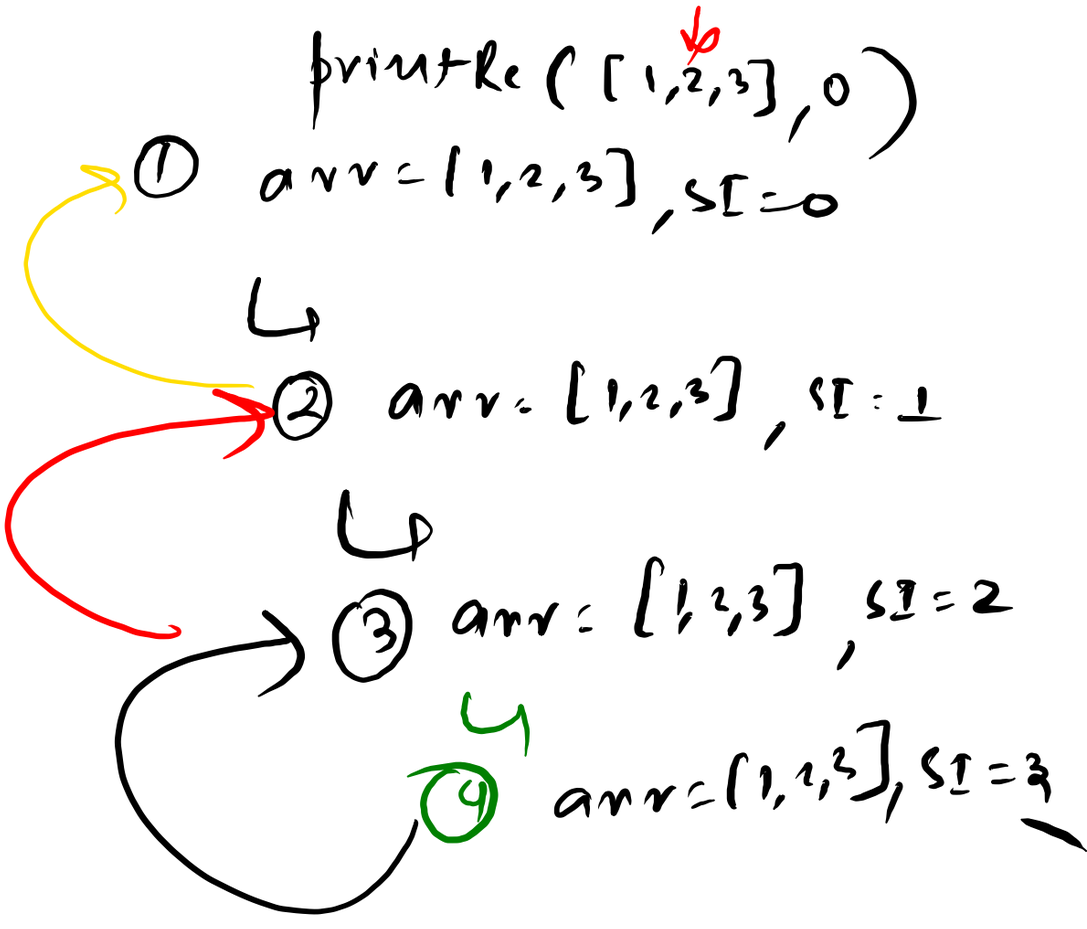
Merge / Quick

$O(n \log n)$

```python
def printRec(arr, sI):
  #Base condition
  if sI >= len(arr):
    return

  #Recursive call
  printRec(arr,sI+1)


  #Logic
  print(arr[sI])
```

printRec([1,2,3,4],0)

3, 2, 1



10:35]

[3, 5, 2, 1, 4]

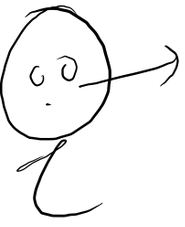
↳ find the smallest no
and remove

↳ 5 minutes]

[3, 5, 1, 2, 4]

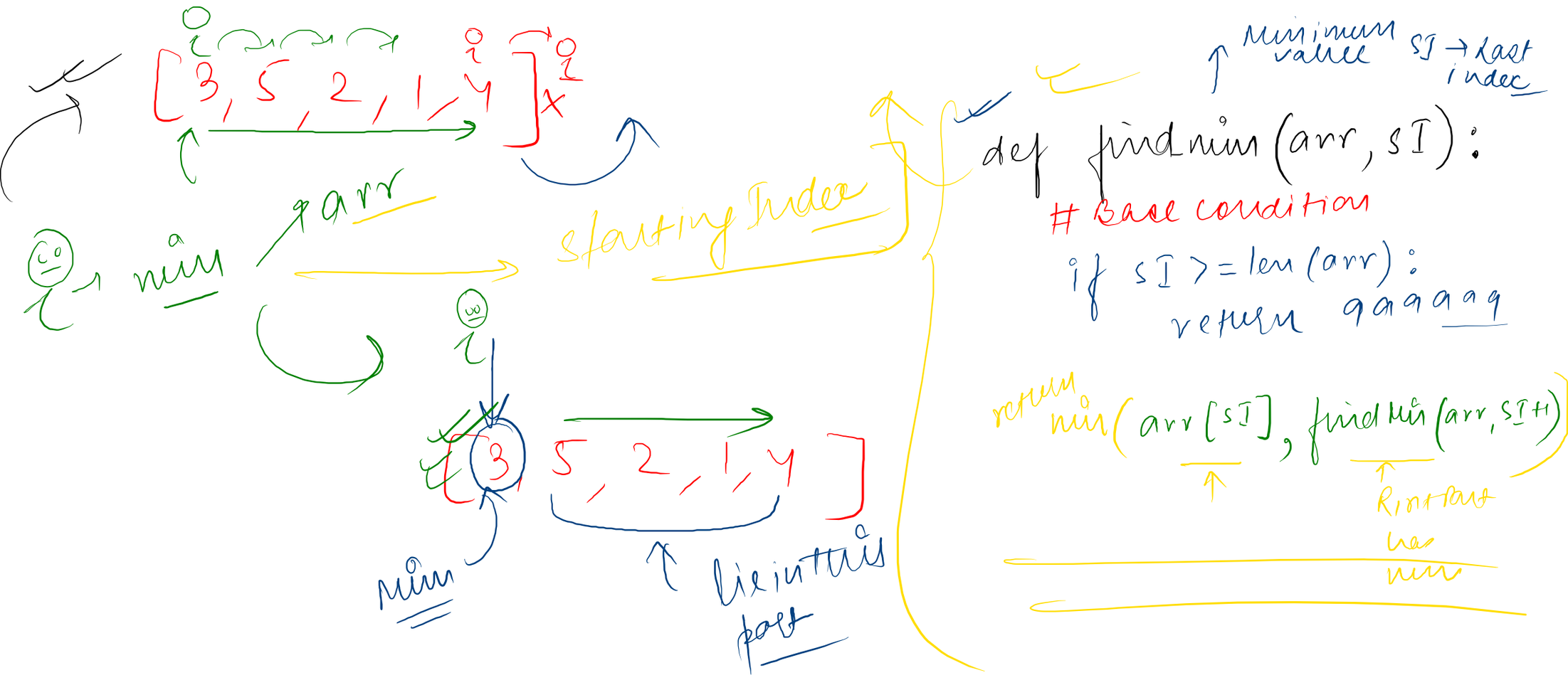
Minimum ?

↳ Recursion !

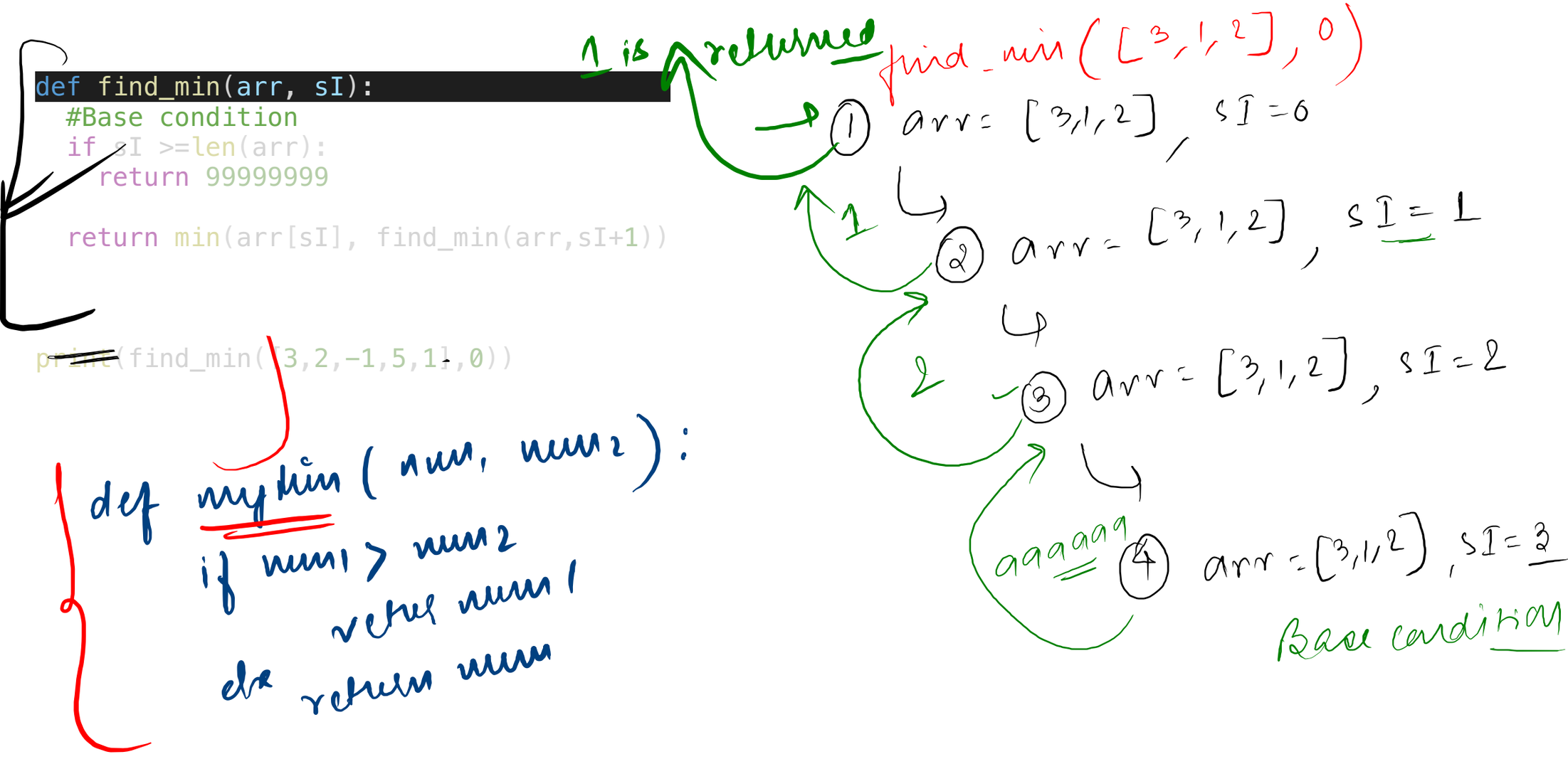


define a function { → name

argument / definition]


3, 5, 2, 1, 4
arr
starting index
min
min
5, 2, 1, 4
list in this part
minimum value
SI → last index
def findmin(arr, SI):
# Base condition
if SI >= len(arr):
return 99999
return min(arr[SI], findMin(arr, SI+1))
Recursive call

```python
def find_min(arr, sI):
  #Base condition
  if sI >=len(arr):
    return 99999999

  return min(arr[sI], find_min(arr,sI+1))


print(find_min([3,2,-1,5,1],0))
```




```
def mymin(num1, num2):
    if num1 > num2
        return num1
    else return num2
```

11:15 PM } I'll be connect back

Recursion Problem in Recursion

BREAK TIME

Let's start at 11:15 AM

Recursion

Pr

Loops

300 lines of code

Recursion

3 lines of code

99x

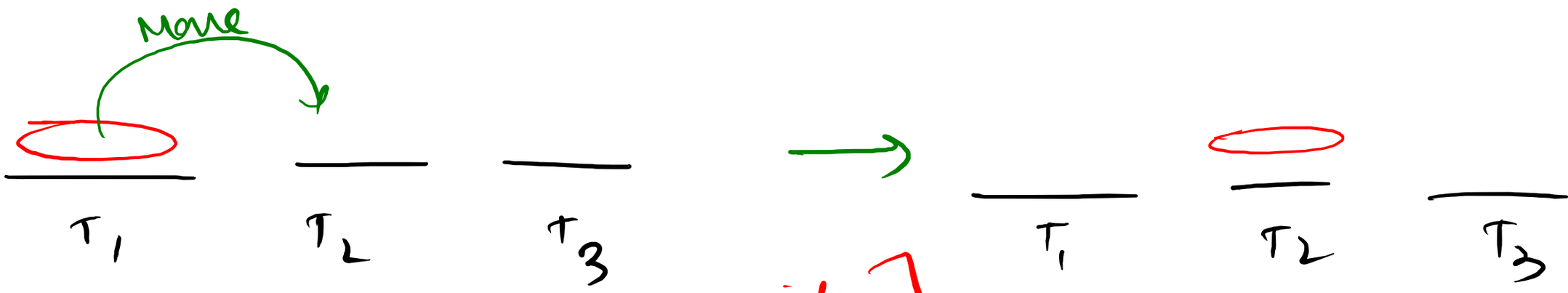
time

single 20

well form

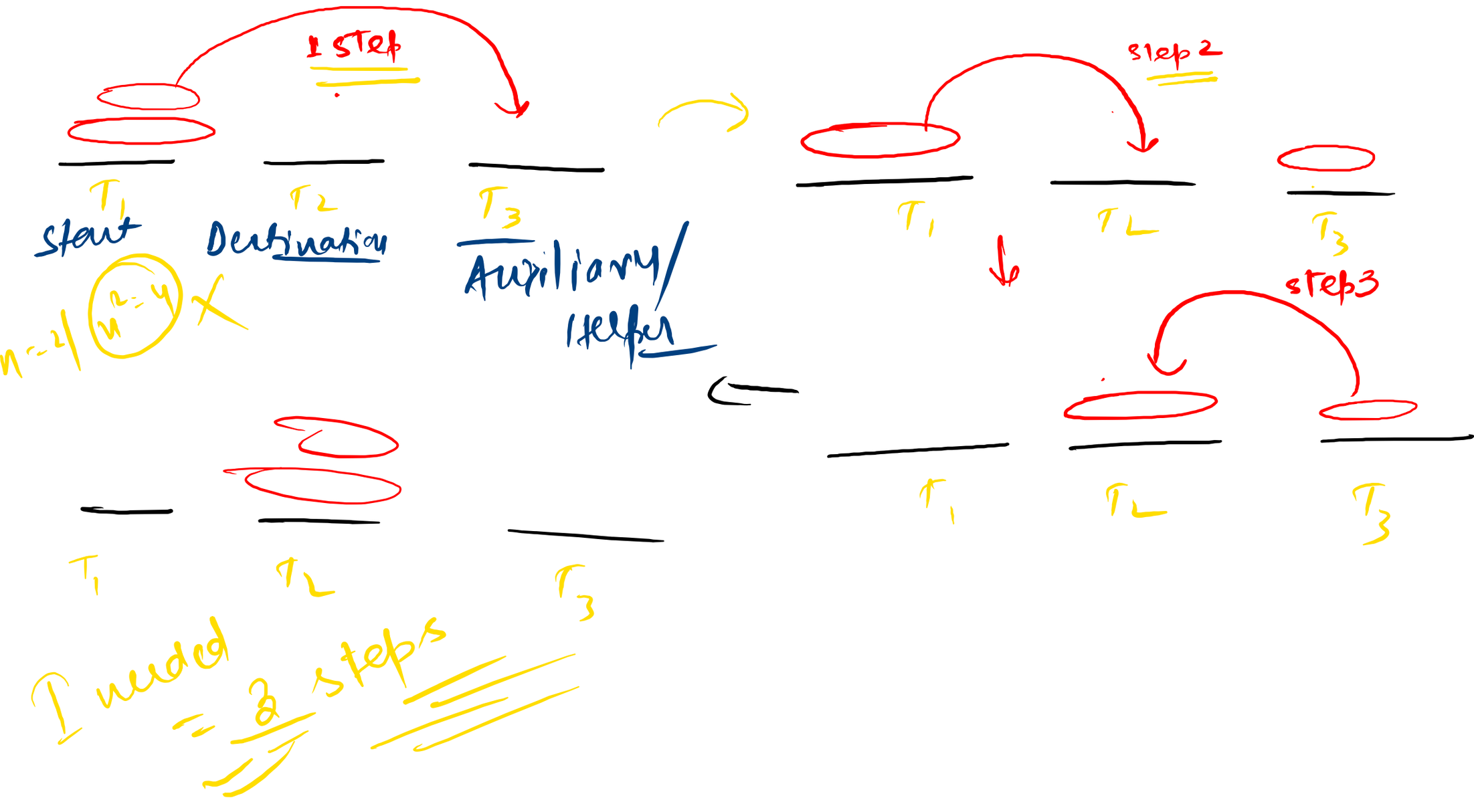
Town of Hamsey

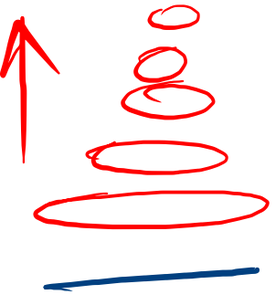

n
① Decreasing size
T_1
T_2
T_3
② At one time only 1 disk can be moved
T 1
T_2
T_3
How many moves will be required



single move

No of steps = 1

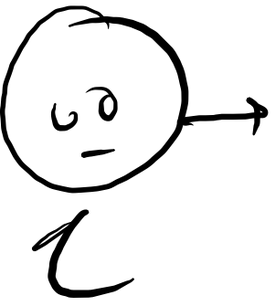

1 step
T1
T2
T3
start
Destination
Auxiliary/
Helper
n-2
n-1
n
X
step 2
T1
T2
T3
step 3
T1
T2
T3
T1
T2
T3
I needed
= 3 steps



T_1 T_2 T_3

Start Destination Auxiliary

Manually ✗



Problem

↳ smaller problem

Recursion


n
(n-1)
S1
S
D
A
Total steps
S1 + 1 + S2
Move n coins
1
n-1
n

moves (n , s , d , a)

↳ n coins, s → d using a → No of moves required

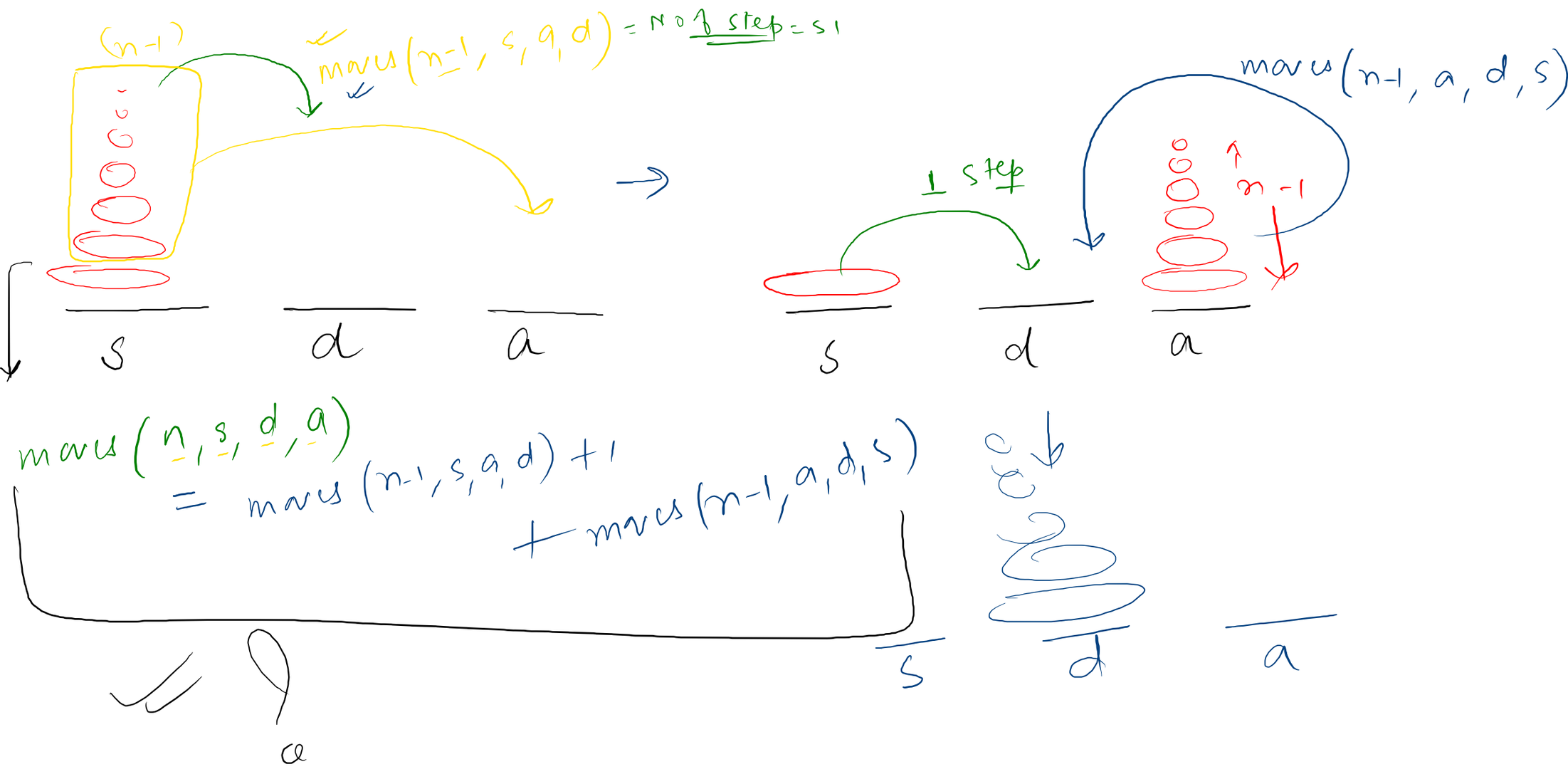
moves (5 , s , d , a)

↳ moves (n , s , a , d)

n coins s → a using d → No of moves

moves (n-1 , s , a , d)

↳ (n-1), s → a using d → No of moves


(n-1)
moves(n-1, s, a, d) = N0 of step = s1
moves(n-1, a, d, s)
1 step
n-1
s
d
a
s
d
a
moves(n, s, d, a)
= moves(n-1, s, a, d) + 1
+ moves(n-1, a, d, s)
s
d
a
a

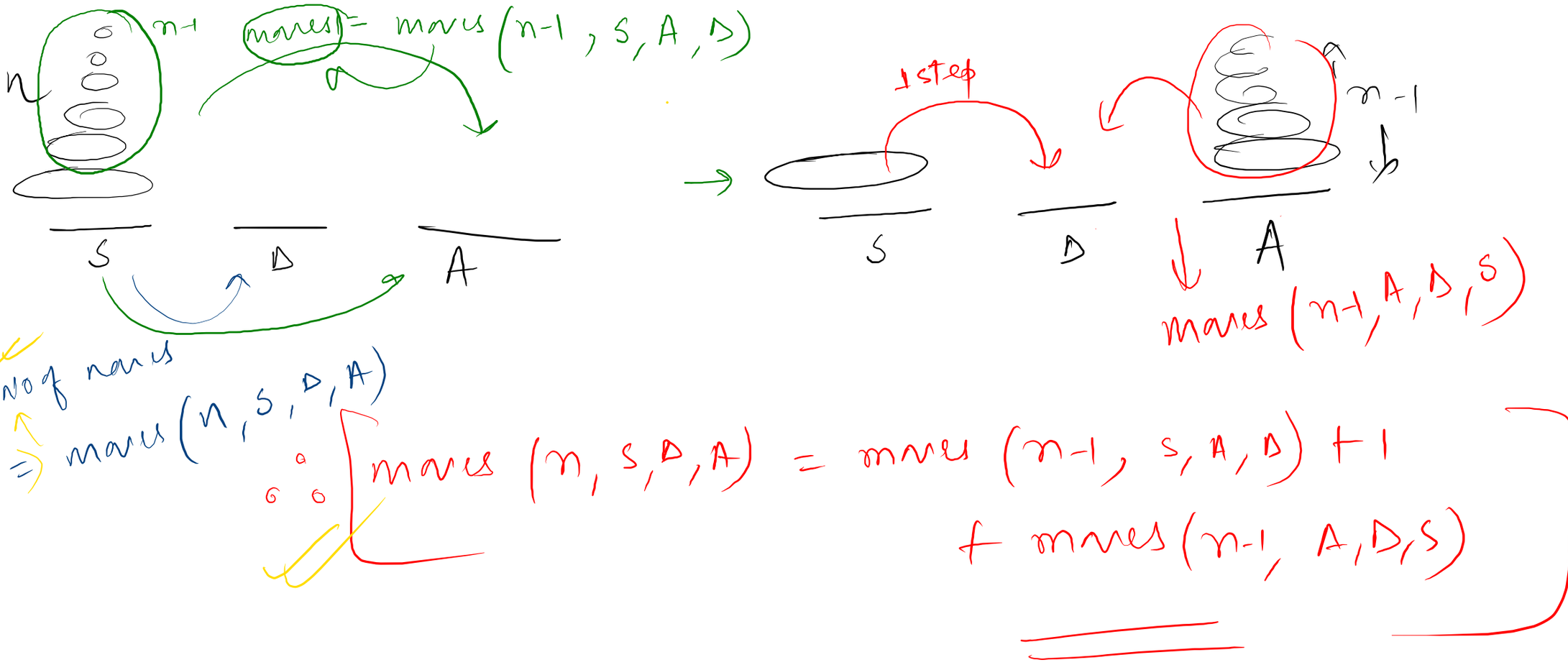
moves (n , s, d, a)

↳ n coins s → d using a → No of steps

No of moves/step ↓

moves (n-1, s, a, d)

↳ No of moves (n-1) s → a, using d



No of moves

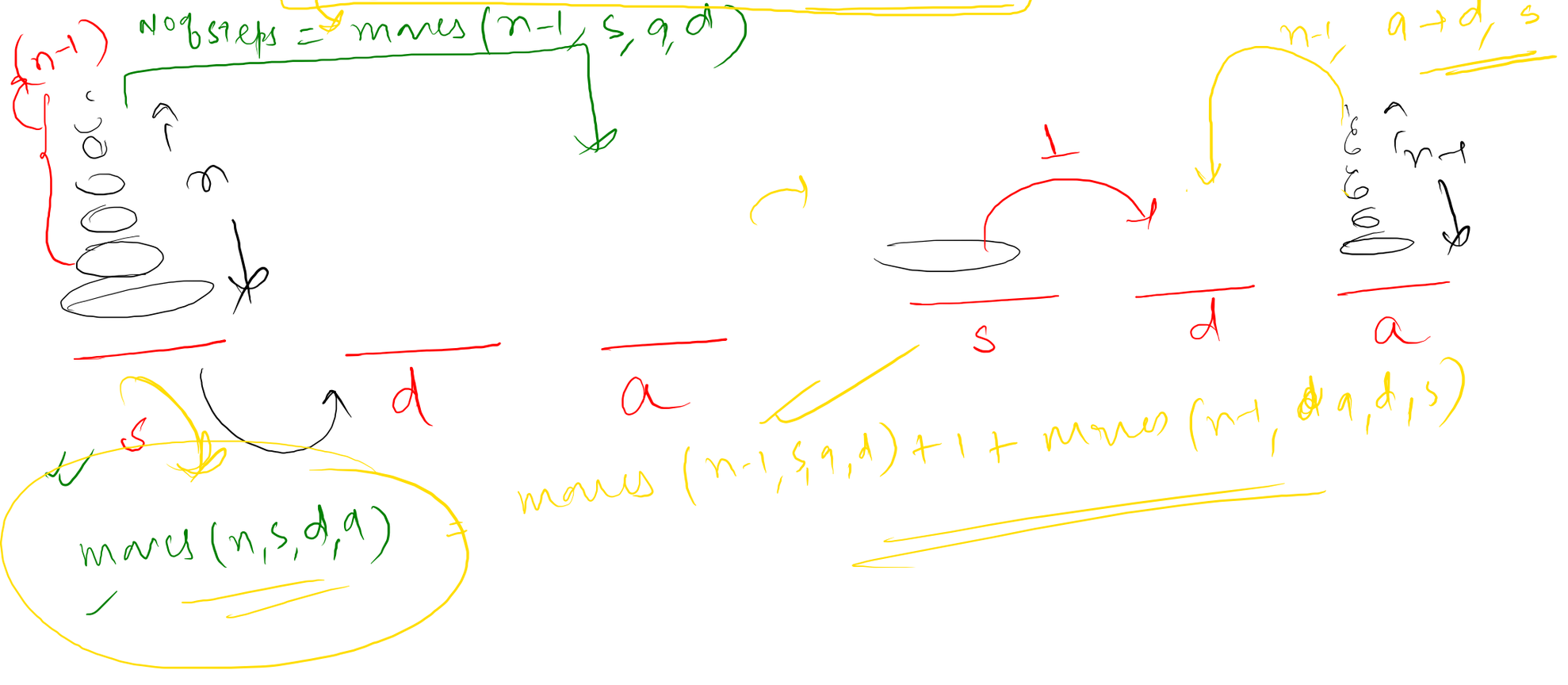
$\Rightarrow$ moves (n, S, D, A)

$$\text{moves}(n, S, D, A) = \text{moves}(n-1, S, A, D) + 1 + \text{moves}(n-1, A, D, S)$$

No of moves, n, s → d, using a

```
def moves(n, s ,d , a):
  # Base condition
  if n == 1 :
    return 1


  # move n-1 s->a , then move 1 biggest coin to s->d and then n-1 coin a-d
  return moves(n-1,s,a,d) + 1 + moves(n-1,a,d,s)
```

→ Exponential



No of moves $= 2^n - 1$

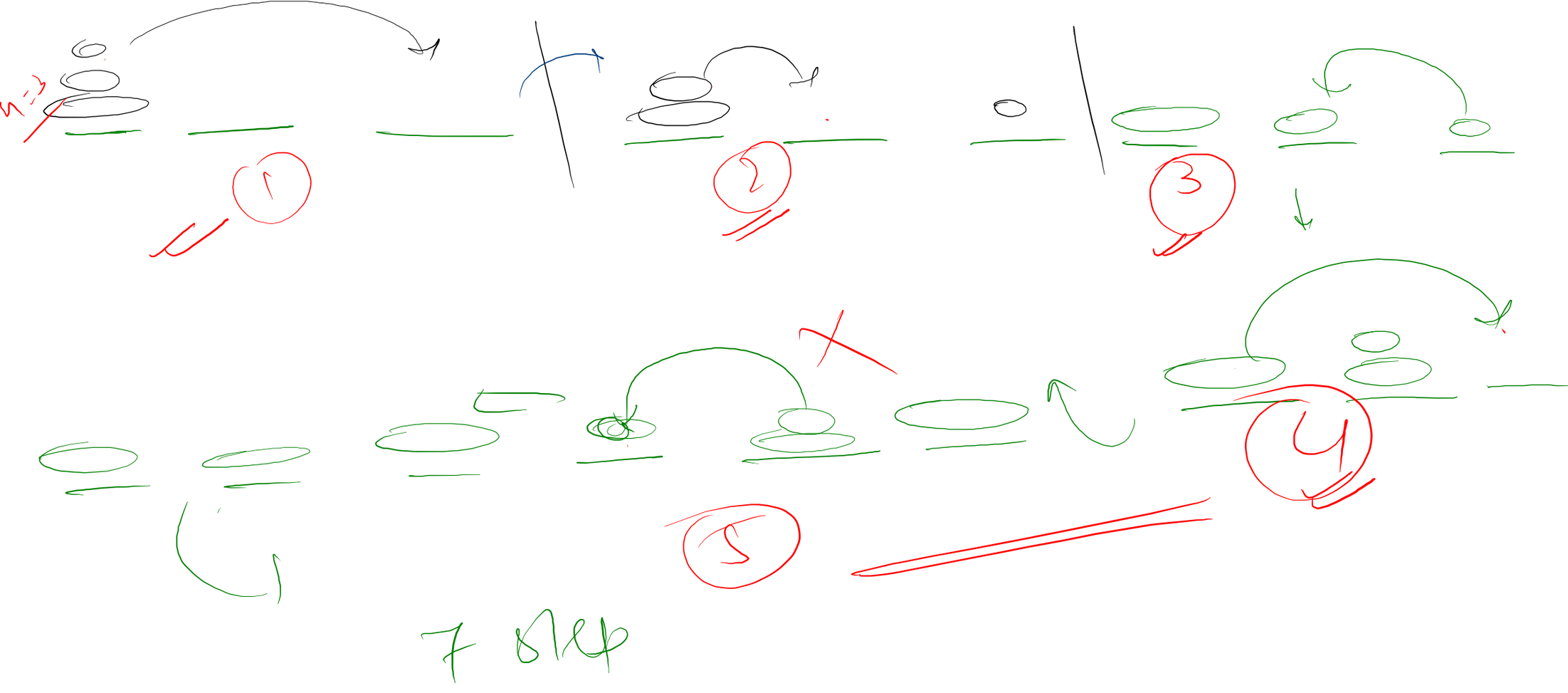
$n = 1 \rightarrow 2^1 - 1 = 1$

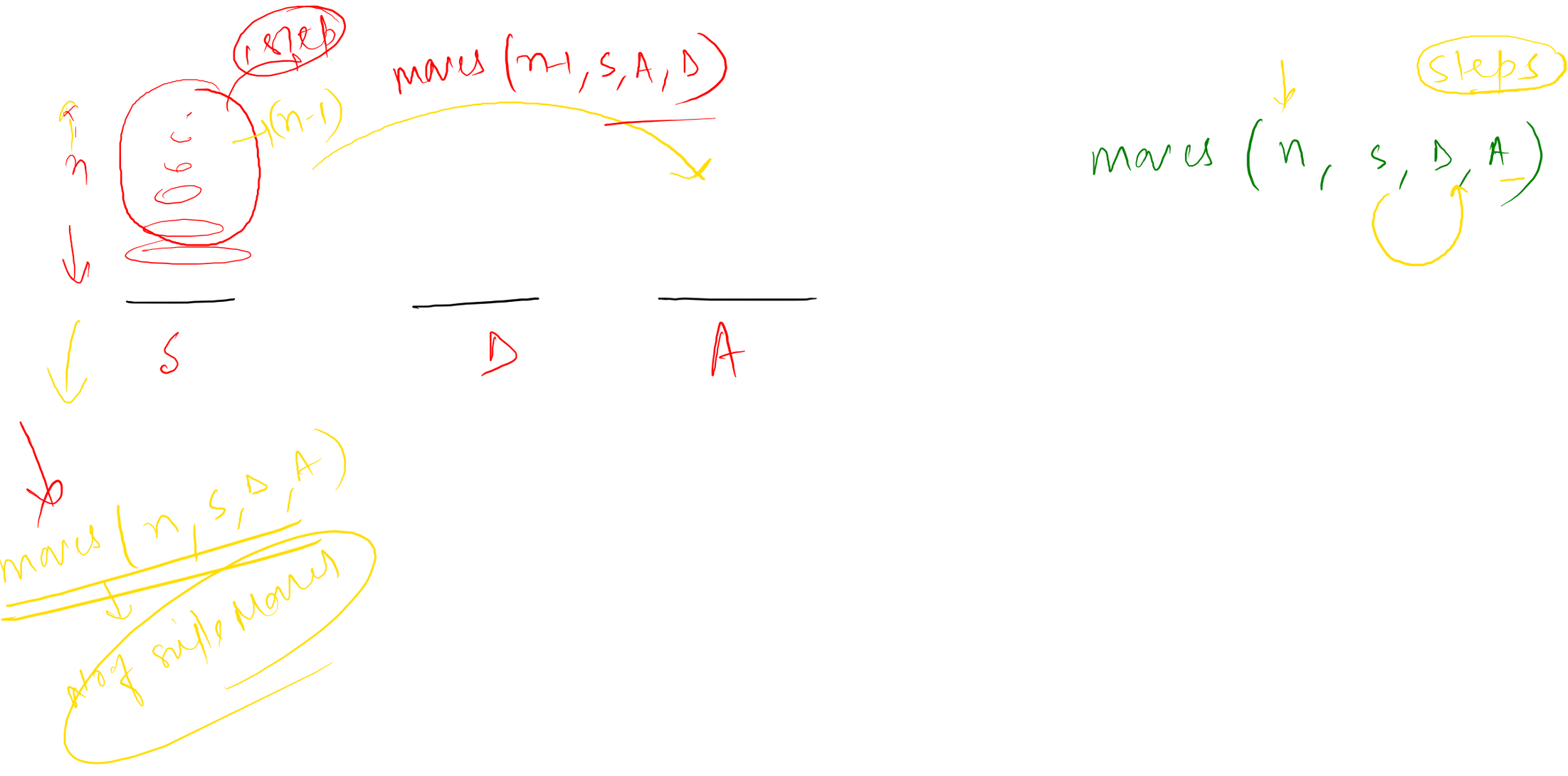
$n = 2 \rightarrow 2^2 - 1 = 3$

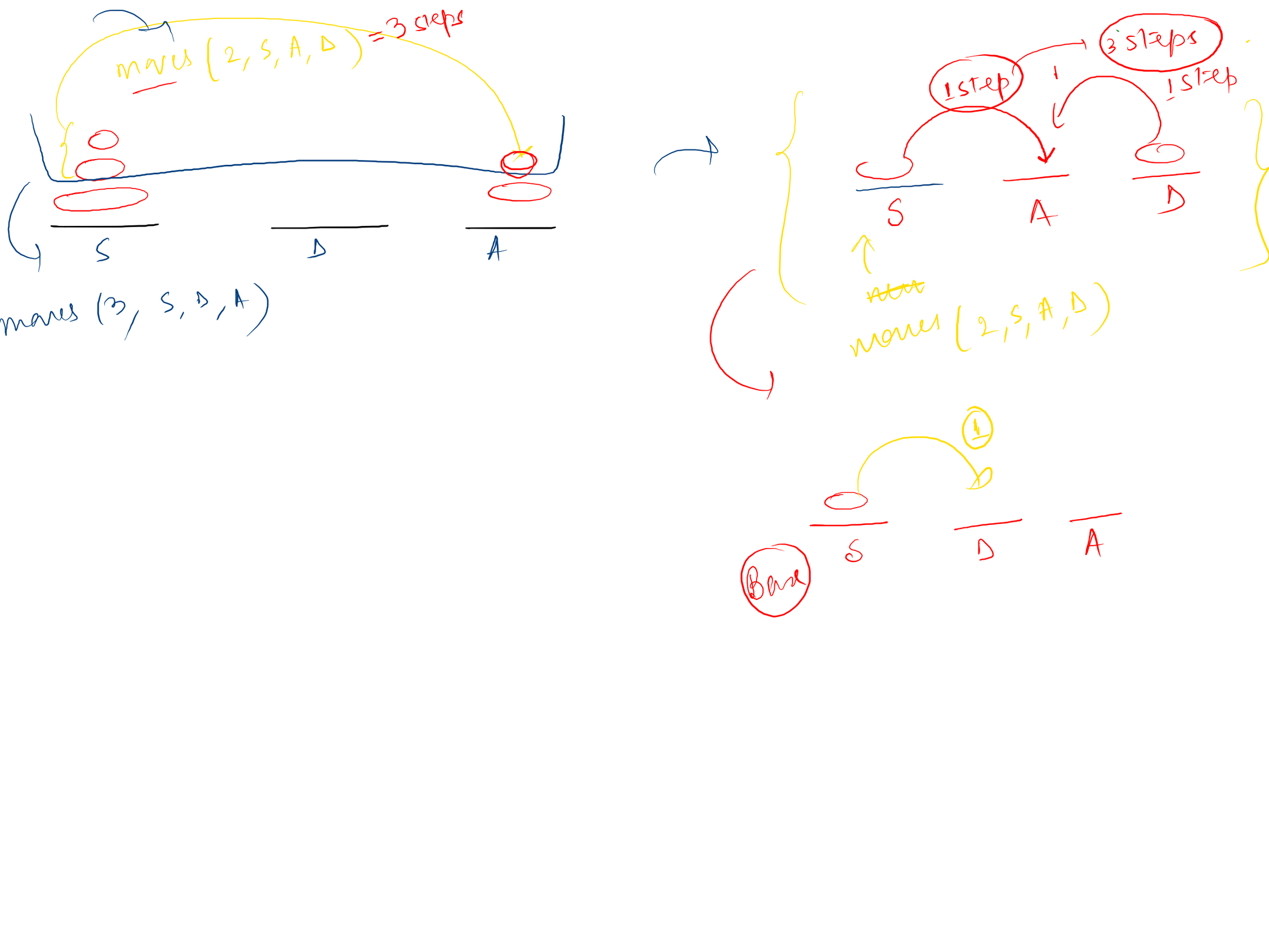
$n = 3 \rightarrow 2^3 - 1 = 7$

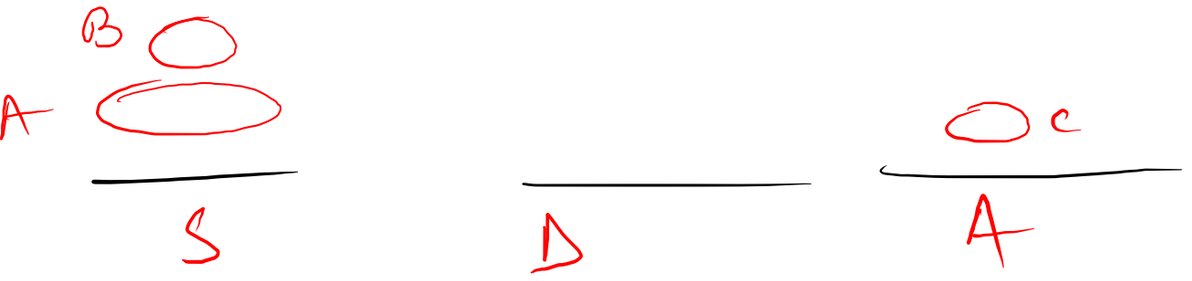
$n = 4 \rightarrow 2^4 - 1 = 15$

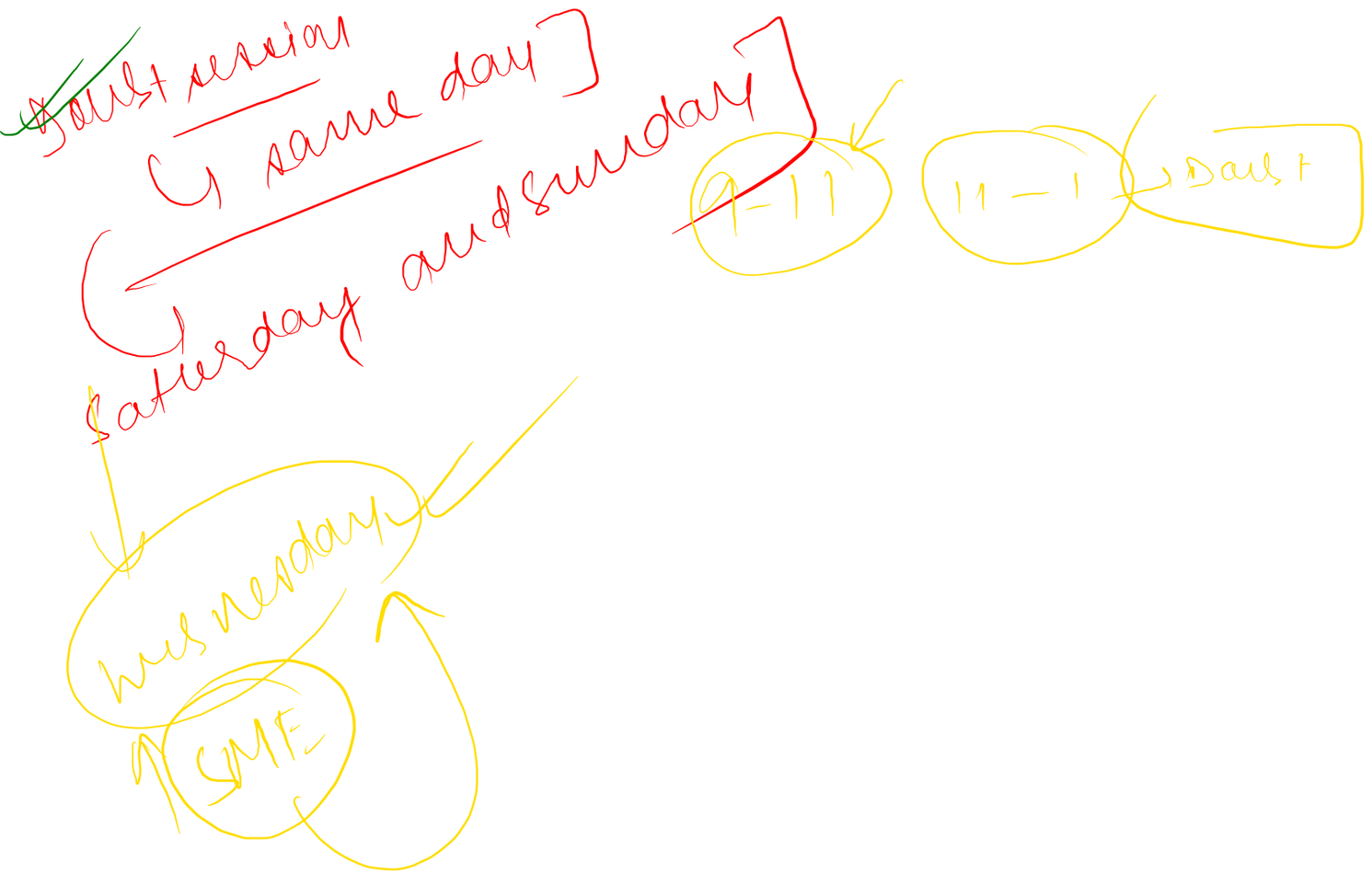
no of moves required


1
2
3
4
5
7 шаг

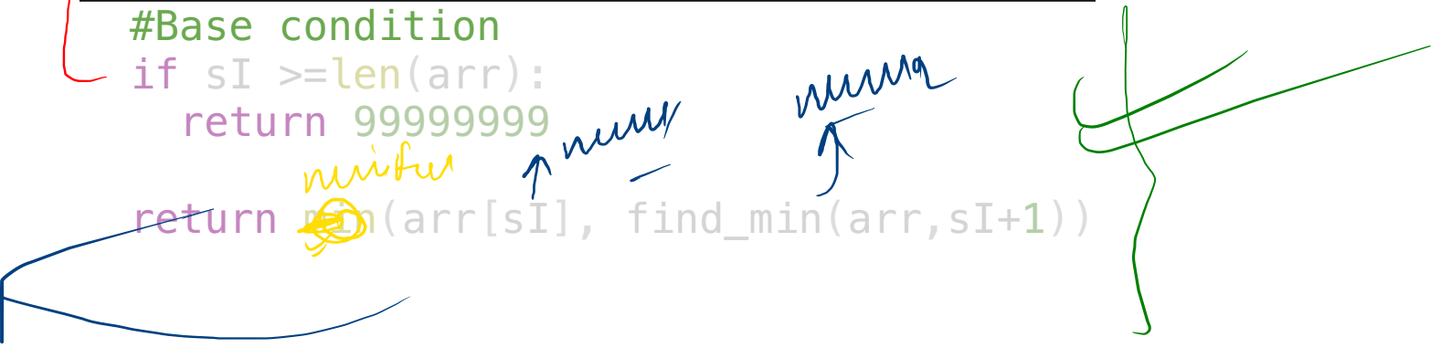

1 step
moves (n-1, S, A, D)
n
(n-1)
S
D
A
moves (n, S, D, A)
A to Z style Moves
steps
moves (n, S, D, A)


moves (2, S, A, D) = 3 steps
S
D
A
moves (3, S, D, A)
3 steps
1 step
1
1 step
S
A
D
moves (2, S, A, D)
1
S
D
A
Base


B
A
S
D
C
A


Saturday and sunday
9-11
11-1
Daily
Wednesday
SMF-

```
def find_min(arr, sI):
  #Base condition
  if sI >=len(arr):
    return 99999999
  return min(arr[sI], find_min(arr,sI+1))
```



def min fun (min1 , min2) :

if min1 < min2 :

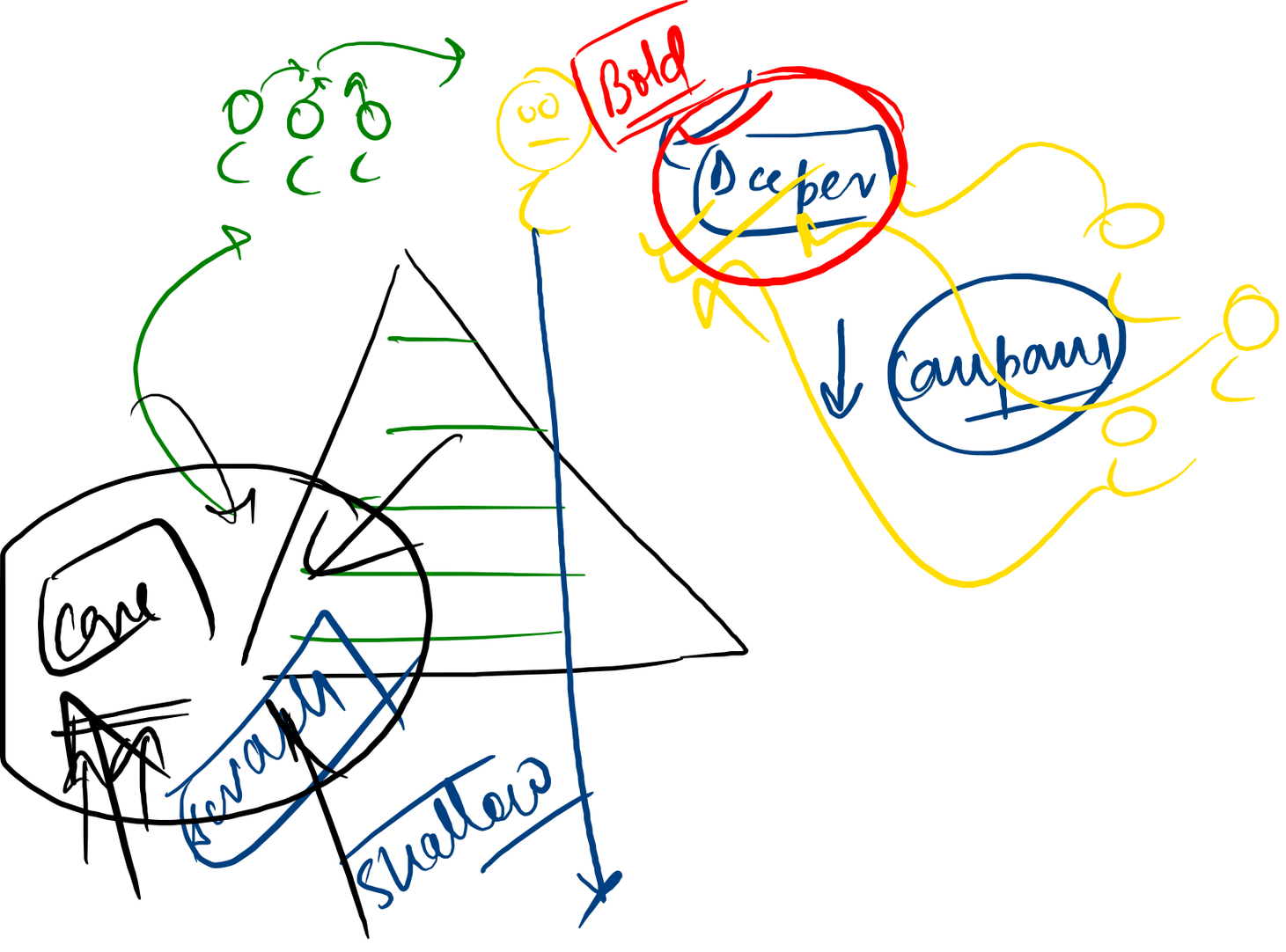
return min1

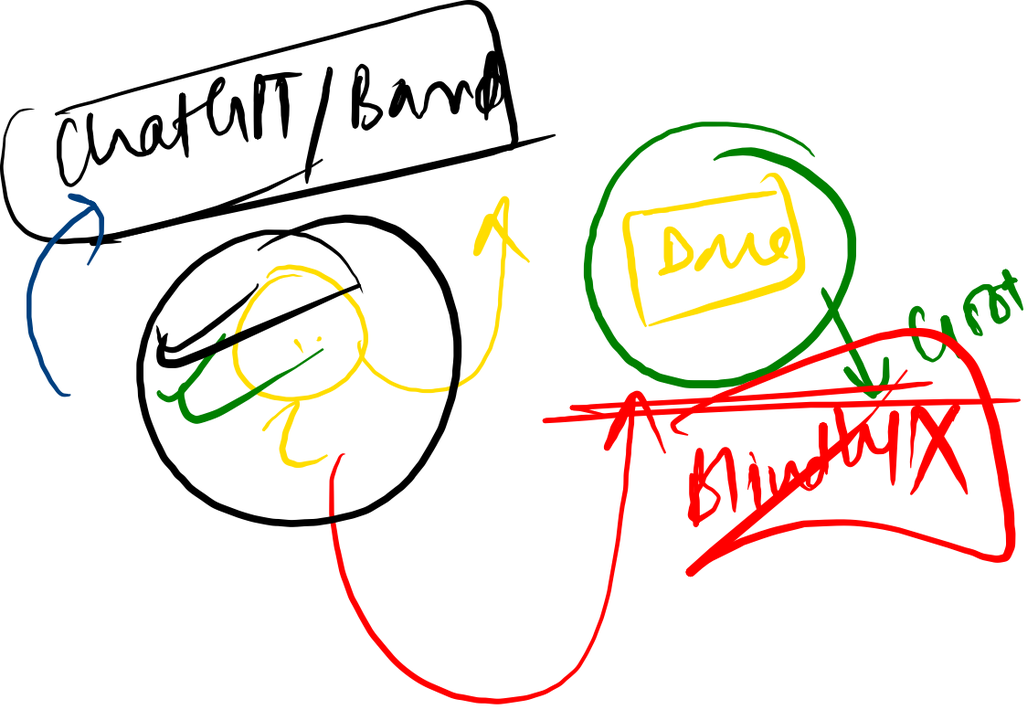
else return min2

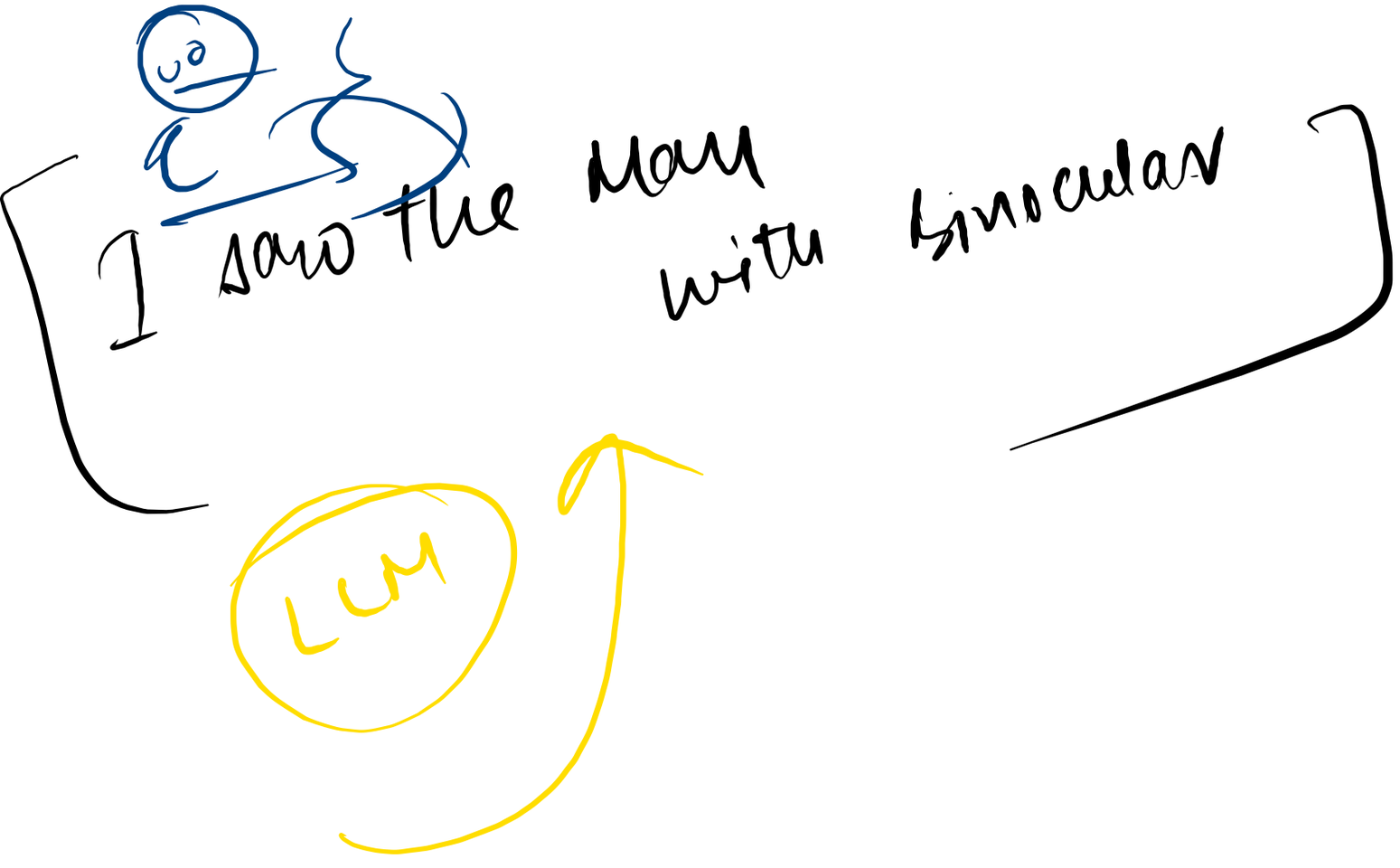
Angle

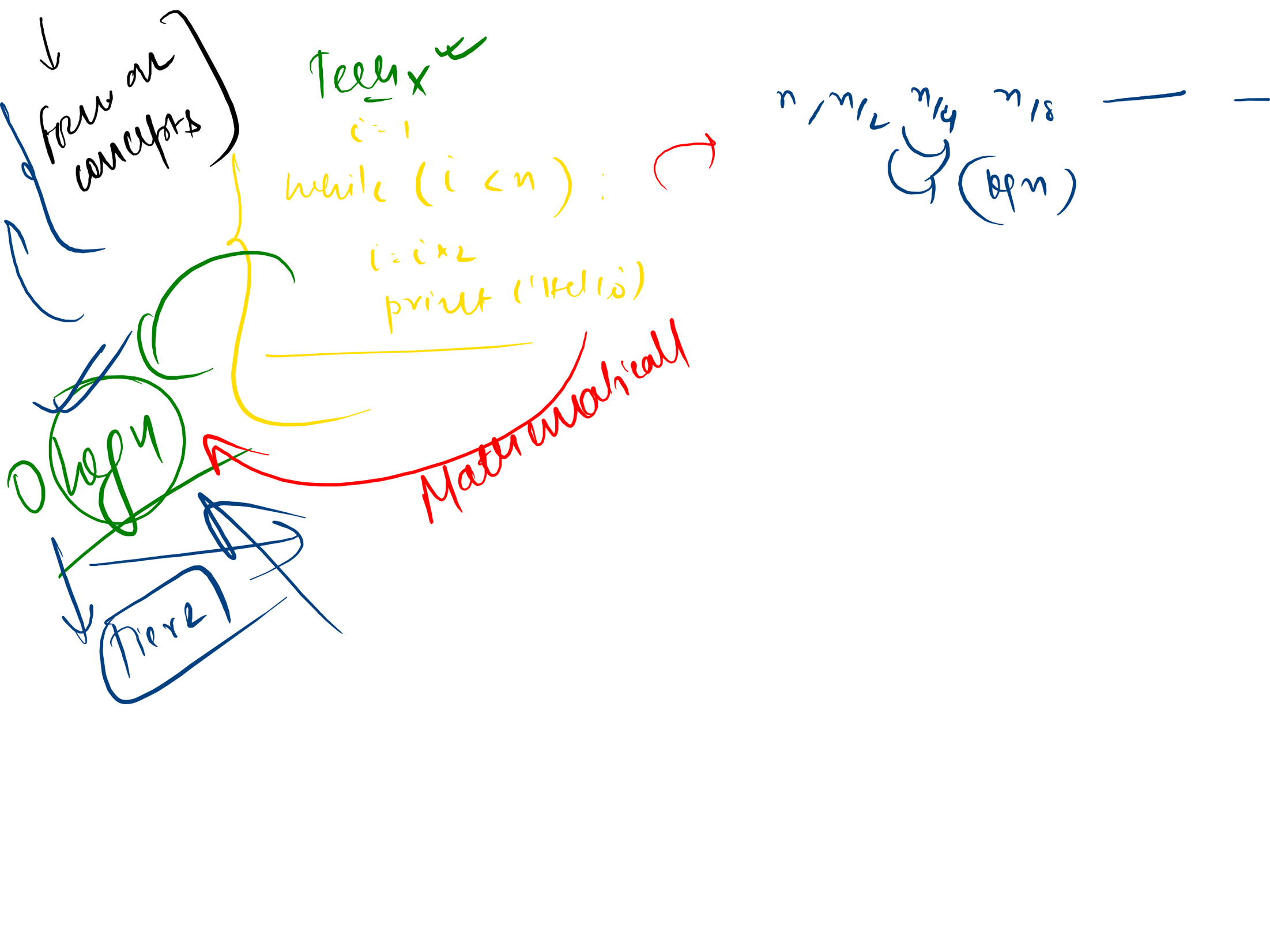
TCS/Accenture

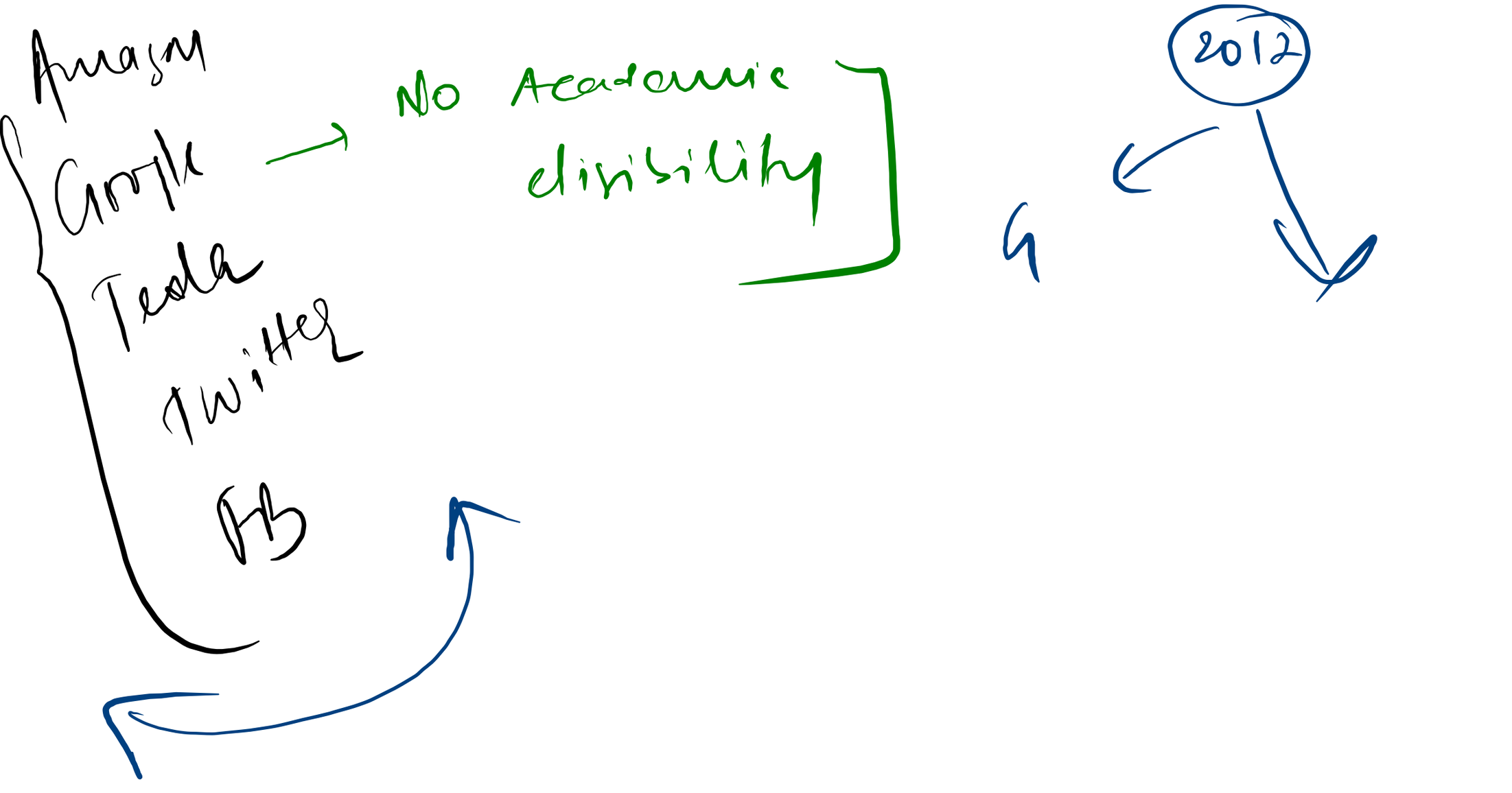

Baten
Academic
Experimental lead
9
PPT
Test
Compu

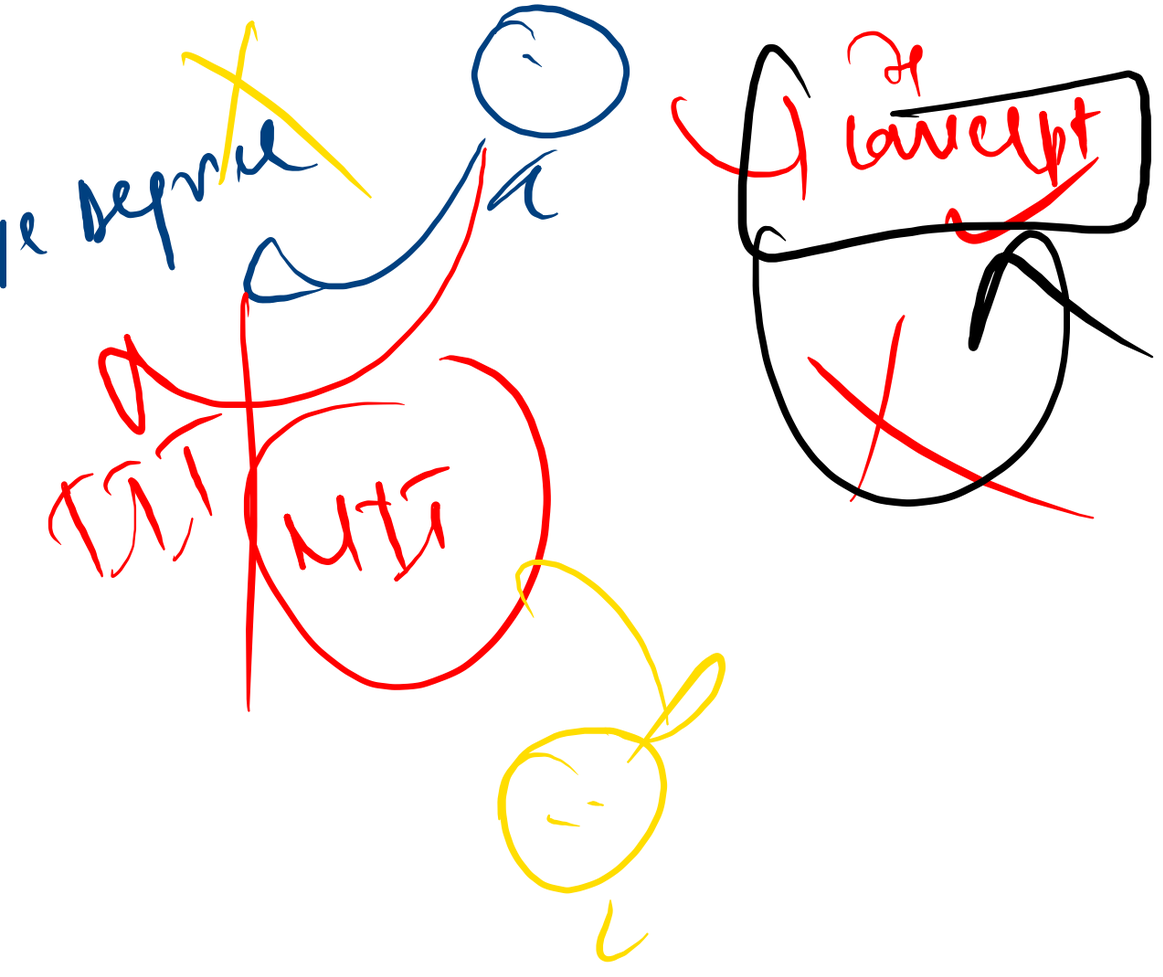

Bold
Deeper
Curious
Self
Growth
Shallow

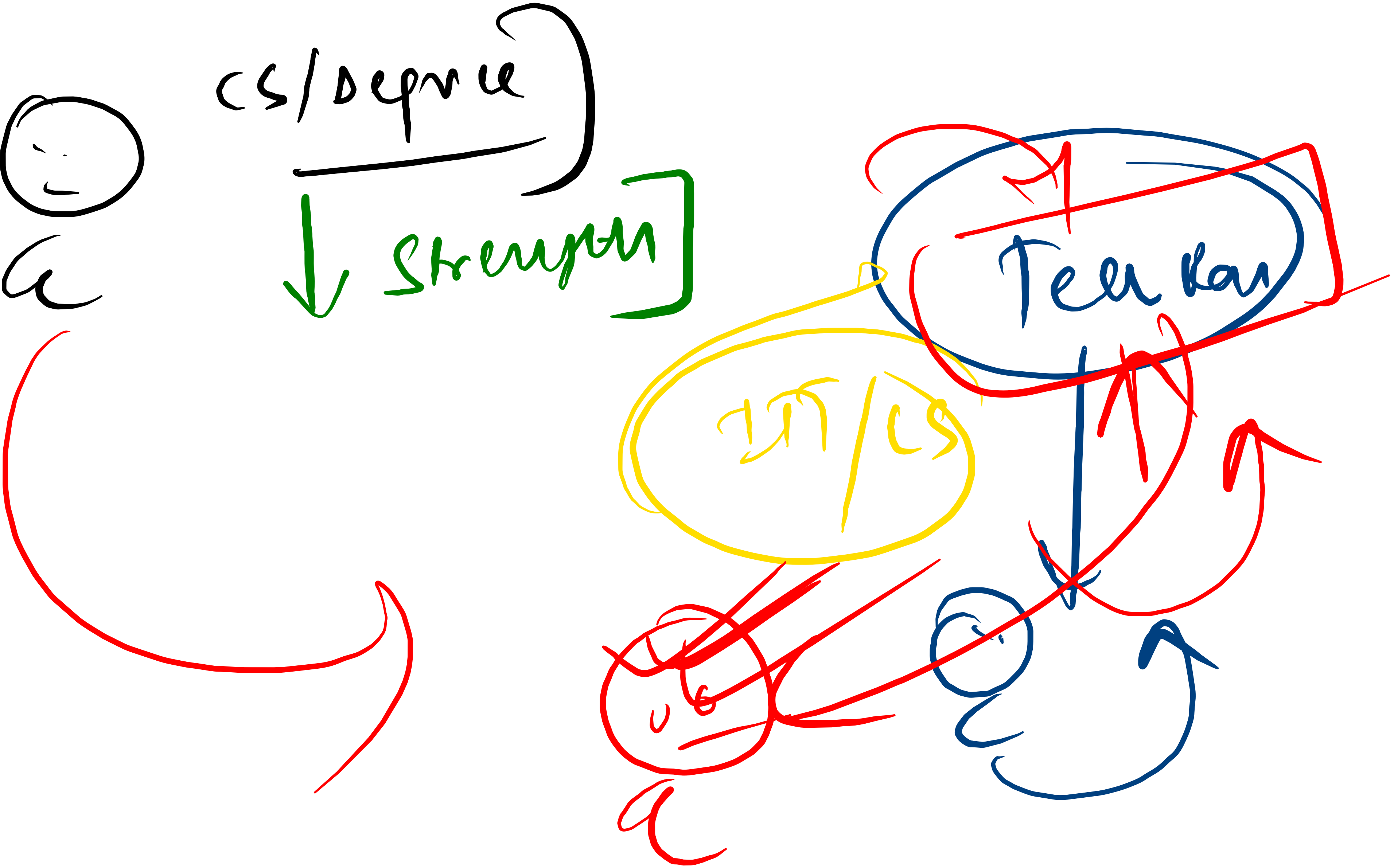

ChatGPT/Bard
Drug
Cost
Blindness

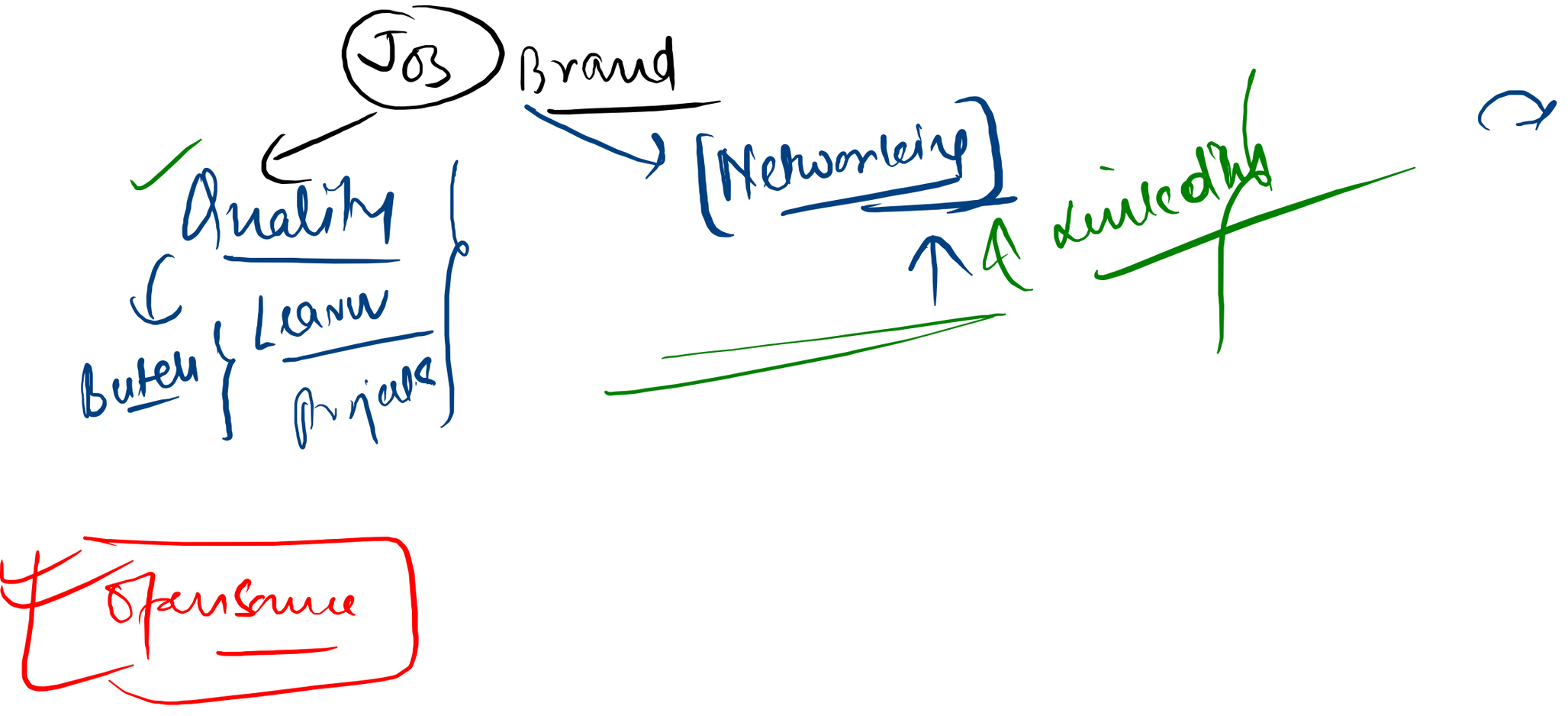

I saw the man with binocular
LM


few or concepts
Teeyx
i = 1
while (i < n):
i = i*2
print ('Hello')
n/n/2 n/4 n/8
(log n)
Mathematically
O log y
Here

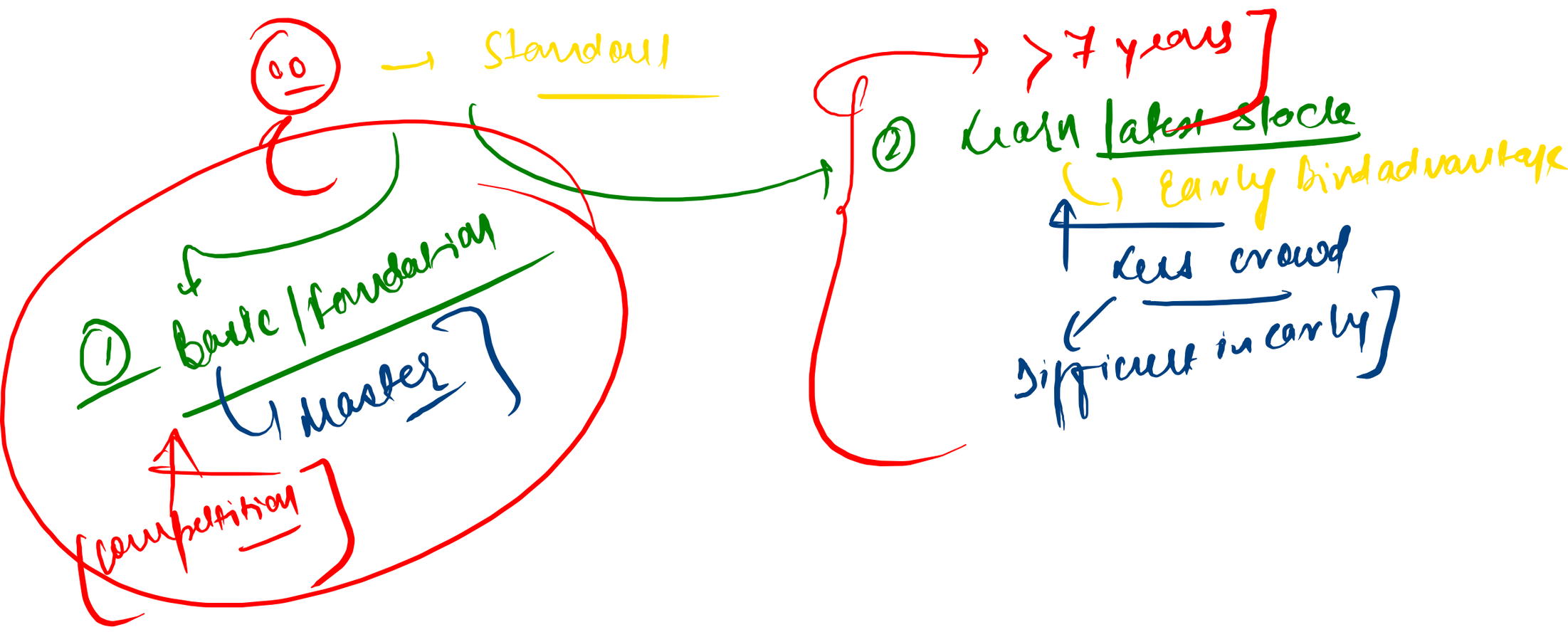

Amazon
Google
Tesla
Twitter
FB
No Academic
eligibility
2012

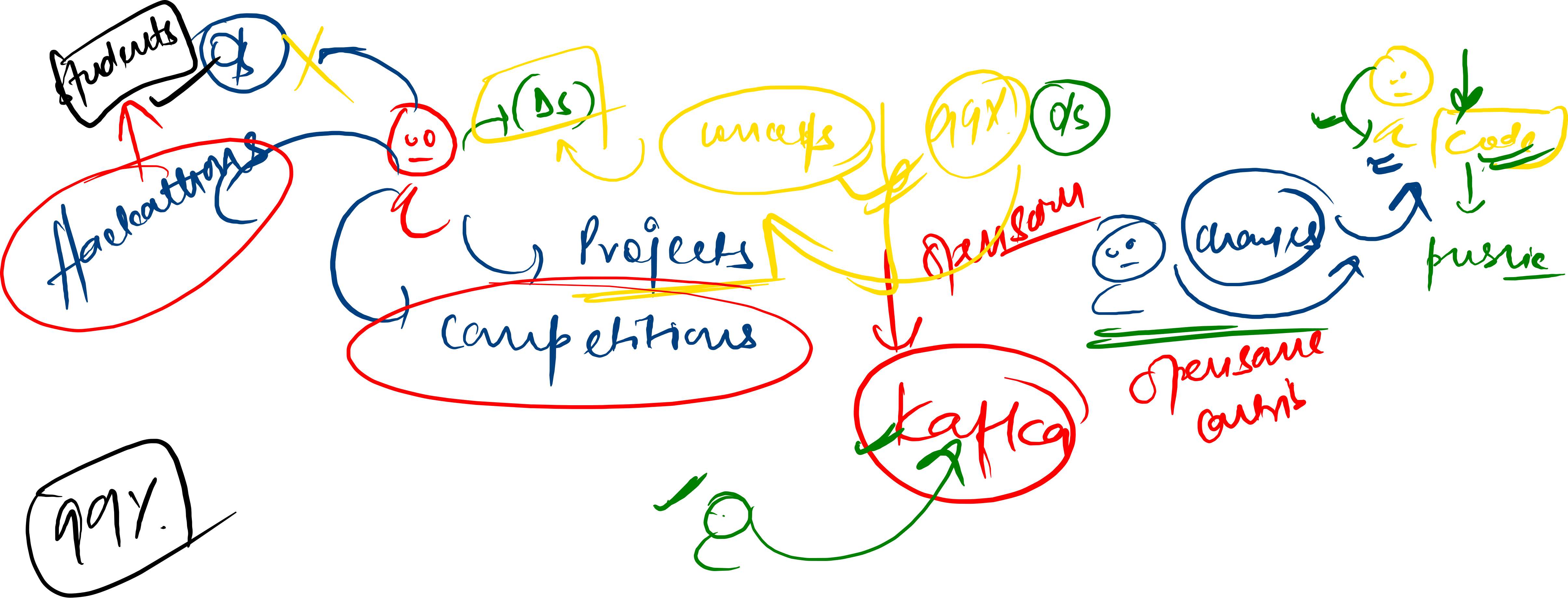

Concept


IT/CS

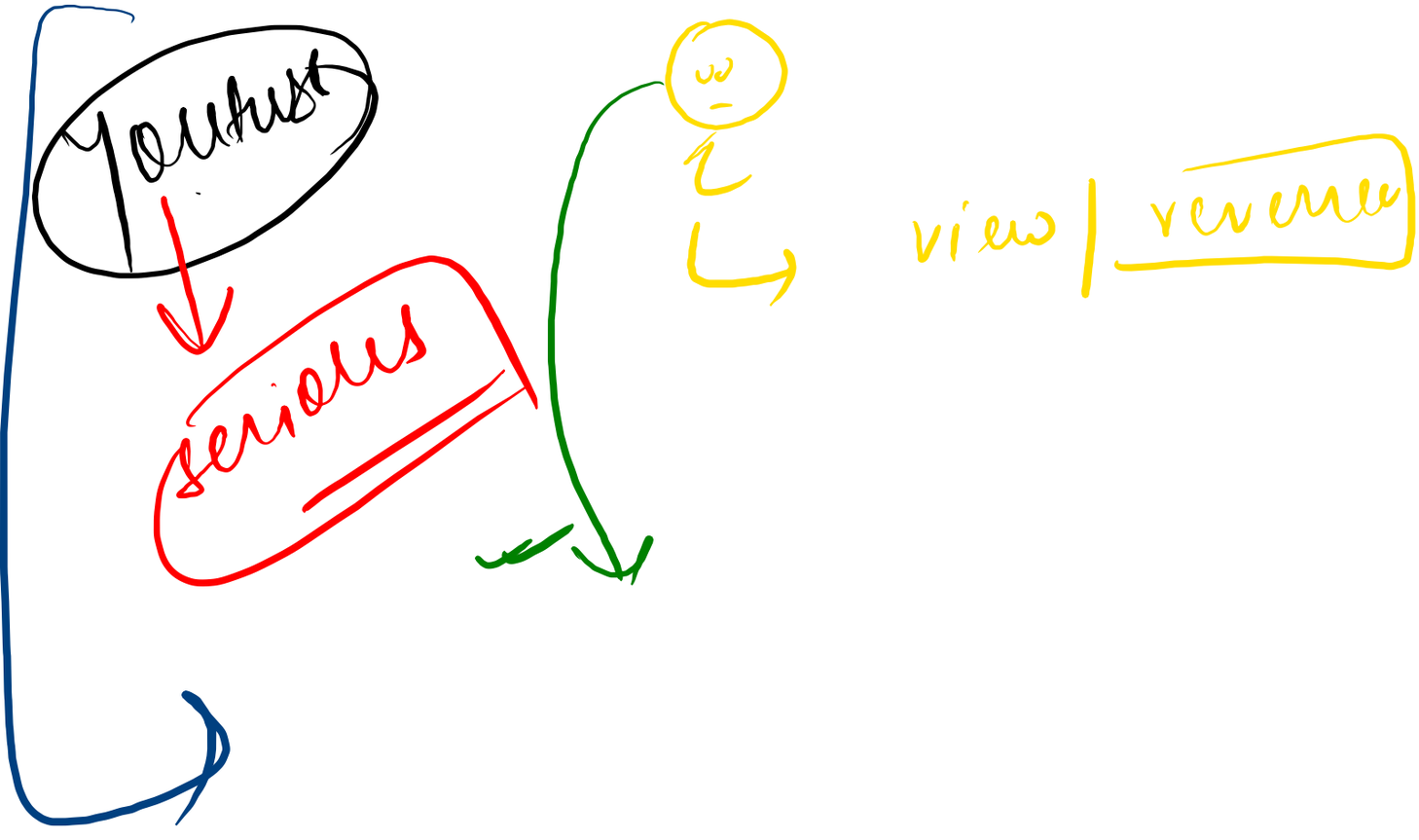


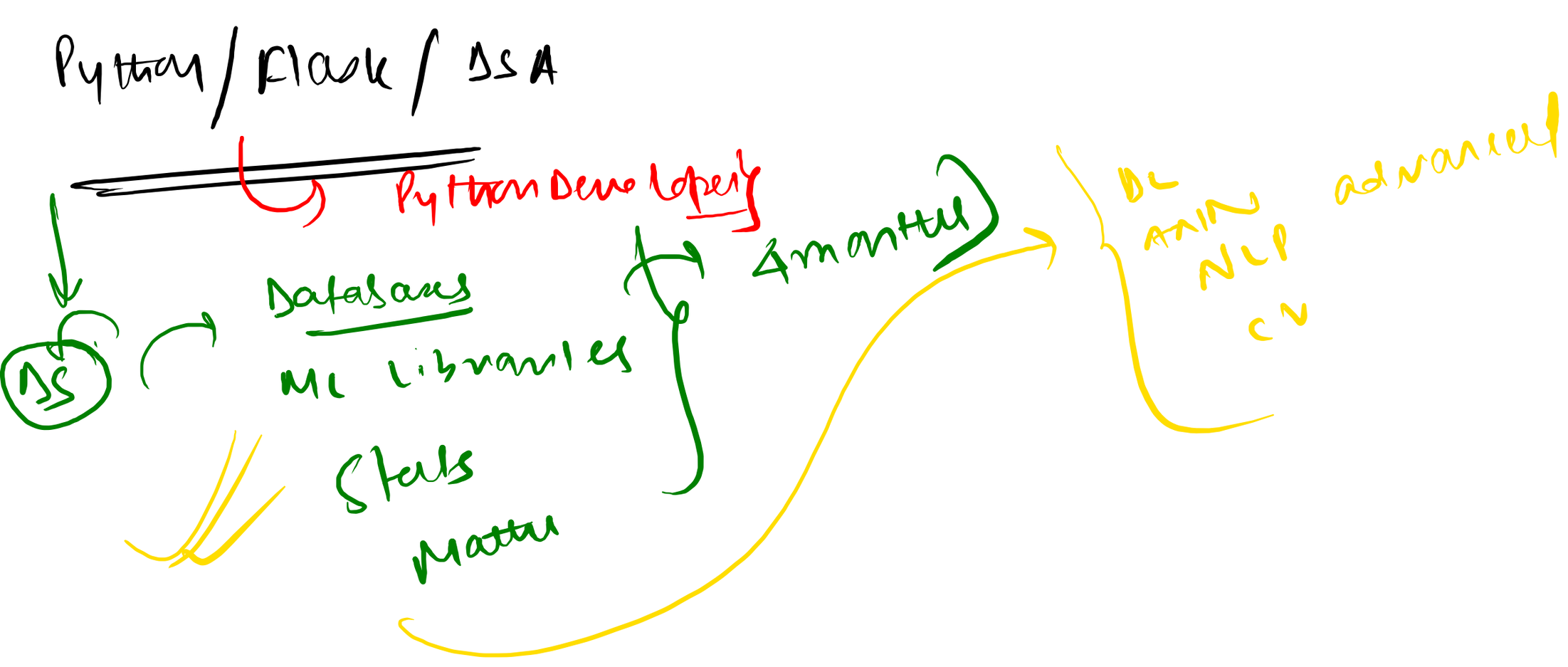
Open Source

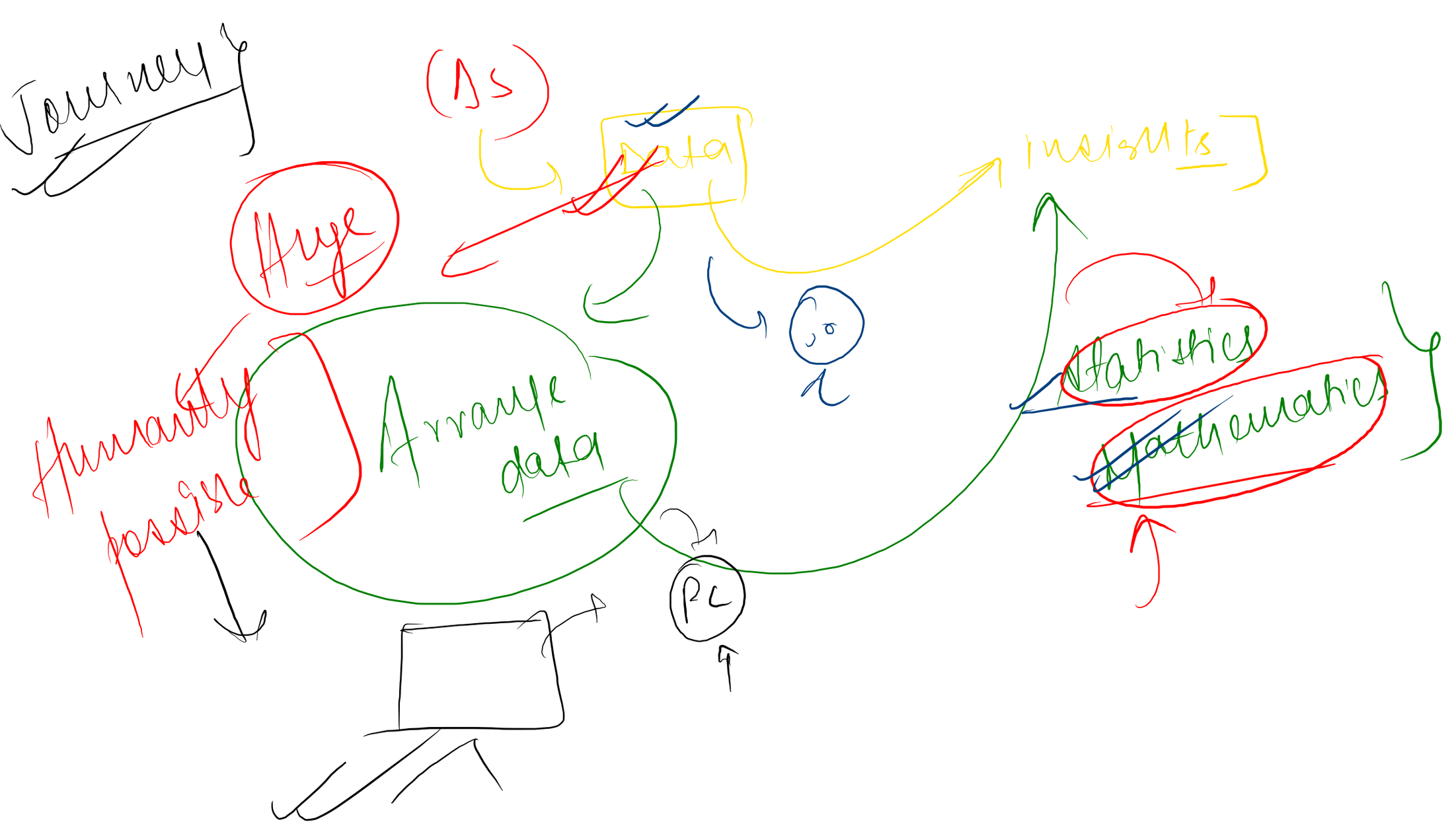

Standout
>7 years
① Basic/foundation
Master
competition
② Learn latest stocle
Early bird advantage
Less crowd
Difficult in early

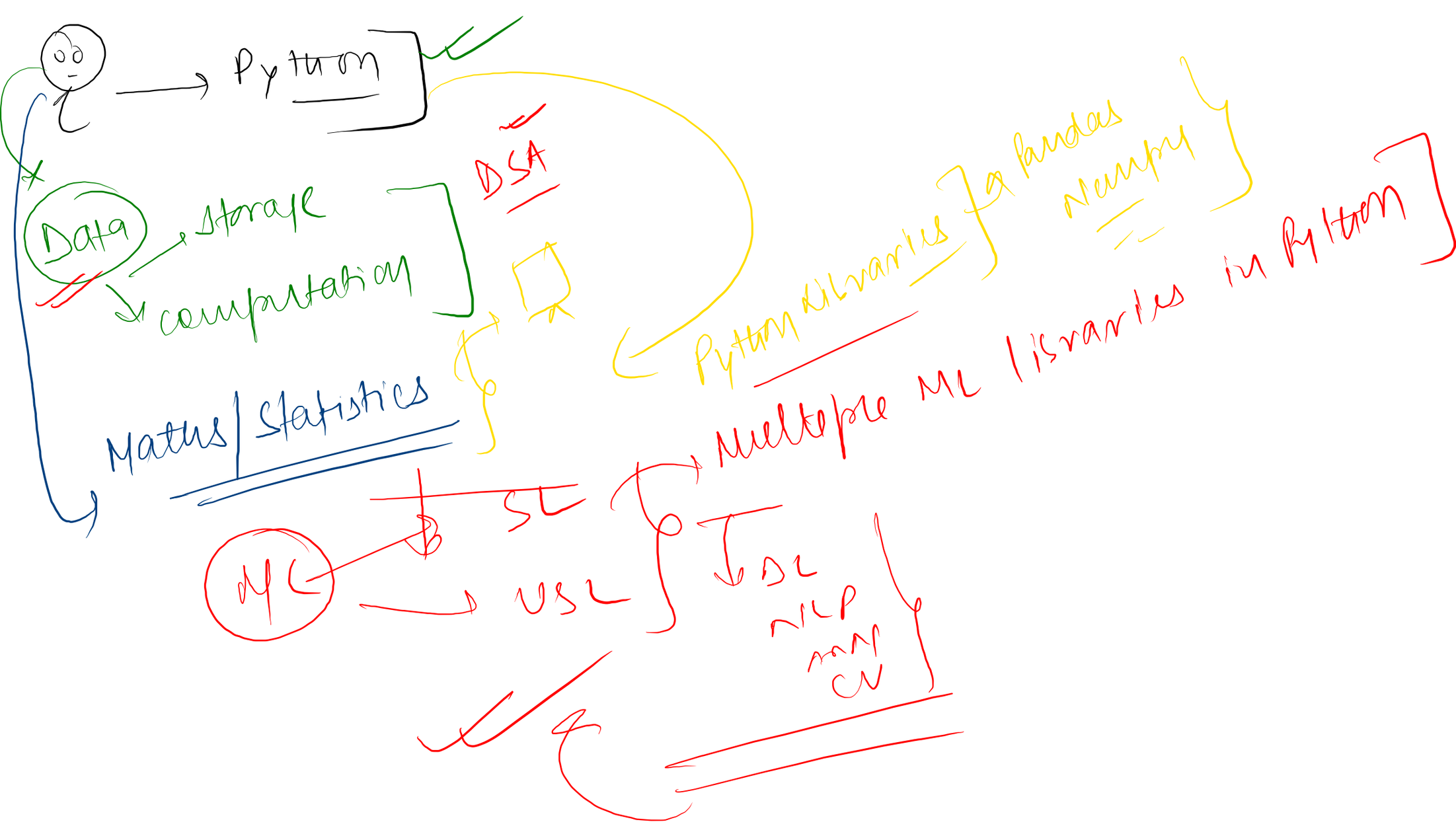


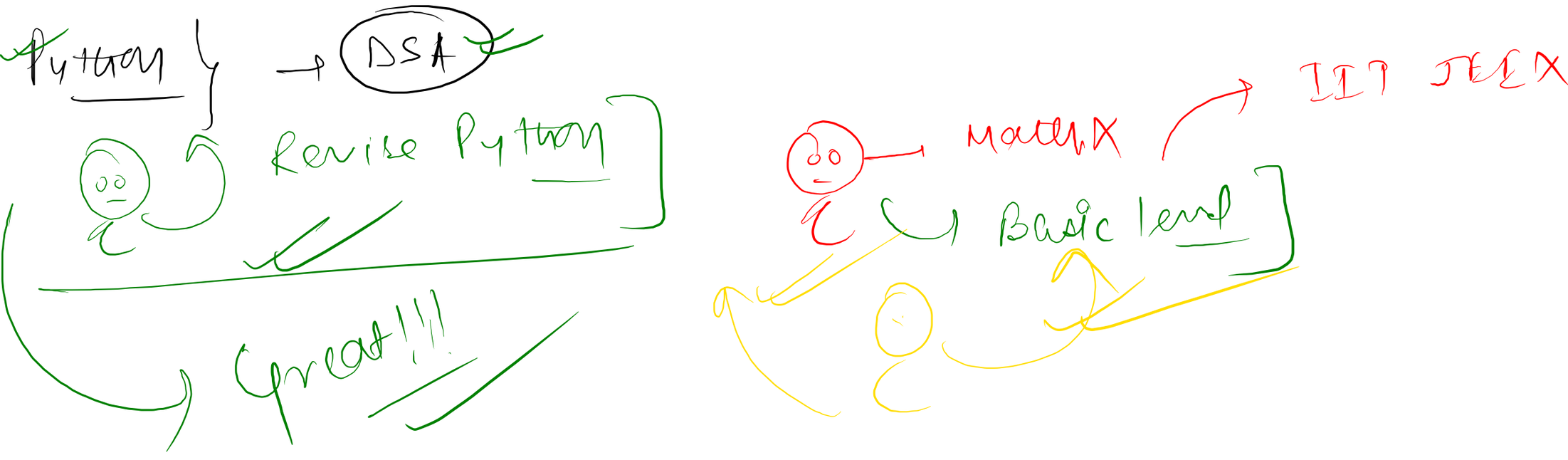
Students
Hackathons
Projects
Competitions
Kafka
changes
code
push
99%

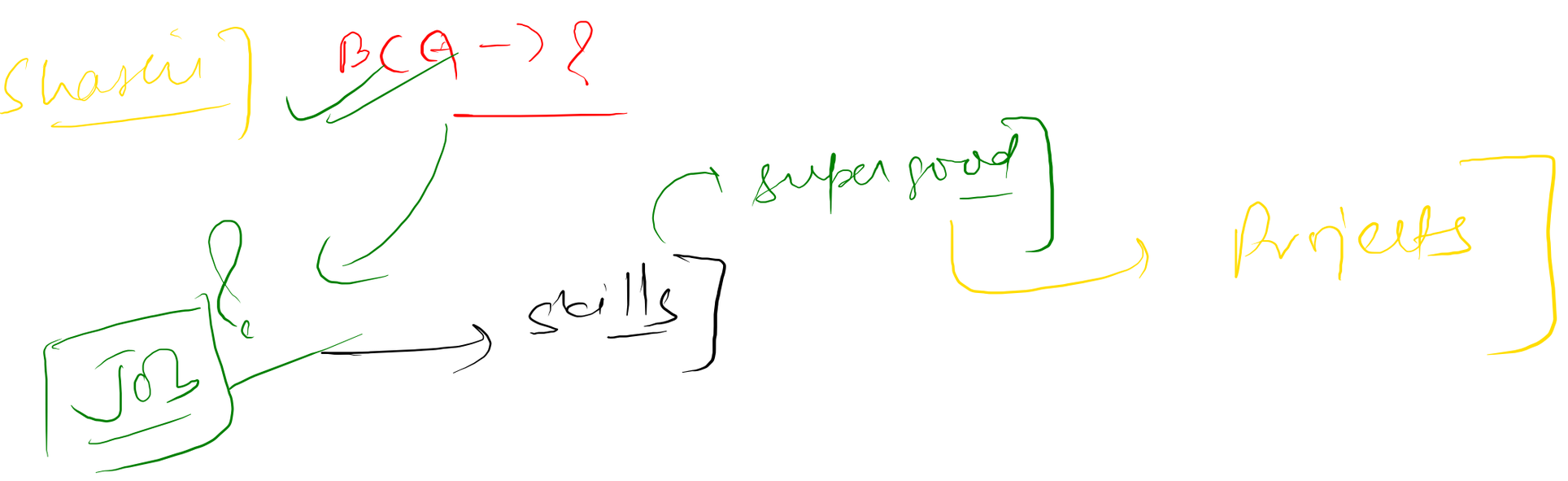

Tourist
serious
views
revenue

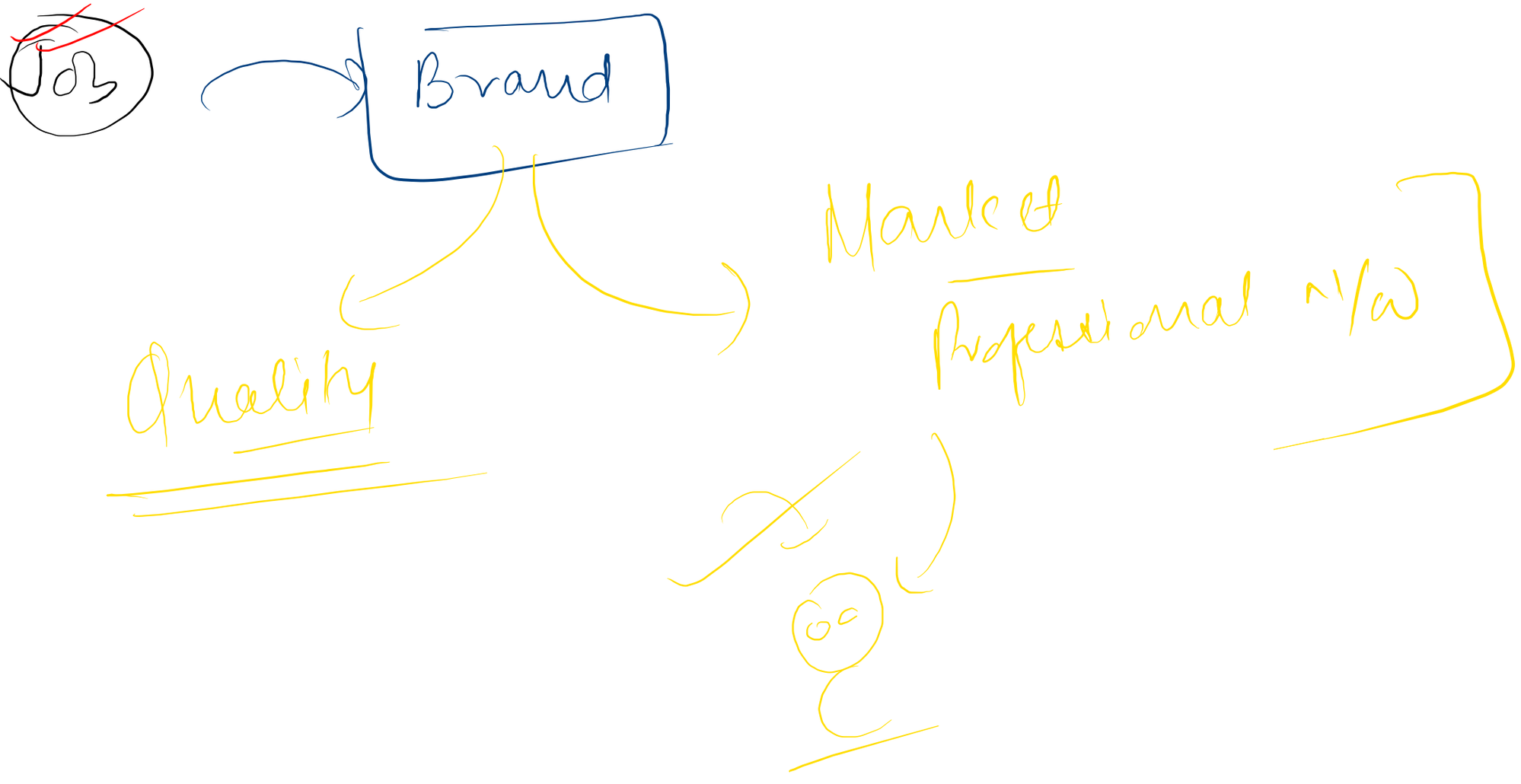

Python / Flask / DSA
Python Developer
4 months
DS
Databases
ML Libraries
Stats
Maths
DL
ANN
NLP
CV
advanced


Journey
(As
Data
Insights
Huge
Arrange data
PC
Statistics
Mathematics


Python
Data
storage
computation
DSA
Maths | Statistics
Python libraries
e.g. pandas
Numpy
Multiple ML libraries in Python
ML
SL
USL
DL
NLP
CV


Python
DSA
Revise Python
Great!!!
MathX
IIT JEE
Basic level


Shastri
BCA -> ?
super good
Projects
skills
Job


Brand
Quality
Market
Professional n/w


Detailed
Curriculum
Topics

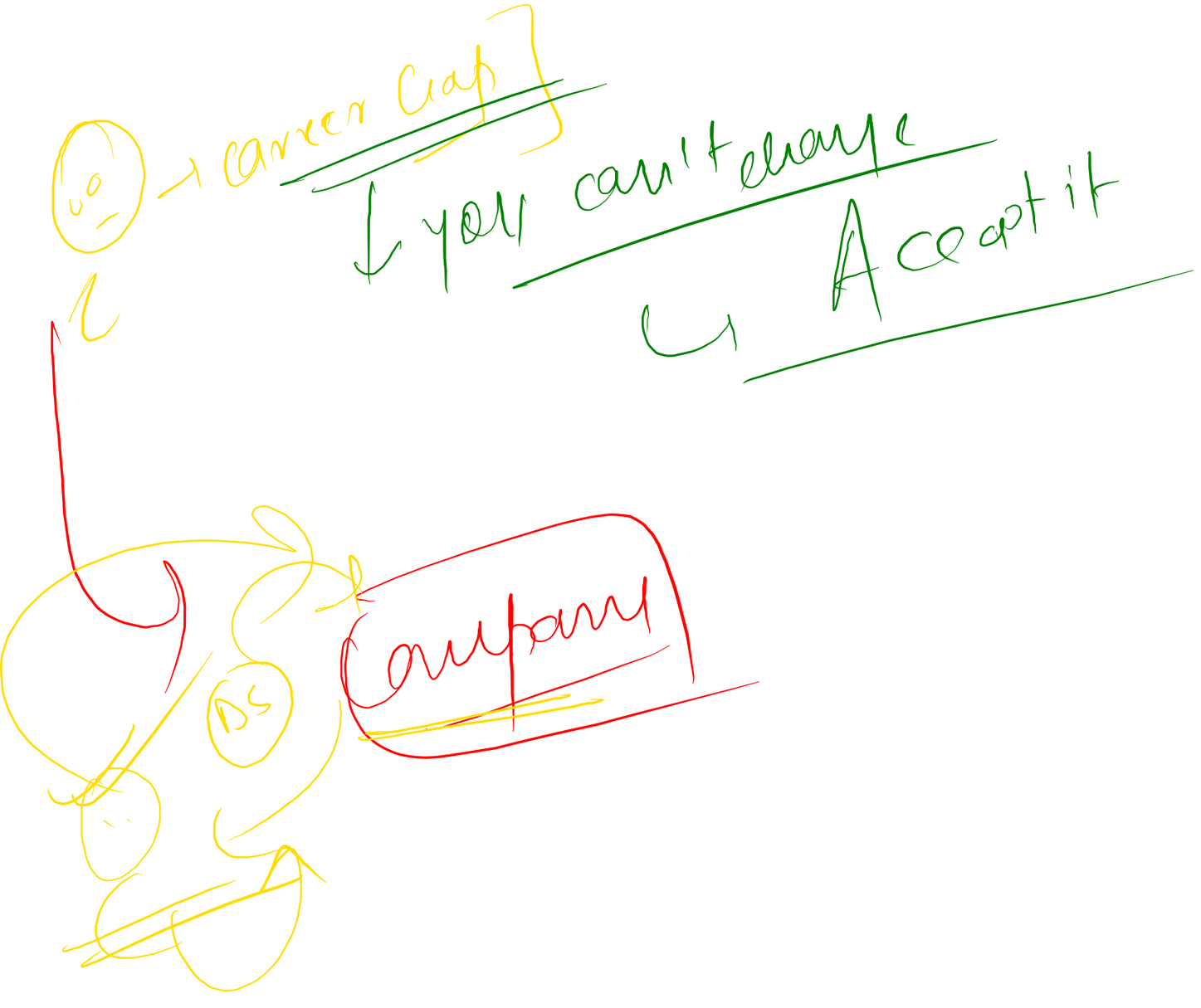
```python
def bubbleSortOptimized(arr):
    for i in range(len(arr)-1, 0 , -1):
        isSorted = True
        for j in range(i):
            if(arr[j]>arr[j+1]):
                isSorted = False
                arr[j],arr[j+1] = arr[j+1],arr[j]


        if isSorted :
            print("Array is already sorted")
            break
```

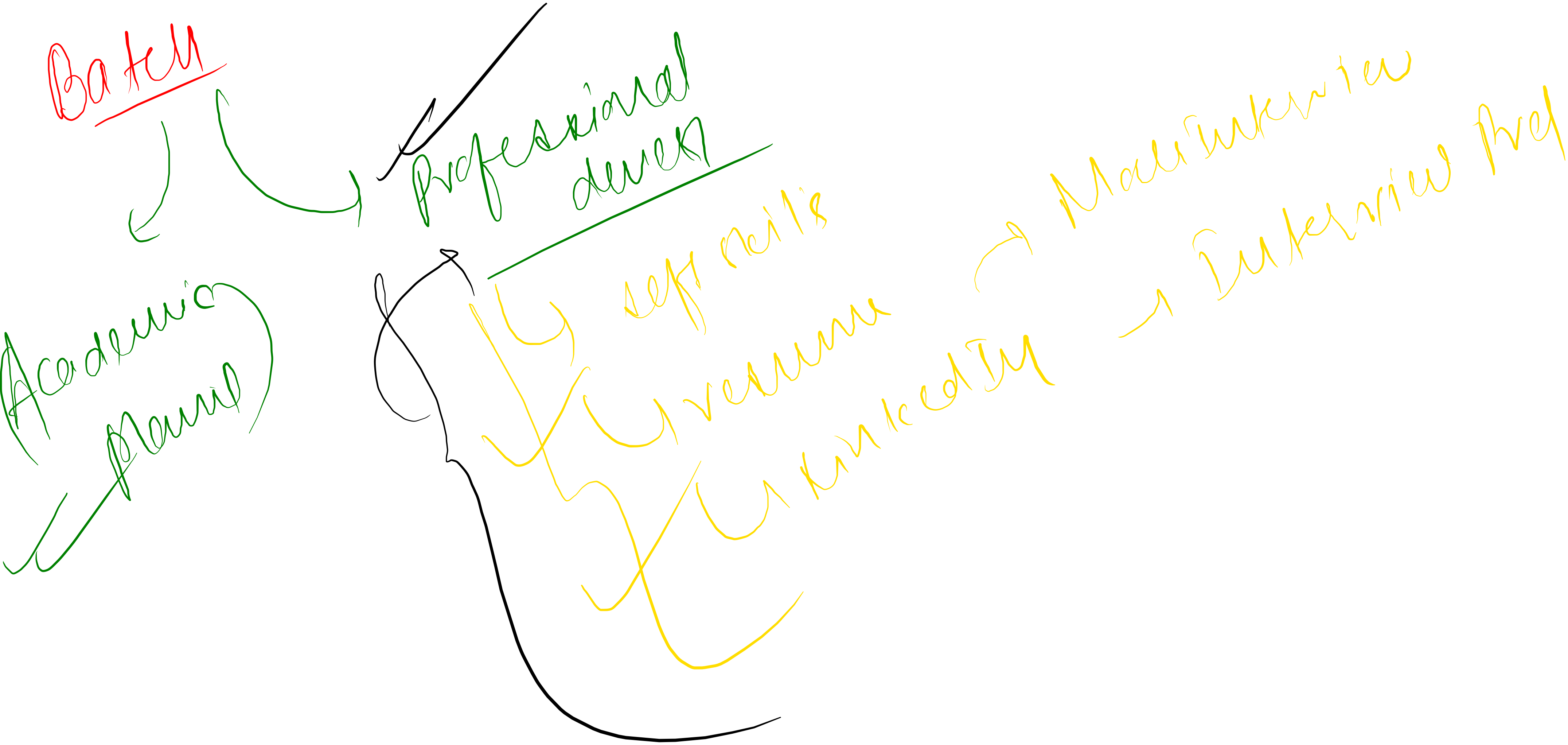
sorted

left will be smaller than right

if all left < right

Isorted


career Gap
If you can't change
Accept it
company


Batch
Academic plans
Professional devel
Manufacturer
Enterprise Dev

Gaurav

competitive

Topics

we start

min

Job interview

Every Topic

Handle understanding

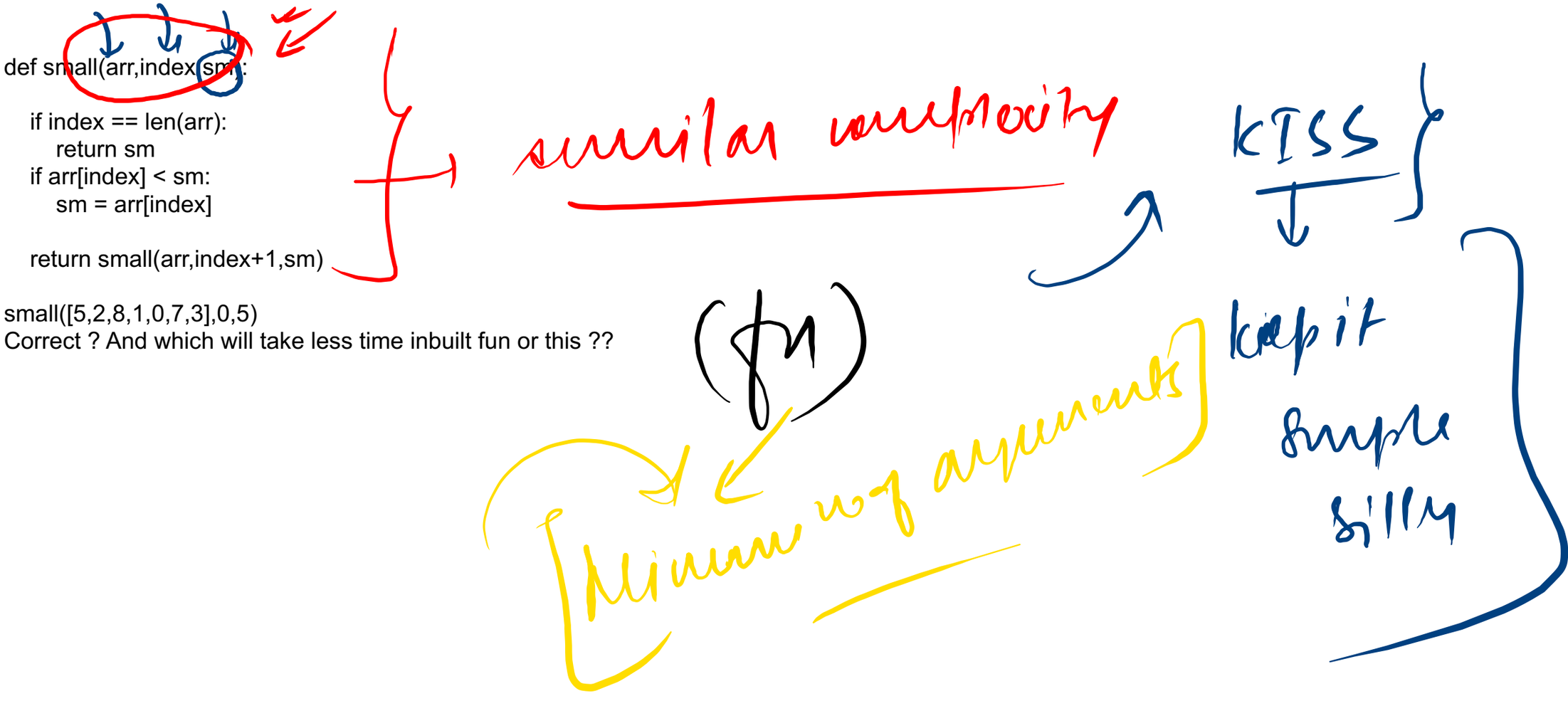
```
def min_fn(num1, num2):
    if (num1 > num2)
        return num2
    else
        return num1
```


Health
Space
Builder

concept → Notebook

```
def small(arr,index,sm):

    if index == len(arr):
        return sm
    if arr[index] < sm:
        sm = arr[index]

    return small(arr,index+1,sm)

small([5,2,8,1,0,7,3],0,5)
```

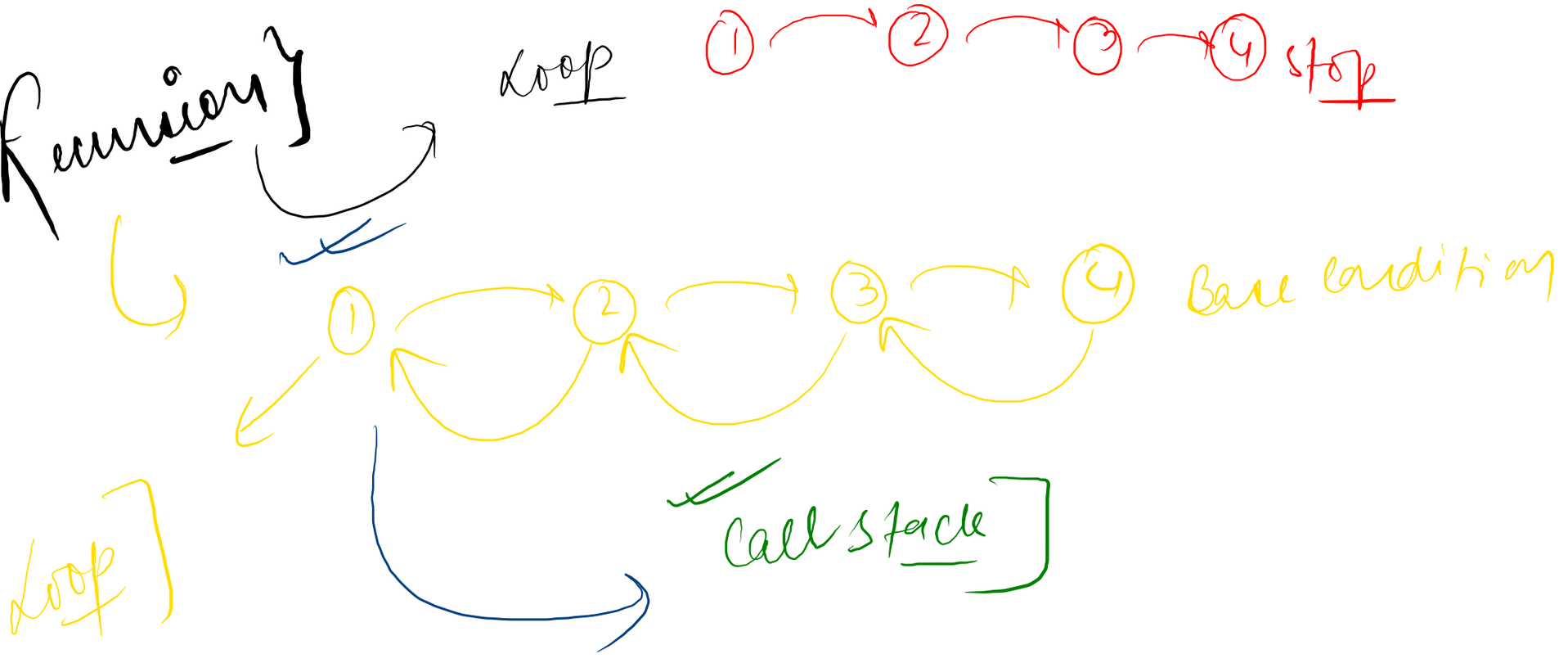
Correct ? And which will take less time inbuilt fun or this ??



Time complexity {

~~Sort~~

Sort main class }

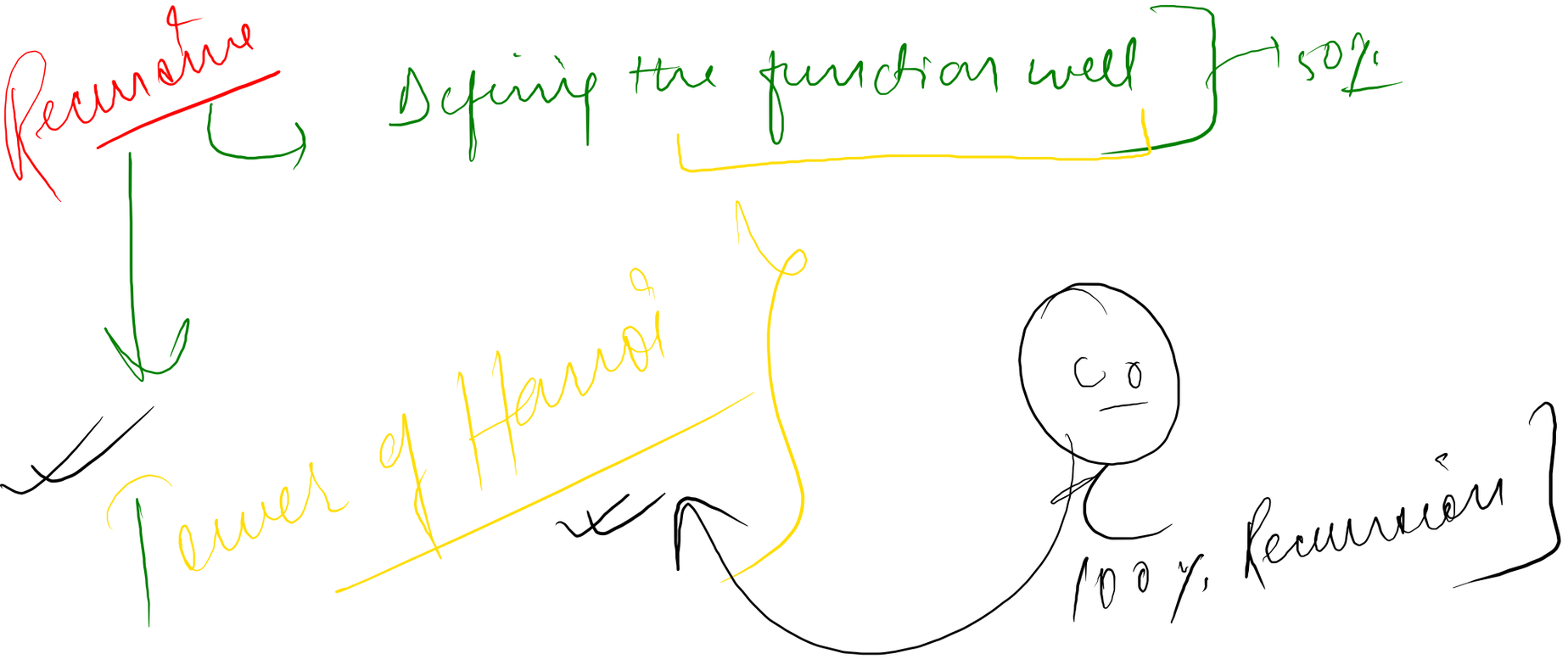

Recursion
Loop
1
2
3
4
stop
1
2
3
4
Base condition
Loop
Call stack

Recursive fn

→ ① Base condition / Termination condition

→ ② Logic

→ ③ Recursive call again


Recursive
Defining the function well
50%
Tower of Hanoi
100% Recursion

Problem solving

Linear search :

for loop

No not present

```
def search(arr, num):
    for i in range(len(arr)):
        if arr[i]==num:
            return i
    return -1
```

Recursion

what is my fn definition

arguments

2?

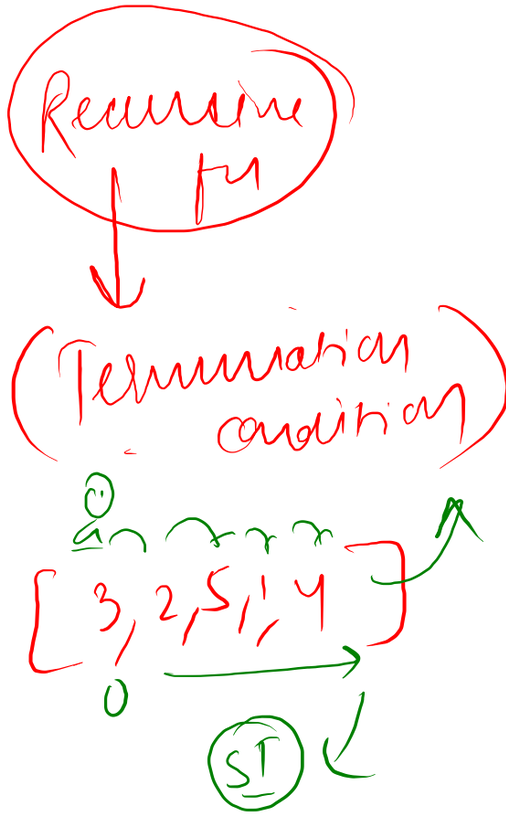
0 1 2 3 4

[3, 1, 5, 2, 4]

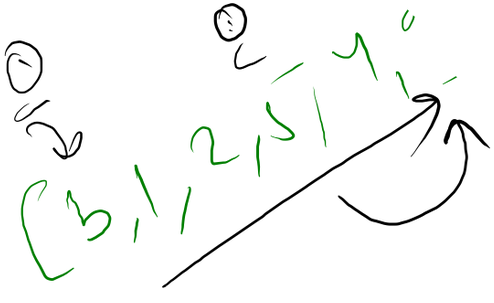
1. arr
2. data
3. sIndex

Function

```
def search (arr, sI, data):
    // Base condition
    if sI >= len (arr)
        return -1

    // Logic
    if arr[sI] == data:
        return sI
    else
        return search(arr, sI+1, data)
```



```python
def search_rec(arr, sI, data):
  #Base condition
  if sI >= len(arr):
    return -1
  if arr[sI] == data :
    return sI
  else :
    return search_rec(arr, sI+1 , data)
```



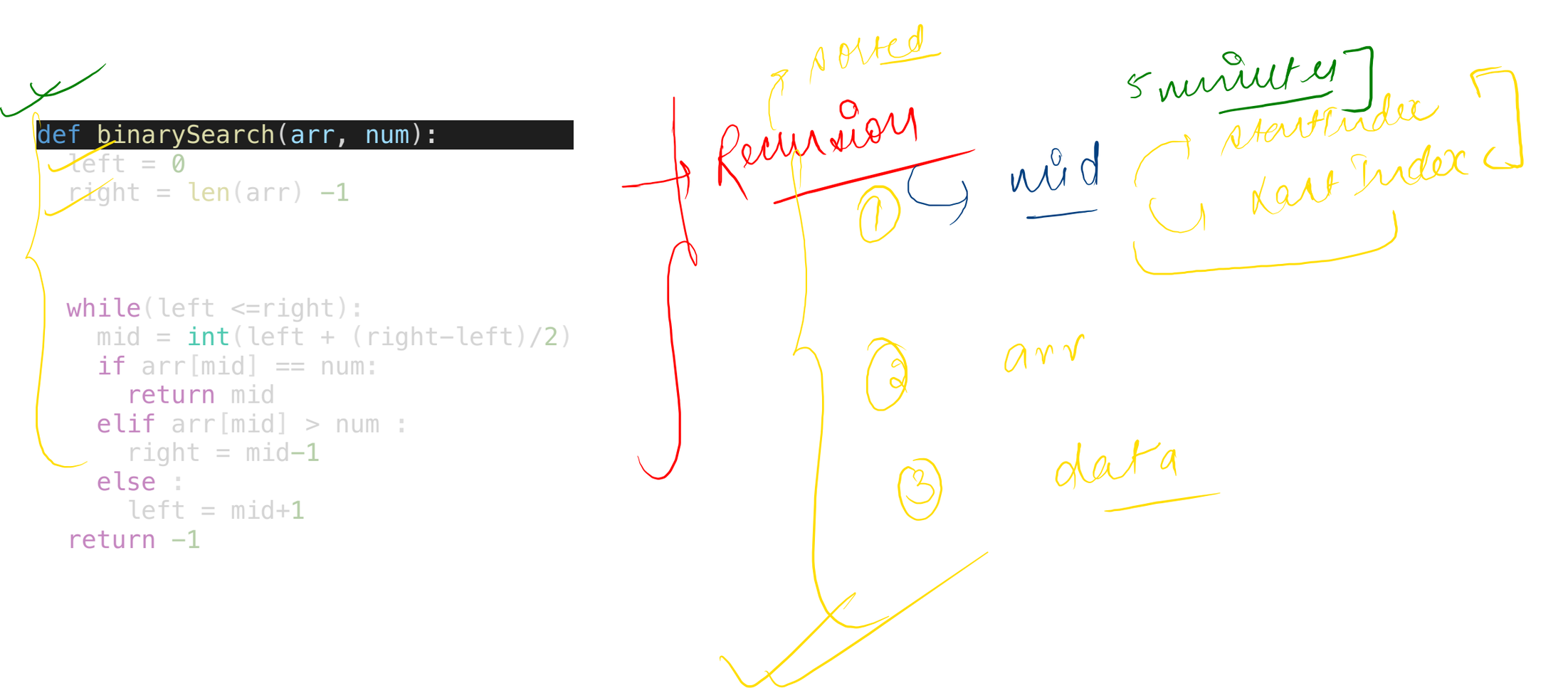
returned search_rec([3,5,2], 0, 2)

① arr = [3,5,2], sI = 0, data = 2

② arr = [3,5,2], sI = 1, data = 2

③ arr = [3,5,2], sI = 2, data = 2

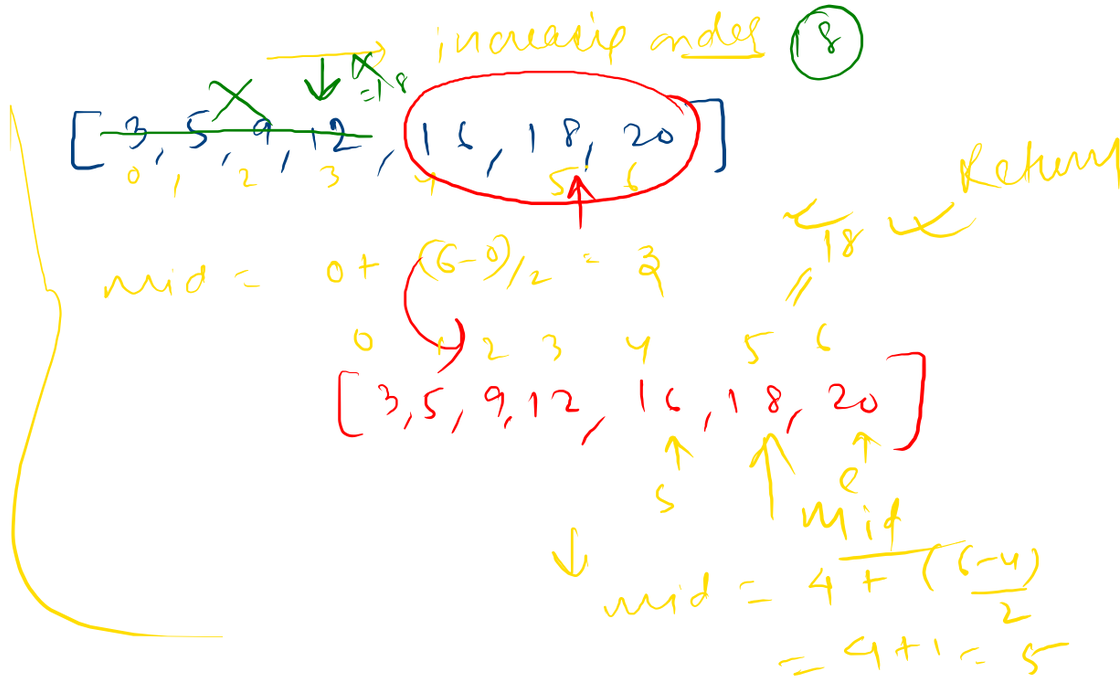
```python
def binarySearch(arr, num):
    left = 0
    right = len(arr) -1


    while(left <=right):
        mid = int(left + (right-left)/2)
        if arr[mid] == num:
            return mid
        elif arr[mid] > num :
            right = mid-1
        else :
            left = mid+1
    return -1
```

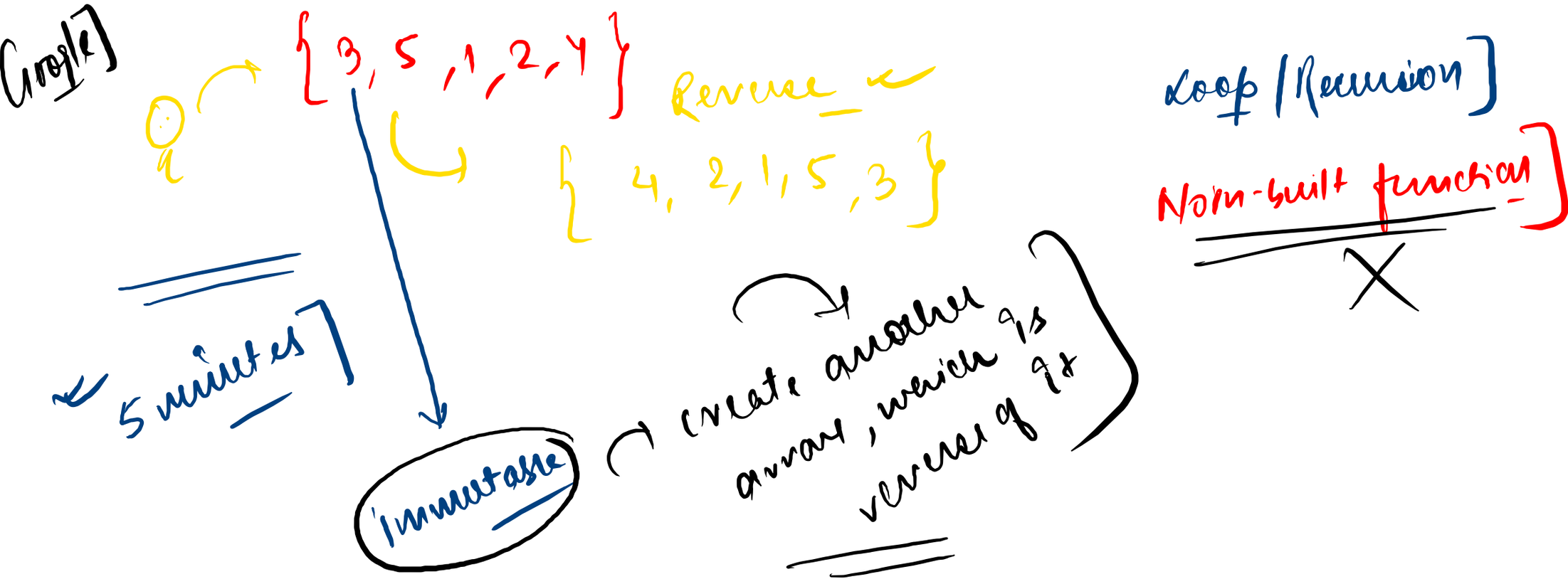


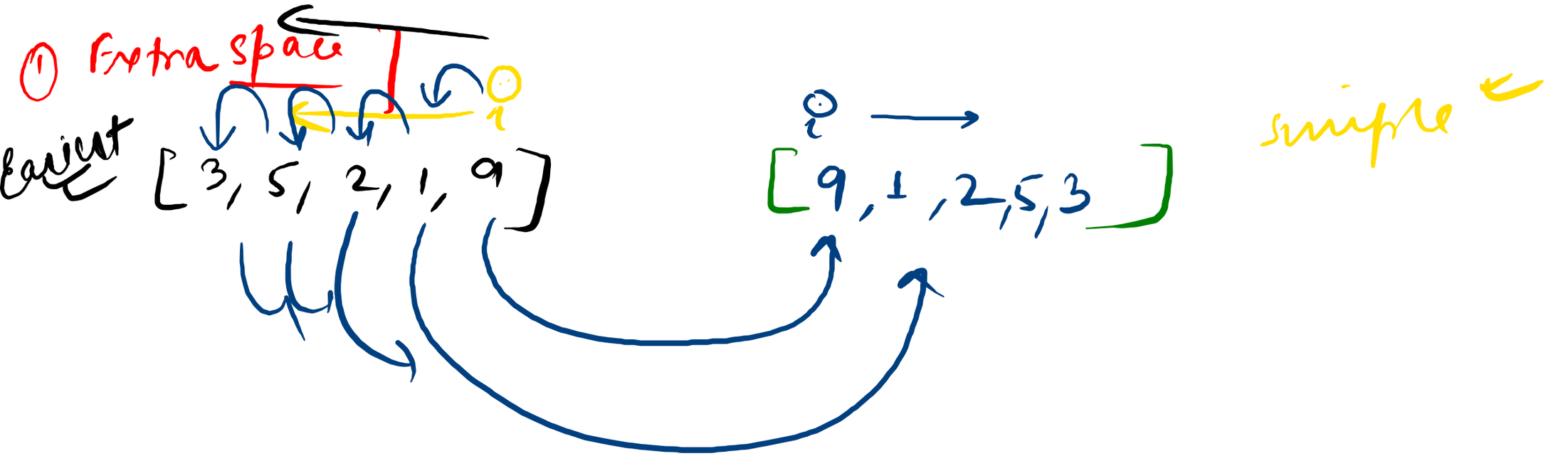
```
def binary_rec(arr, sI, eI, data)
    → Base condition
    if sI > eI :
        return -1

    → mid = int(sI + (eI - sI)/2)
    if(arr[mid] == data):
        return mid
    elif arr[mid] > data
        return binary_rec(arr, sI, mid-1, data)
    else:
        return binary_rec(arr, mid+1, eI, data)
```

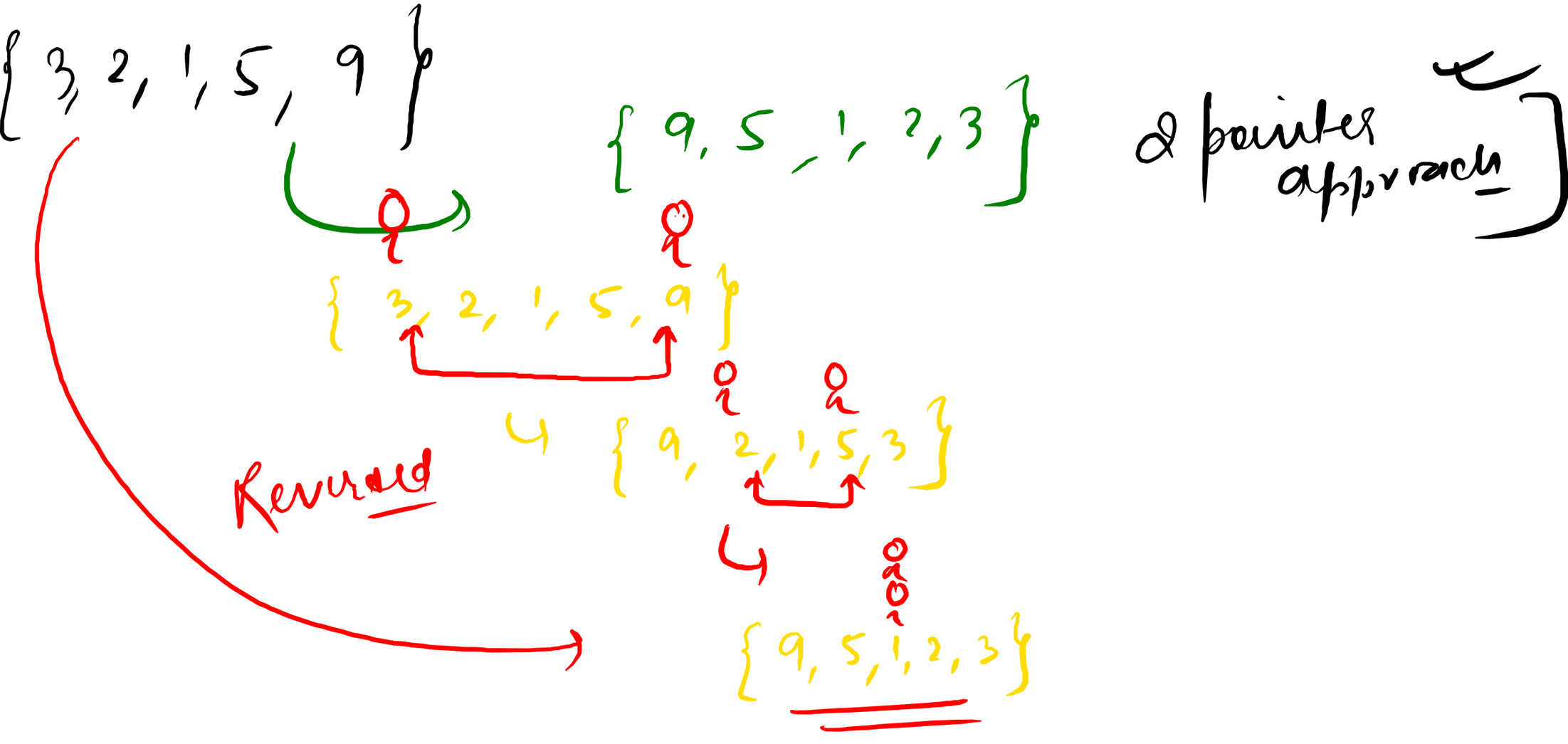

```python
def binarySearch(arr, left, right, num):
  #Base condition
  if left > right :
    return -1

  mid = int( left + (right-left)/2)


  if arr[mid] == num :
    return mid

  elif arr[mid] > num :
    return binarySearch(arr, left, mid-1, num)
  else :
    return binarySearch(arr, mid+1, right, num)
```

O(log n)

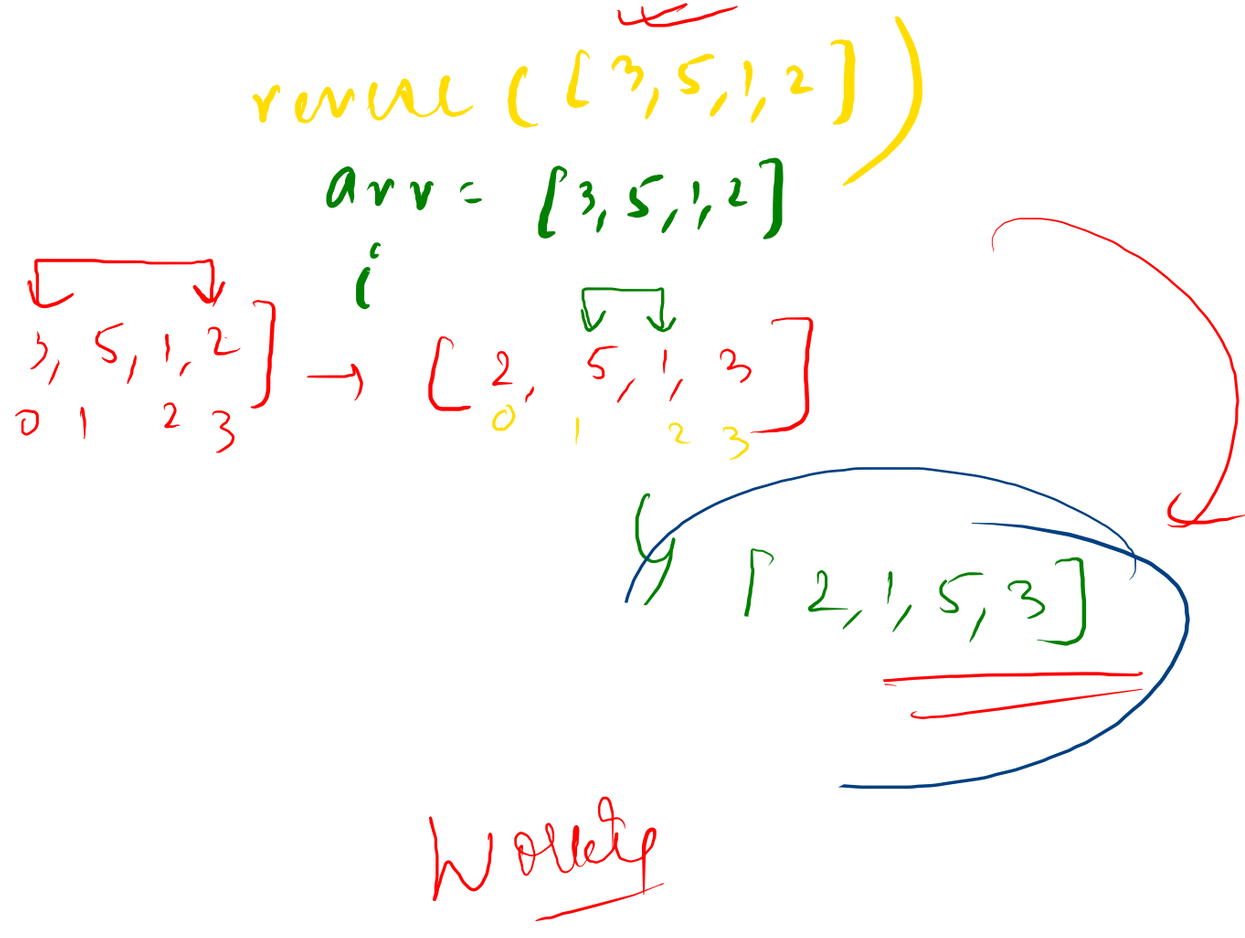


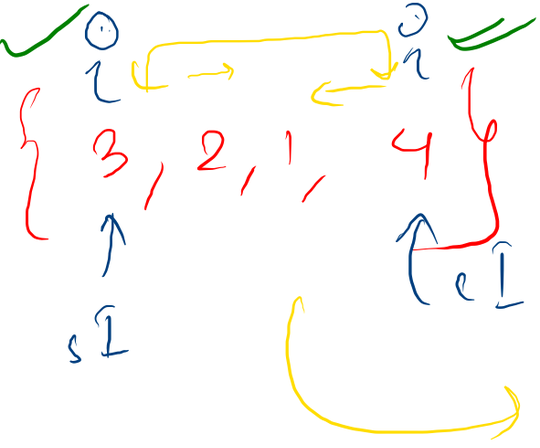

Google
{3, 5, 1, 2, 4}
Reverse
{4, 2, 1, 5, 3}
5 minutes
Interview
create another array, which is reverse of it
Loop / Recursion
Non-built function
X


① Extra space
i
[3, 5, 2, 1, 9]
i
[9, 1, 2, 5, 3]
simple


{ 3, 2, 1, 5, 9 }
{ 9, 5, 1, 2, 3 }
2 pointer approach
i
j
{ 3, 2, 1, 5, 9 }
i
j
{ 9, 2, 1, 5, 3 }
i
j
{ 9, 5, 1, 2, 3 }
Reversed

```python
def reverseArr(arr):
  i =0
  j = len(arr)-1


  while(i<j):
    #swap elements at i and j index
    arr[i],arr[j] = arr[j],arr[i]
    i +=1
    j -=1
```



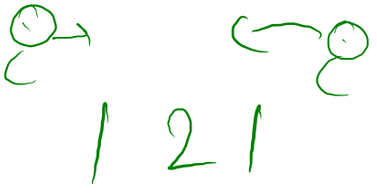


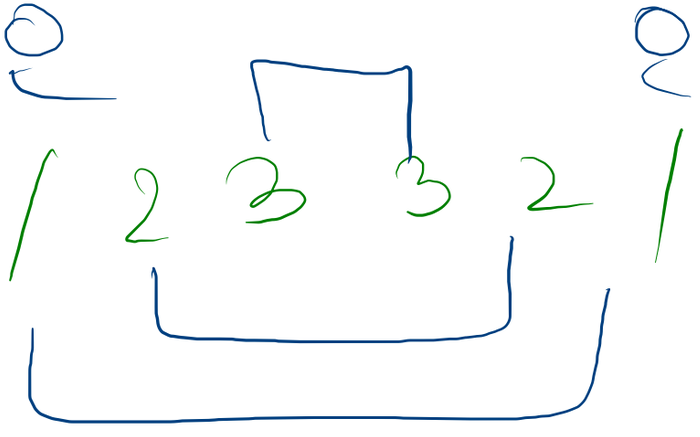
→ Recursion } ① Array
② si → left
③ ei → right

Inplace — No point returning anything

```python
def reverse_rec(arr, sI, eI):
  if sI >= eI :
    return
  arr[sI],arr[eI] = arr[eI],arr[sI] #swapping values
  reverse_rec(arr,sI+1,eI-1)


arr = [1,2,3,4,5]
reverse_rec(arr, 0, len(arr)-1)
print(arr)
```

Palindrome





5 x palindrom

arr = [1, 2, 3, 3, 2, 1]

→ arr is palindrome or Not

Iteratively first

50 minutes

Name of insult function

(1 C) → Recursion
→ Loop

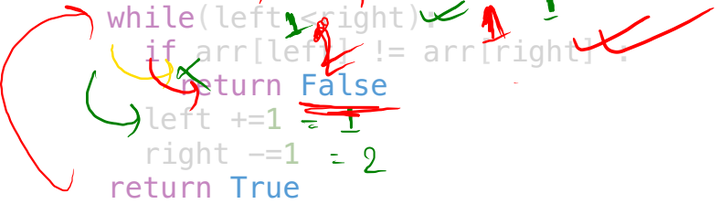
→ No of operations of input size

```
def printNum(arr, sI):
    if sI >= len(arr)
        return
    print(arr[sI])
    printNum(arr, sI+1)
```

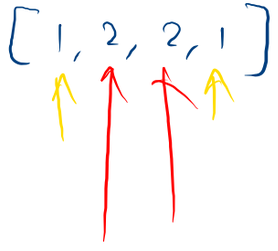
size of array

No of operations of input size
TC = O(n)

```python
def isPalindrome(arr):
  # 2 pointer approach
  left =0
  right = len(arr)-1

  while(left<=right):
    if arr[left] != arr[right]:
      return False
    left +=1
    right -=1
  return True
```



isPalindrome([1, 2, 2, 1])



True

isPalindrome([1, 2, 1, 1])

False

Not a palindrome

Arrays interview

Rotate it any direction

→ L→R

[3, 1, 2, 5, 4]

Rotate once L→R

[4, 3, 1, 2, 5]

Rotate again

[5, 4, 3, 1, 2]

[2, 5, 4, 3, 1]

Rotation is taking place

→ arr, Rotate it k times } L→R

Tell me How The final array look like -

[1,2,3,4,5] k = 2

swapping

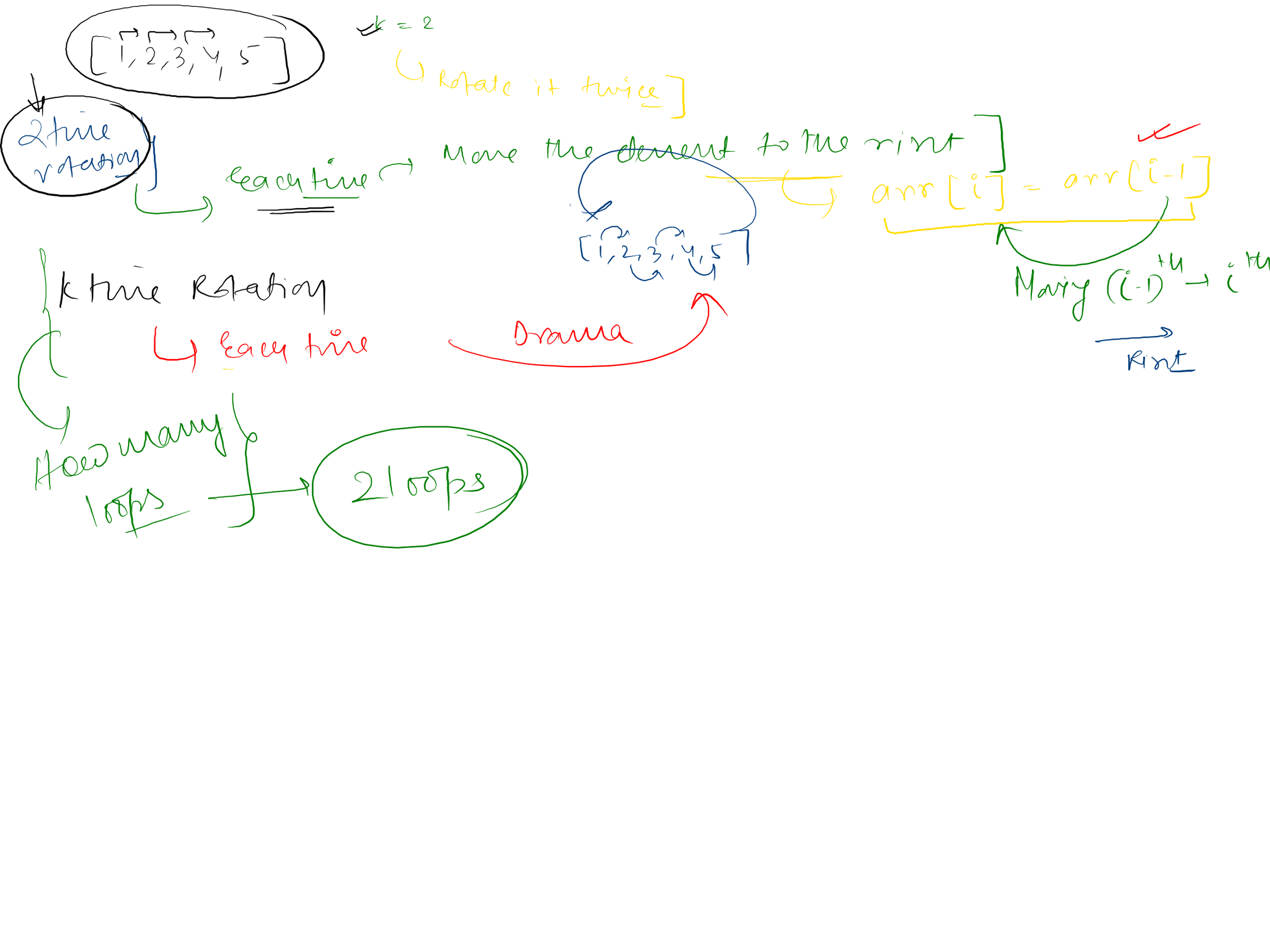
[5,1,2,3,4] → [4,5,1,2,3]

7 minutes

11:17 → 11:24 AM

[1,2,3,4,5] k = 3

[5,1,2,3,4] → [4,5,1,2,3] → [3,4,5,1,2]


[1, 2, 3, 4, 5]
k = 2
↳ Rotate it twice]
2 time rotation
each time
Move the element to the right
arr[i] = arr[i-1]
[1, 2, 3, 4, 5]
Drama
Many (i-1)th → ith
Print
k time Rotation
↳ Each time
How many loops
loops
21 loops

Loop1 → Take care of rotations] k

↳ Loop2 → Take care of Moving item to the right]

a Rotation

arr = [1, 2, 3]

k = 2

k = 5

k = 141

141 % 3

k=2 } [1, 2, 3] → [3, 1, 2] → [2, 3, 1]

k=5 } [1, 2, 3] → [3, 1, 2] → [2, 3, 1] → [1, 2, 3] → [3, 2, 1] → [2, 3, 1]

after 3 times

k times you need to rotate

k = k % size of array

Actual time we need to rotate

same as original

[1, 2, 3]

1 → [3, 1, 2]

2 → [2, 3, 1]

3 → [1, 2, 3]

4 → [3, 1, 2]

3 time

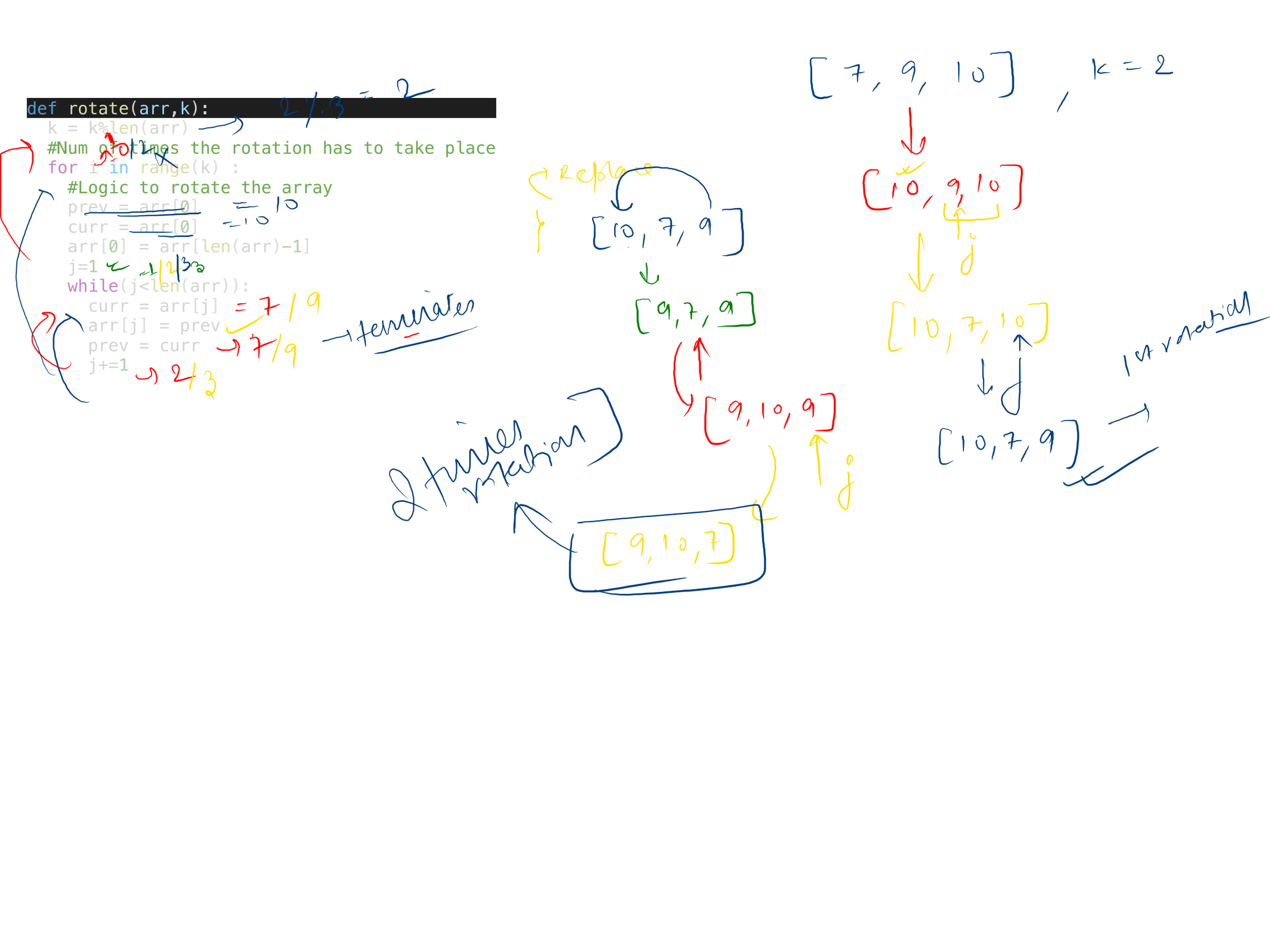
1 / 4 / 7 / 10 → [3, 1, 2]

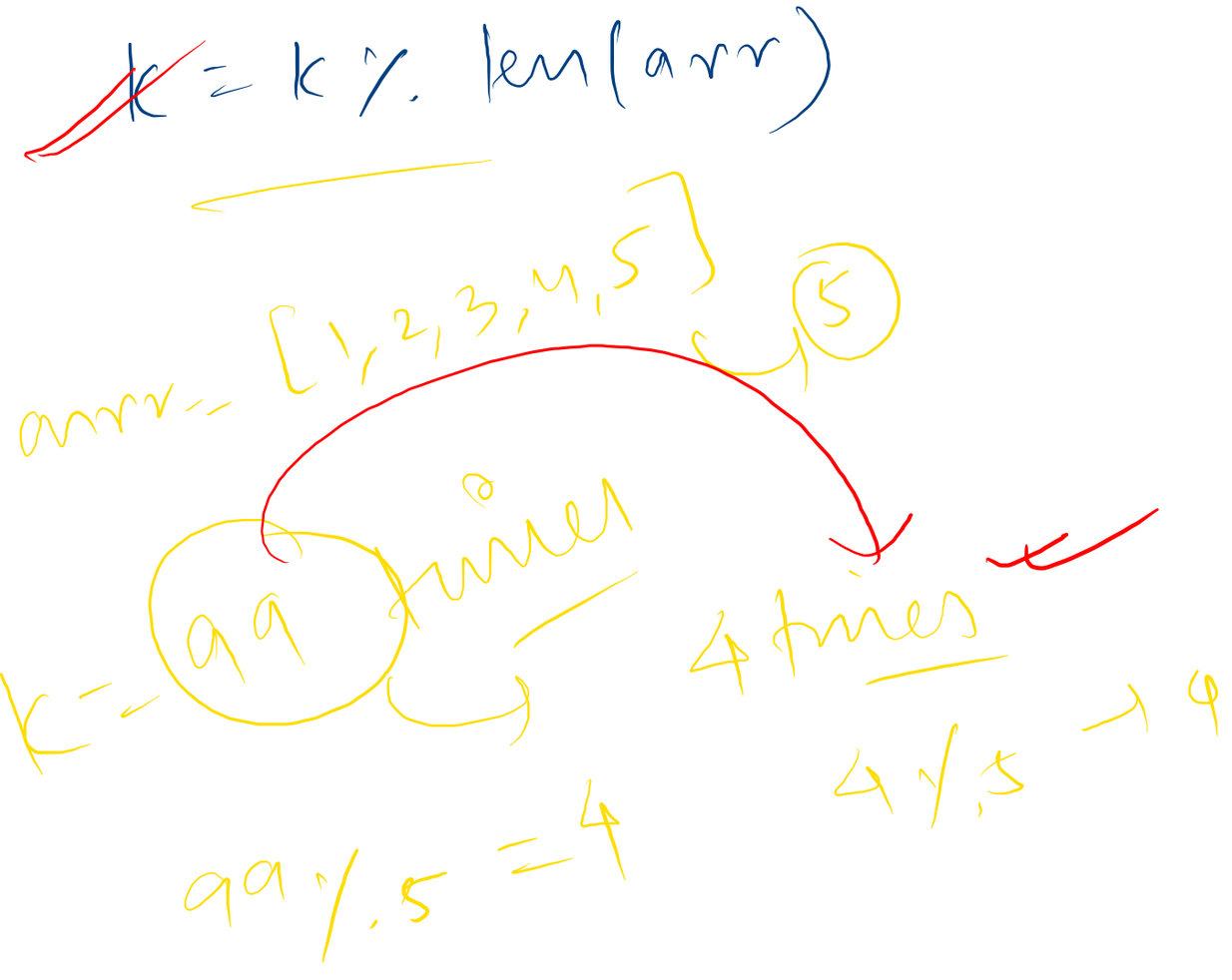
1 / 3 / 3 / 3

Result :-

1 % size of arr

```
def rotate(arr,k):
    k = k%len(arr)
    #Num of times the rotation has to take place
    for i in range(k) :
        #Logic to rotate the array
        prev = arr[0]
        curr = arr[0]
        arr[0] = arr[len(arr)-1]
        j=1
        while(j<len(arr)):
            curr = arr[j]
            arr[j] = prev
            prev = curr
            j+=1
```




k = k % len(arr)
arr = [1,2,3,4,5]
5
99
times
4 times
k =
99 % 5 = 4
4 % 5 → 9

* Theory → Moderate
* Practical

[1, 2, 3, 4, 5] → k = 3 times

↓

Reverse the entire array

[5, 4, 3, 2, 1]

↑ Reverse 1st k elements

↓

[3, 4, 5, 2, 1]

Reverse last (n-k) elements

↳ [3, 4, 5, 1, 2]

[1, 2, 3, 4, 5]

k times = 2

↳ Reverse entire array

[5, 4, 3, 2, 1]

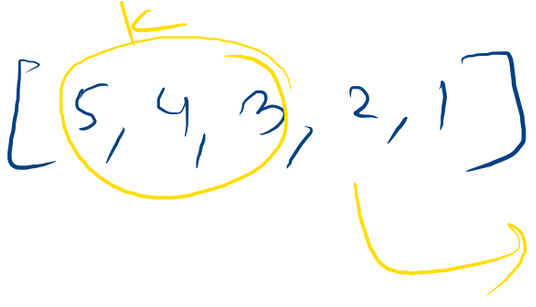
↳ [4, 5, 3, 2, 1]

↳ [4, 5, 1, 2, 3]

//

[1,2,3,4,5] Rotate it 3 times

↳ Reverse the array



↳ [3,4,5,1,2]

[1, 2, 3, 4, 5]

k = 50 times

(k > size)

k = k % . size = 50 % . 5 = 0

Moved to the end

[1, 2, 3, 4, 5]

k = 40

40 % 5 = 0

5

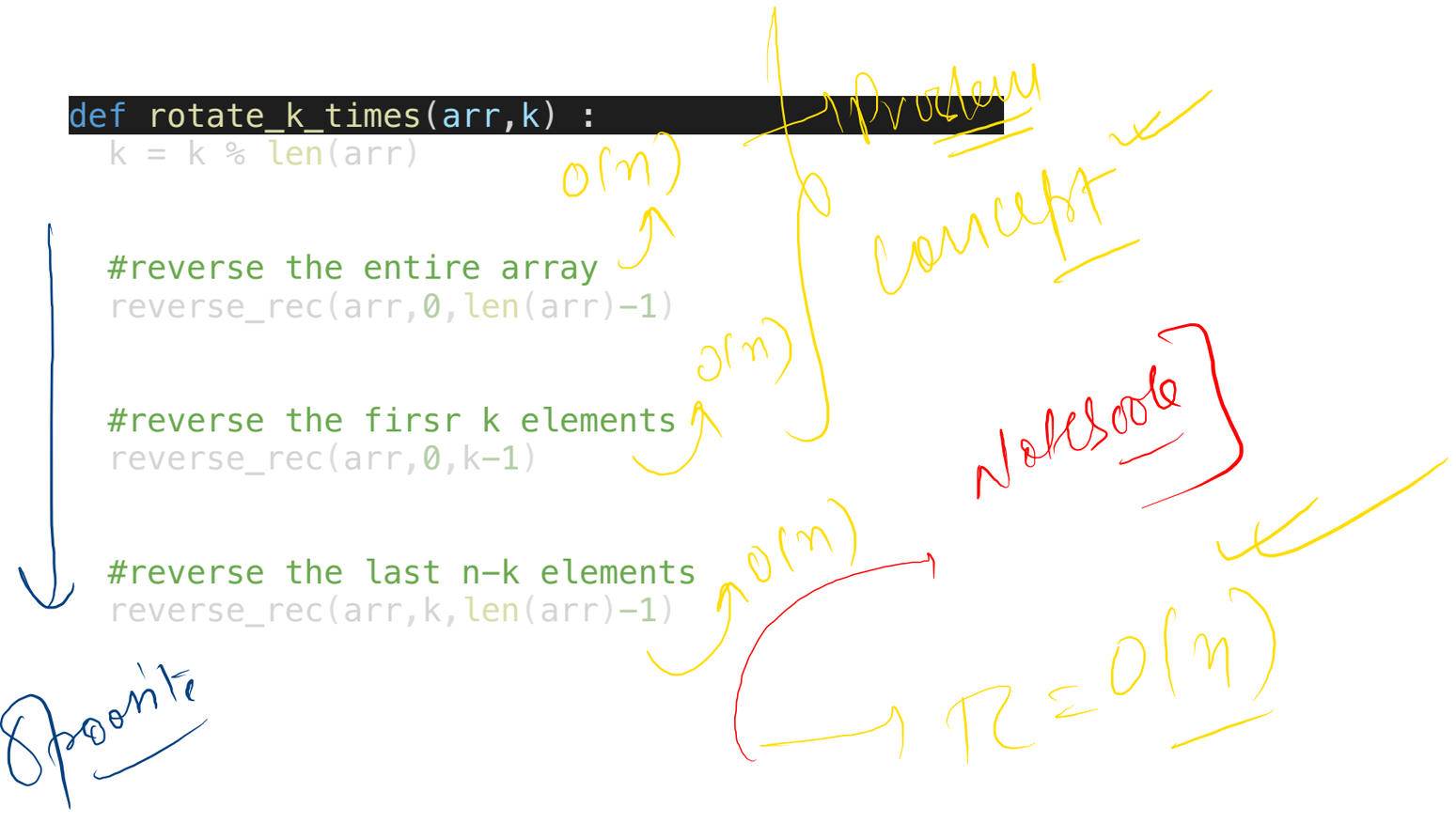
[1, 2, 3, 4, 5]

41

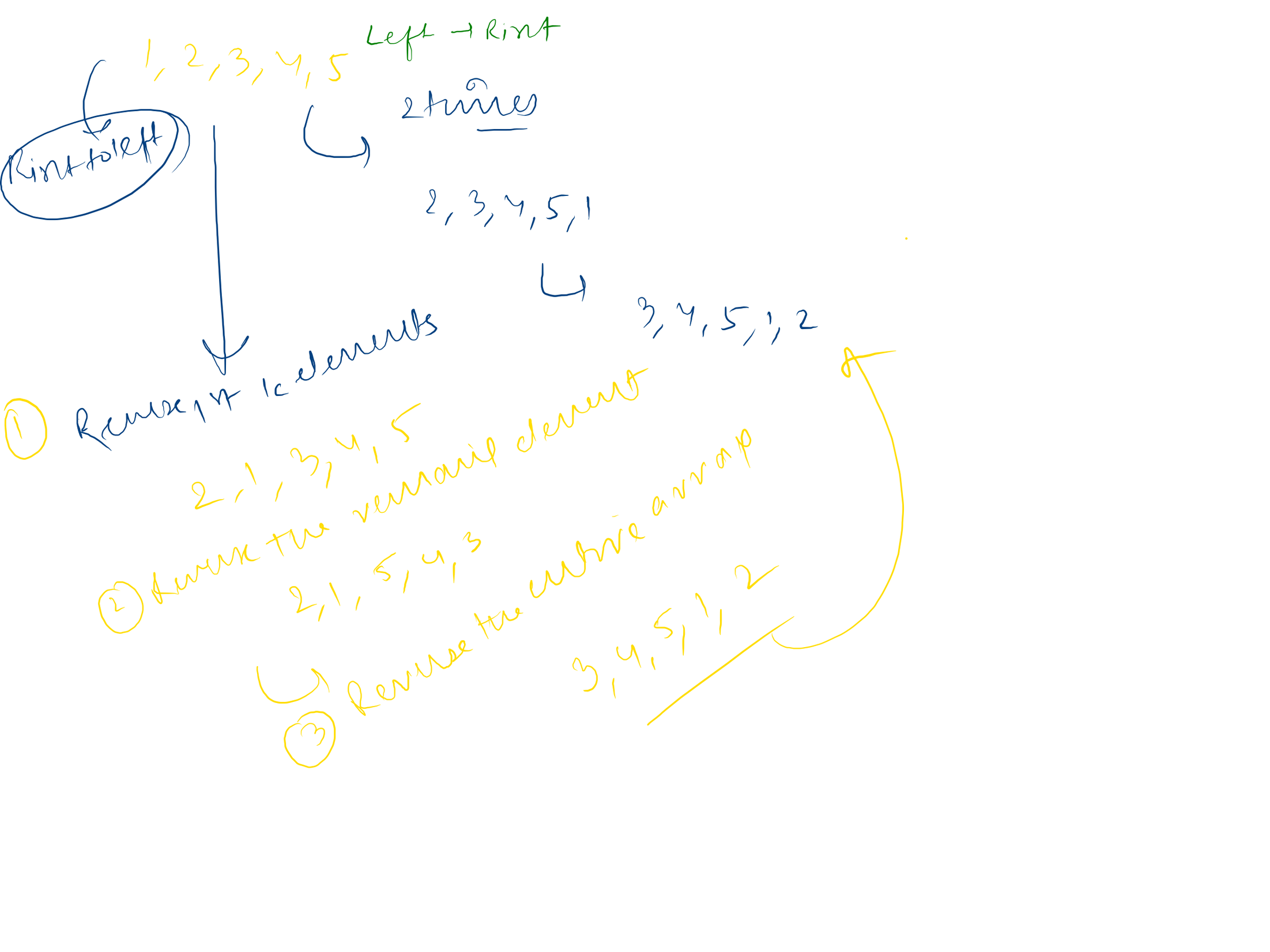
41 % 5 = 1 → Straight for this

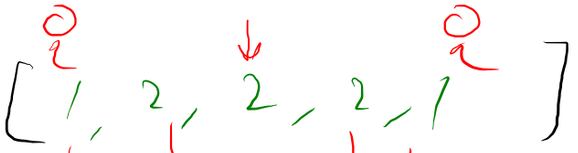
x % y → Remainder when x is divided by y

```
def rotate_k_times(arr,k) :
    k = k % len(arr)

    #reverse the entire array
    reverse_rec(arr,0,len(arr)-1)

    #reverse the firsr k elements
    reverse_rec(arr,0,k-1)

    #reverse the last n-k elements
    reverse_rec(arr,k,len(arr)-1)
```



Pointer


Left → Right
1, 2, 3, 4, 5
2 times
Right to left
2, 3, 4, 5, 1
3, 4, 5, 1, 2
① Reverse 1st k elements
2, 1, 3, 4, 5
② Reverse the remaining elements
2, 1, 5, 4, 3
③ Reverse the whole array



Palindrome

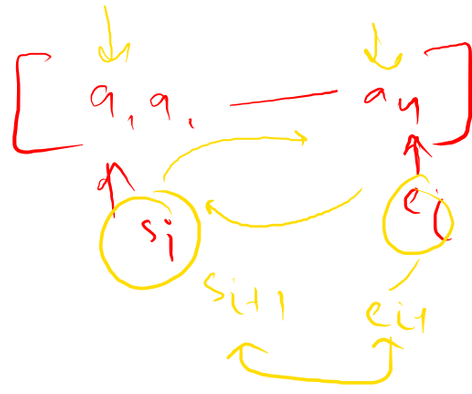
Recursion

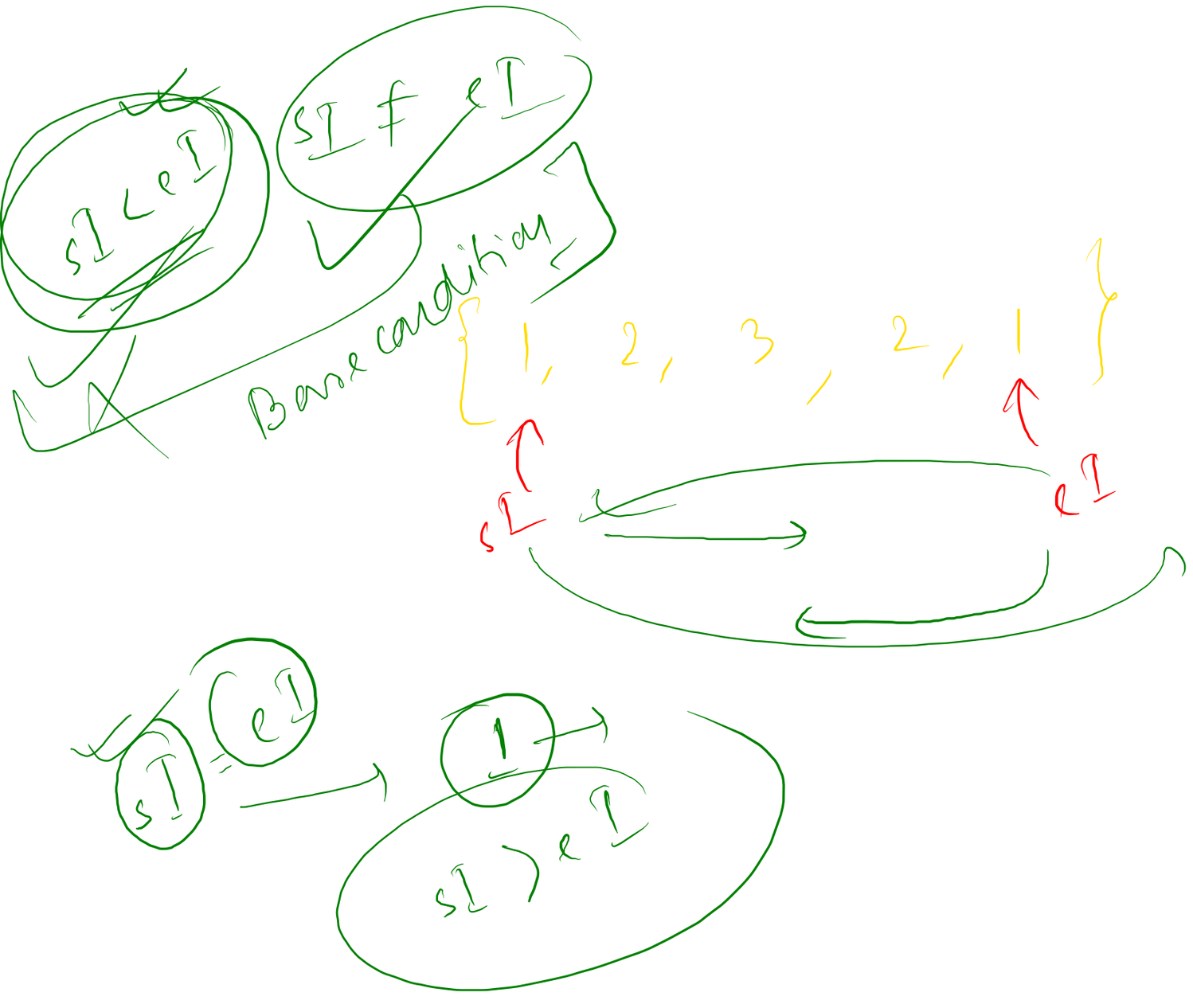
function

① arr
② start
③ end

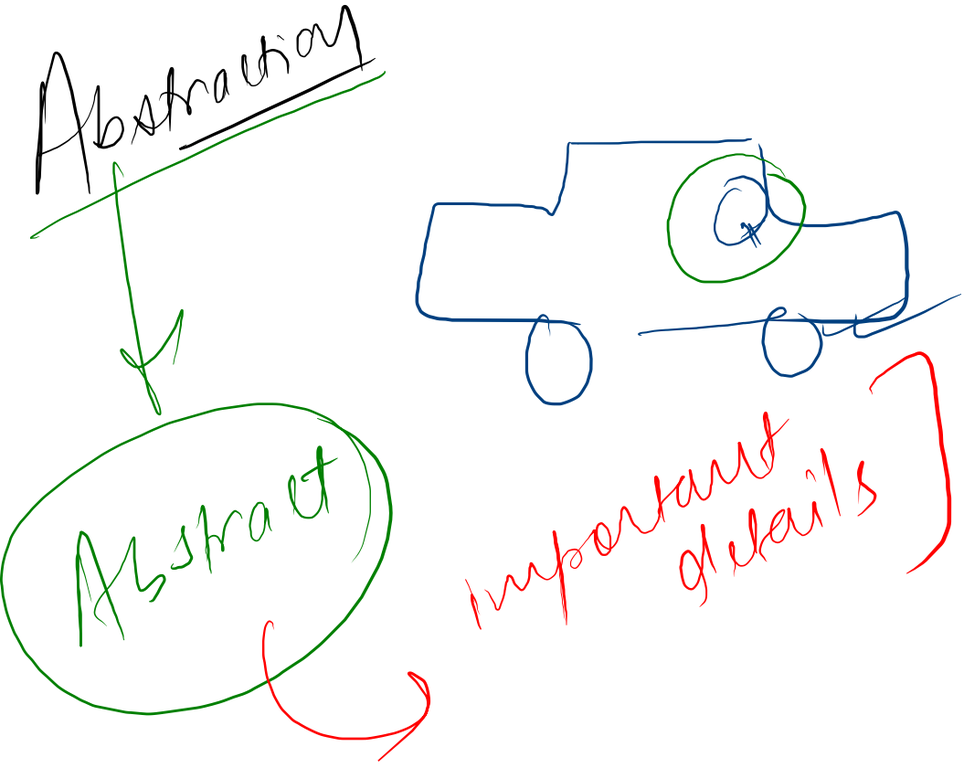
SO ✗ ✓

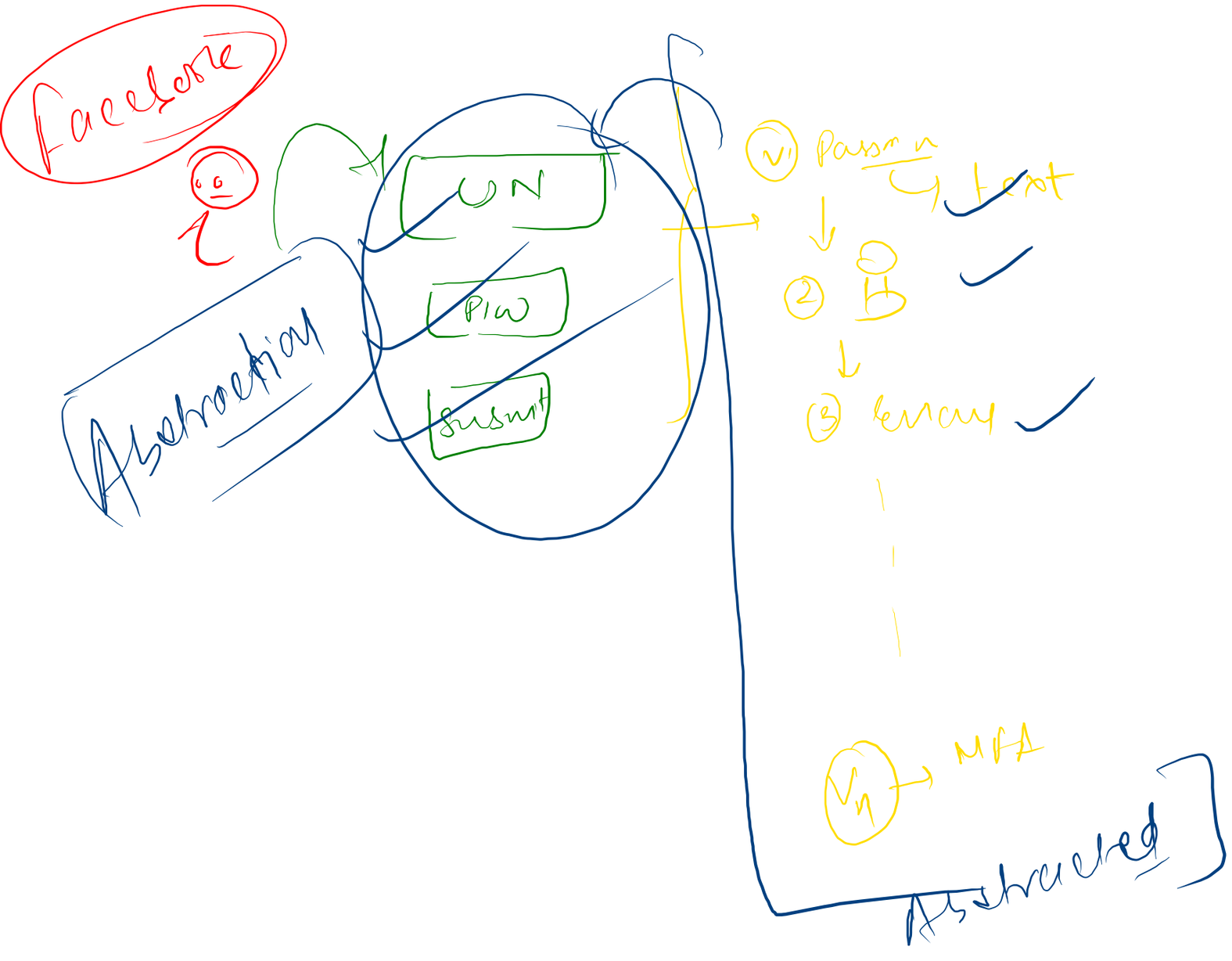
```
def isPalindrome(arr, si, ei):
    #Base condition
    if (si > ei):
        return false

    if (si == ei):
        return true

    if (arr[si] != arr[ei]):
        return false

    return isPalindrome(arr, si+1, ei-1)
```

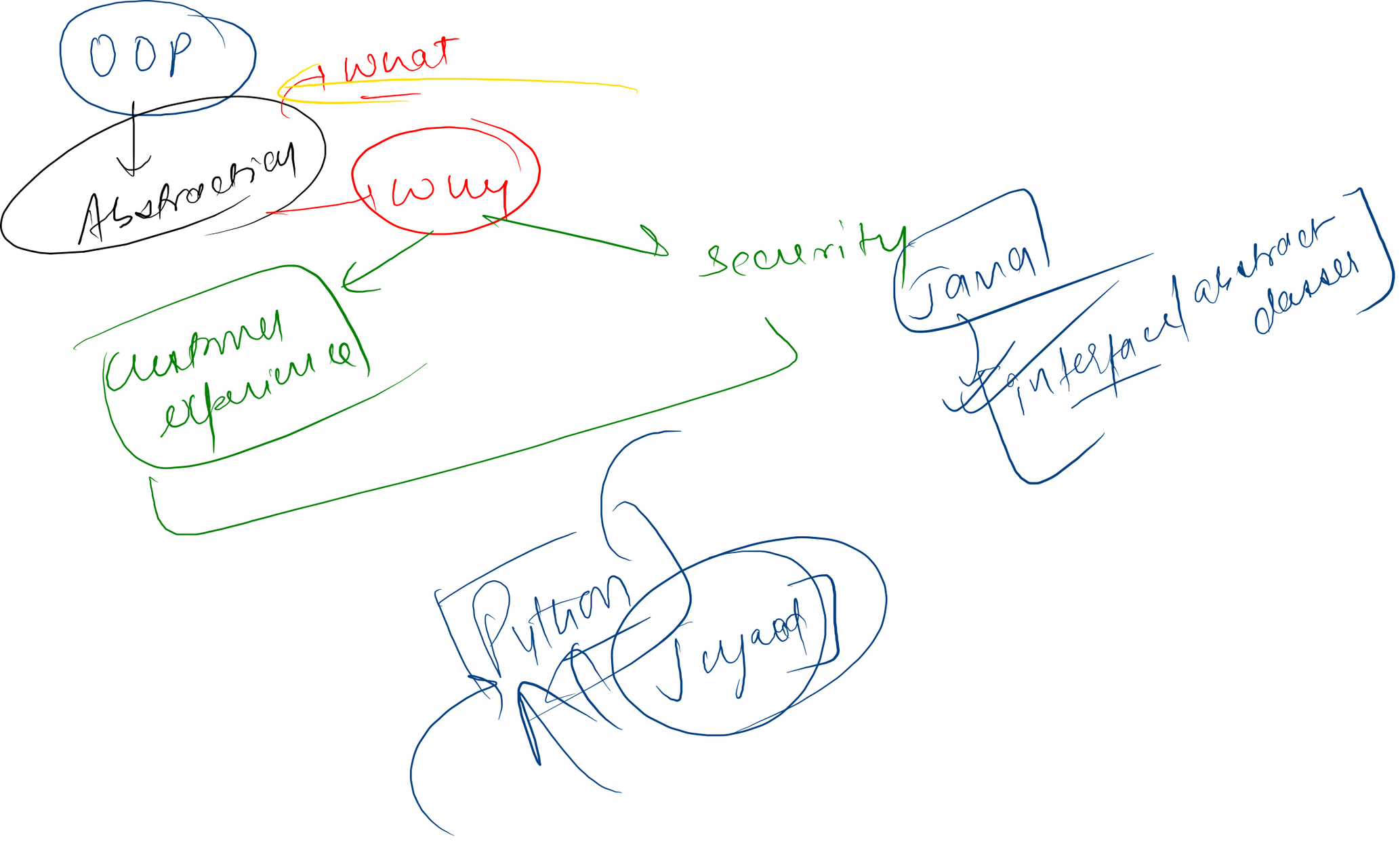



s1 < e1
s1 = e1
Base condition
1, 2, 3, 2, 1
s1
e1
s1 = e1
1
s1 > e1


Vaishnavi
Problem
Regular!
Every day
1-2 hours

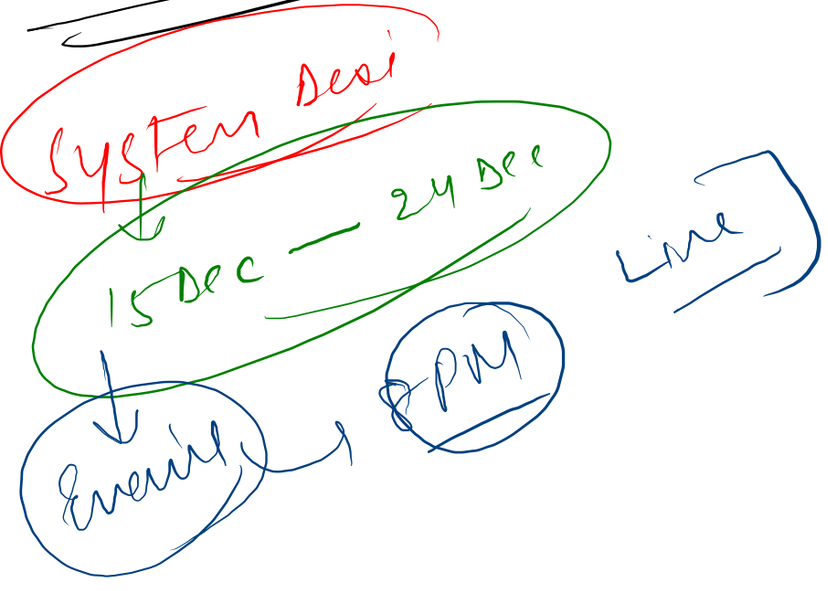

Abstraction
Abstract
important details

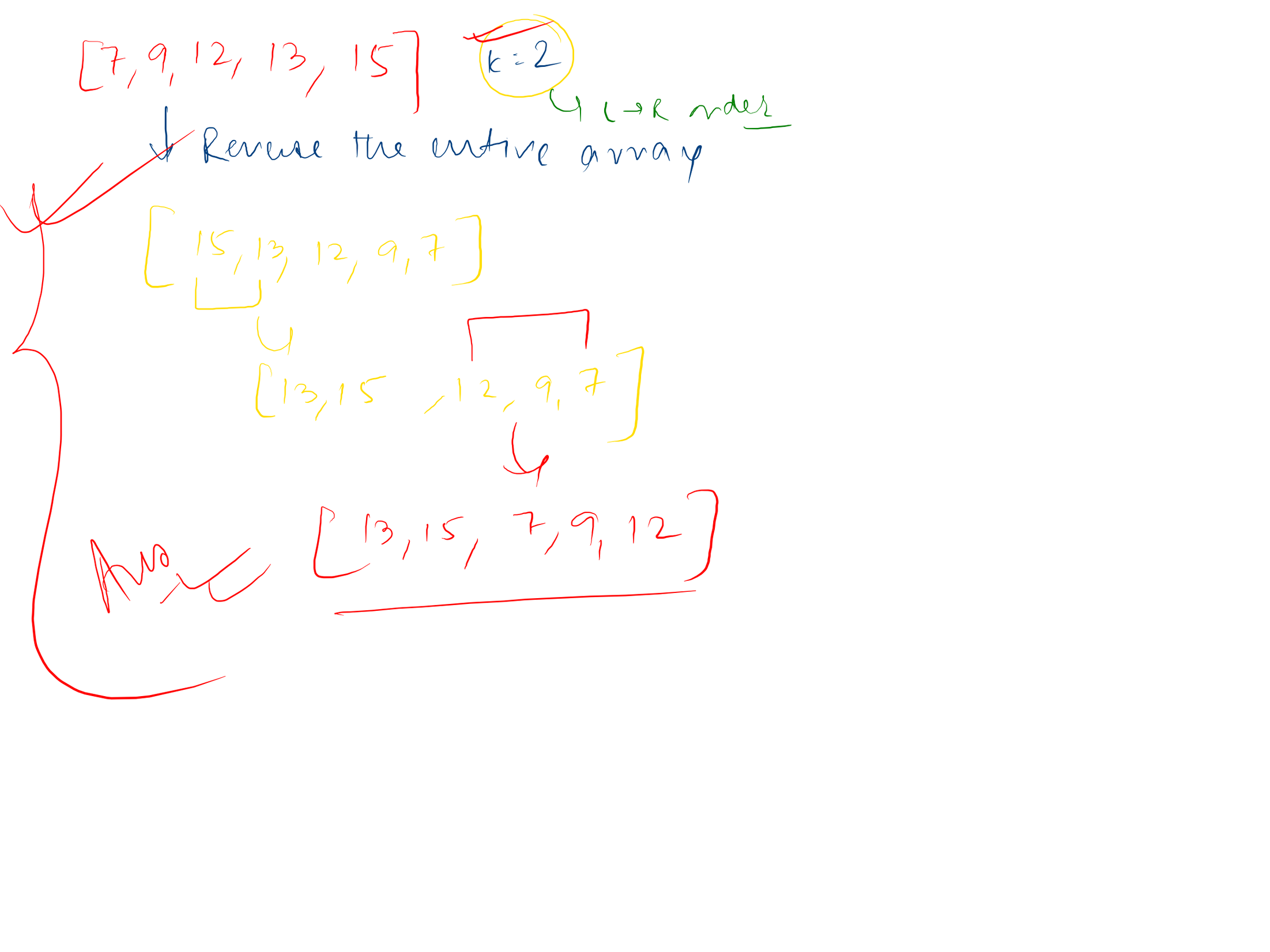

Facebook
UN
PIN
Submit
Abstraction
Passw
text
MFA
Abstracted

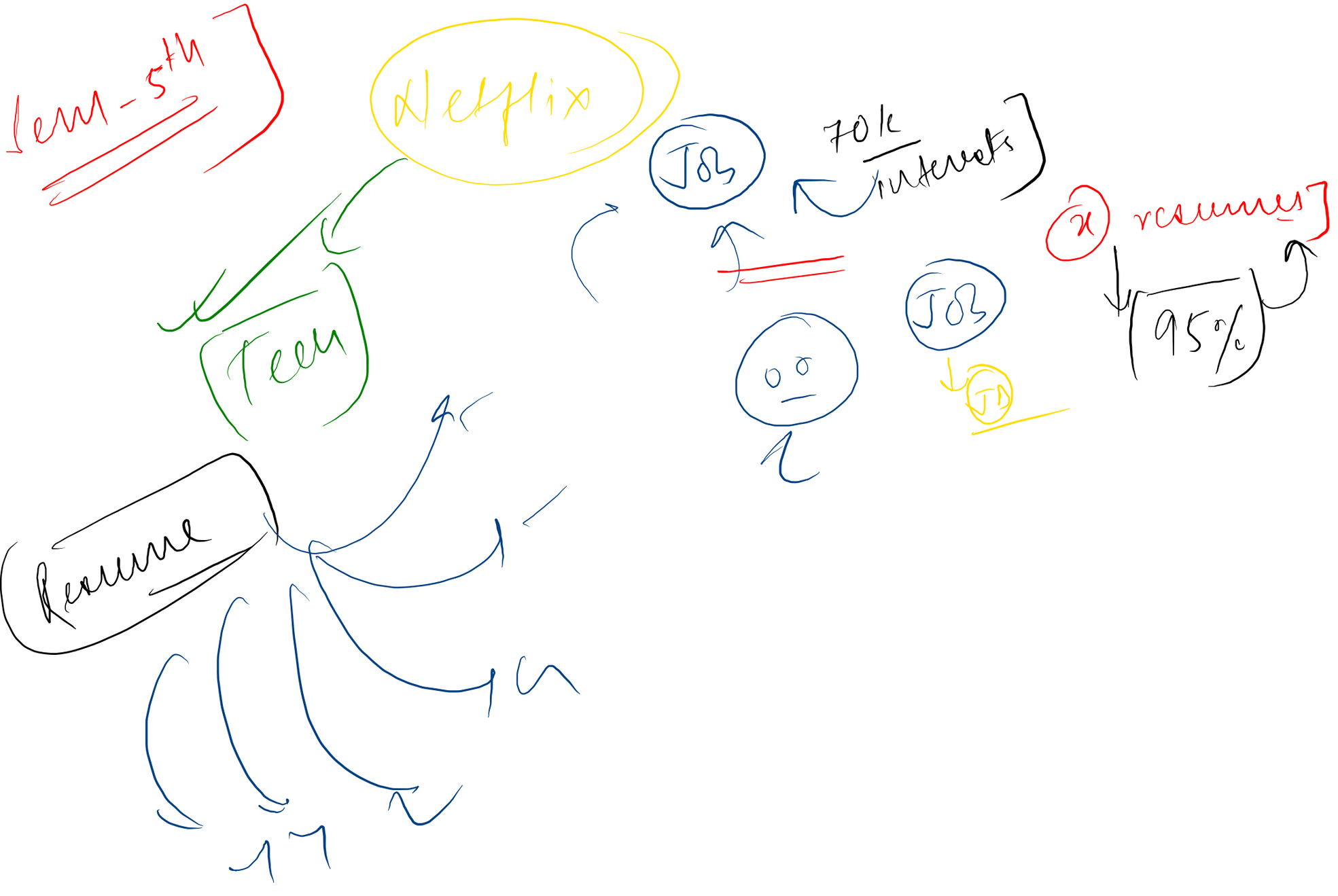

OOP
what
Abstraction
why
security
Java
interface/abstract classes
Python
Cyber

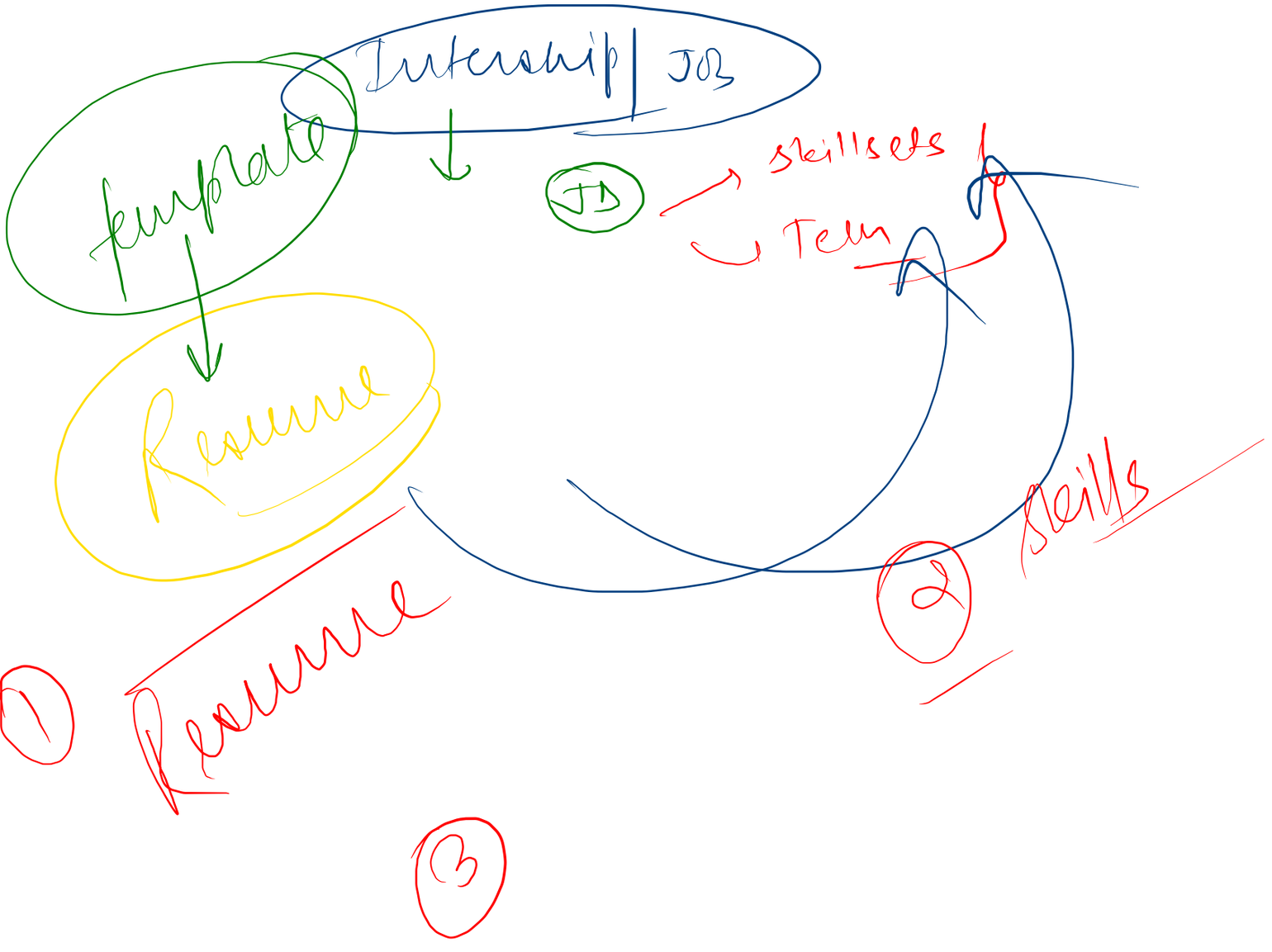
```
class pwskills(ABC):

    @abstractmethod
    def data_base(self):
        pass

    @abstractmethod
    def user_emailid(self, email_id):
        pass

    @abstractmethod
    def check_enrollment(self, user_id, class_id):
        pass

    @abstractmethod
    def check_lab_uses(self, user_id):
        pass

    @abstractmethod
    def check_internship(self, user_id):
        pass
```

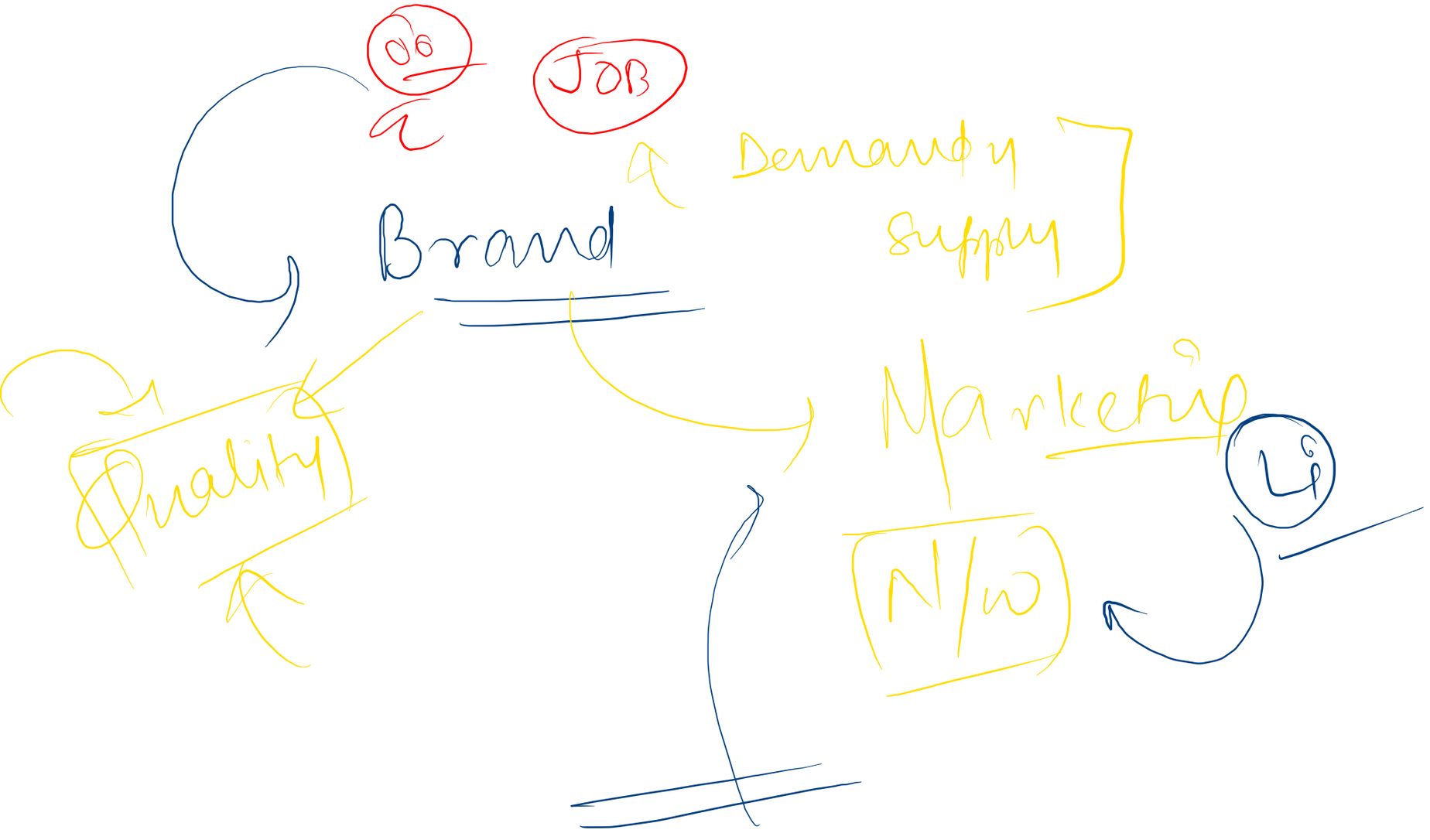
Last of Array Rotation Logic

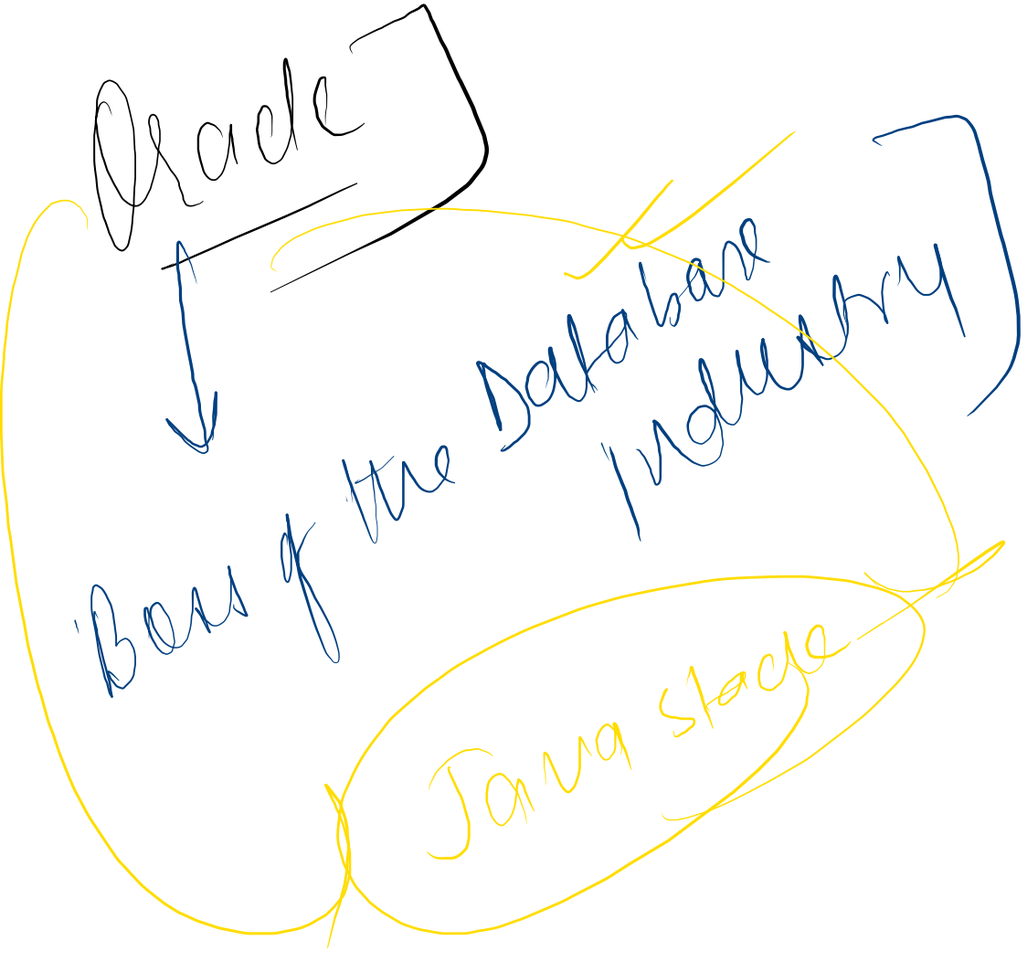


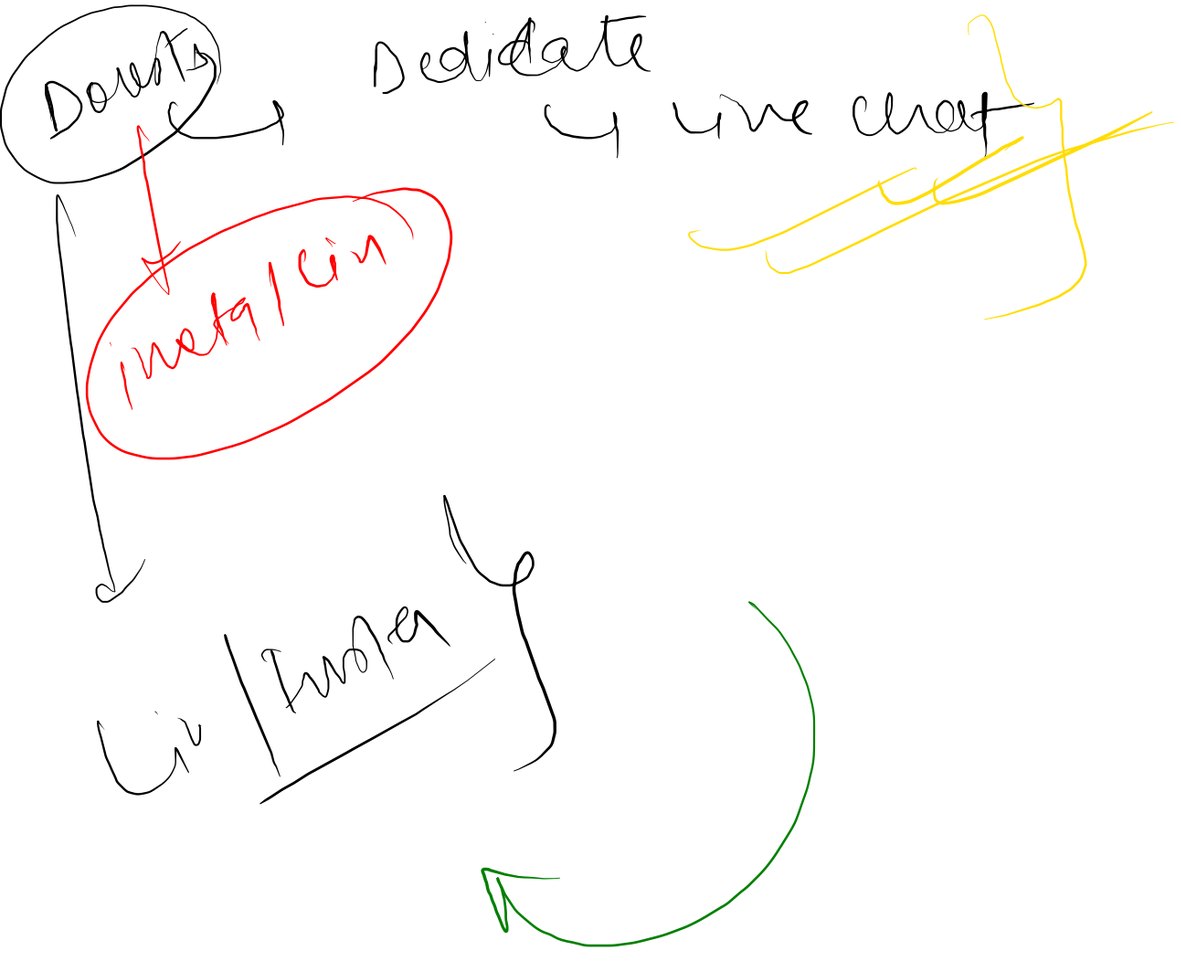

[7, 9, 12, 13, 15]
k = 2
1 → k order
Reverse the entire array
[15, 13, 12, 9, 7]
[13, 15, 12, 9, 7]
[13, 15, 7, 9, 12]
Ans


Netflix
Team
Resume
70k
interviews
95%


Internship | JOB
template
JD
skillsets
Tem
Resume
Resume
skills
1
2
3

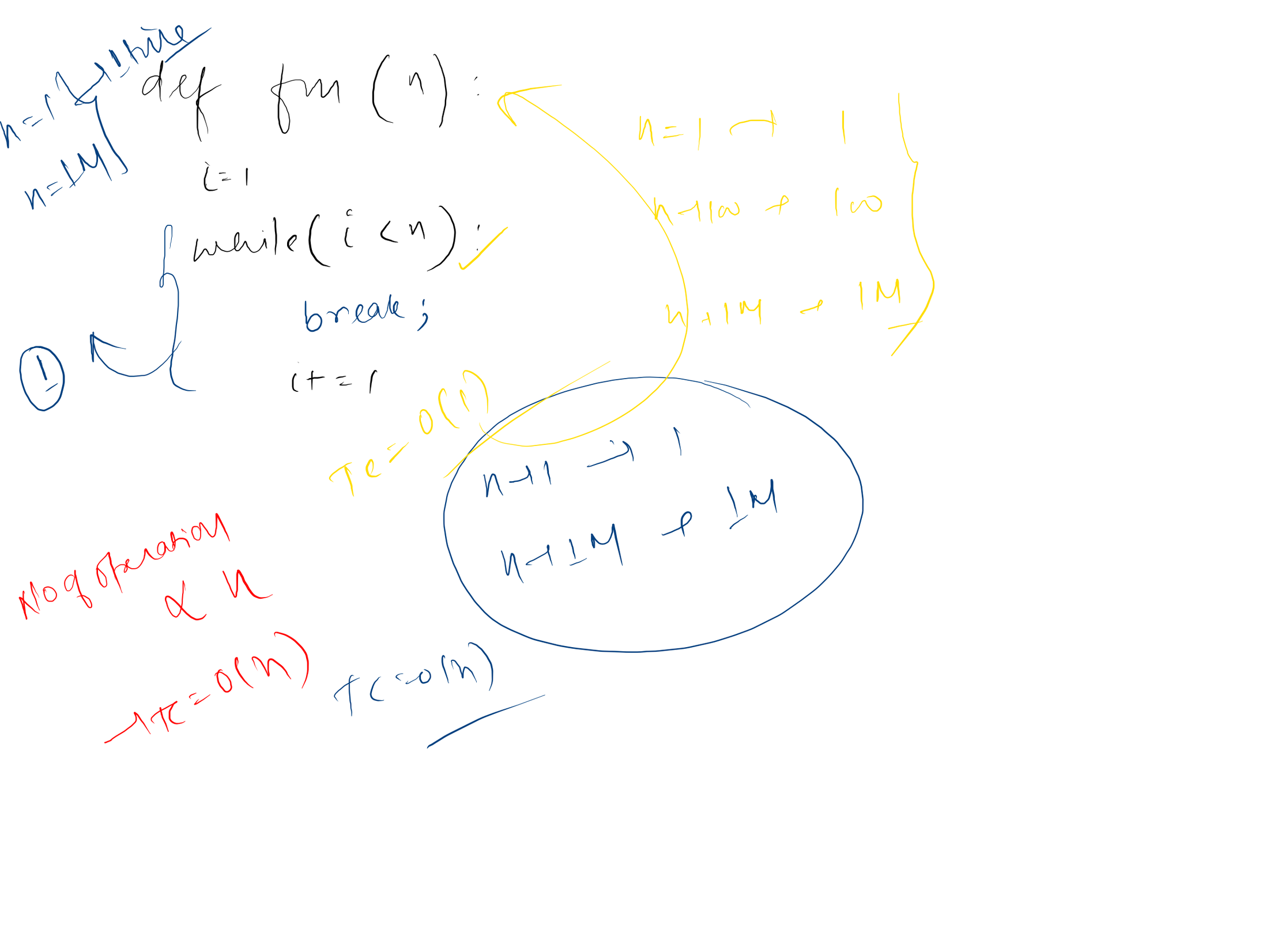

Job
Demand y
Supply
Brand
Quality
Marketing
N/w


Oracle
Boss of the Database industry
Java stack


Doubts
Dedicate
live chat
input/cin
Loops

Time complexity

↳ No. of operations $\propto$ input size


n=1 while
n=1M
def fun(n):
i=1
while(i<n):
break;
i+=1
1
n=1 → 1
n=1ω → 1ω
n=1M → 1M
TC=O(1)
n=1 → 1
n=1M → 1M
No of operation
α n
TC=O(n)
TC=O(n)

```
def fn(n):

    j = 1
    k = 2
    while ( j < n )
        k += 1
        j += 1
```

n times

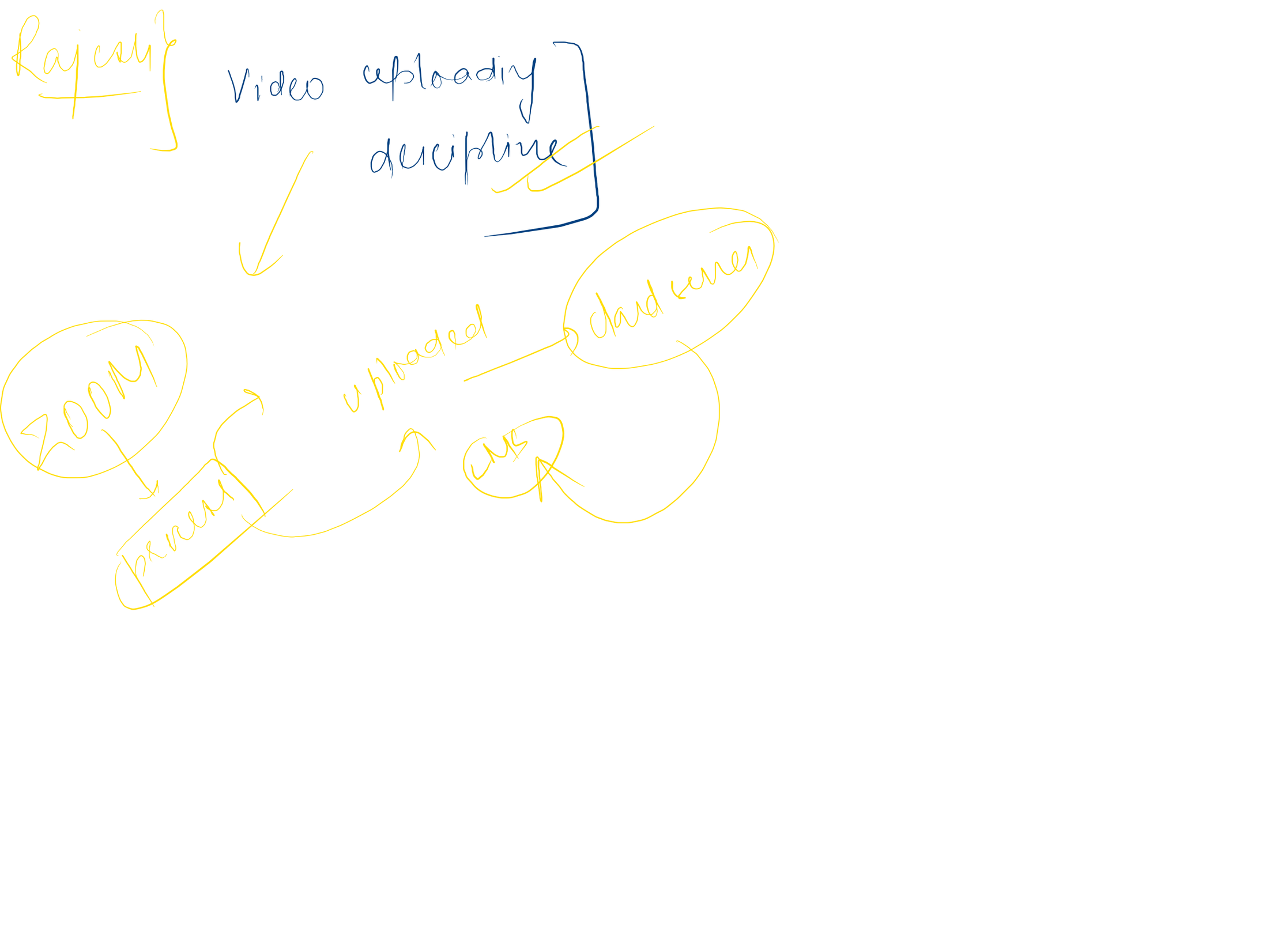
No of operation

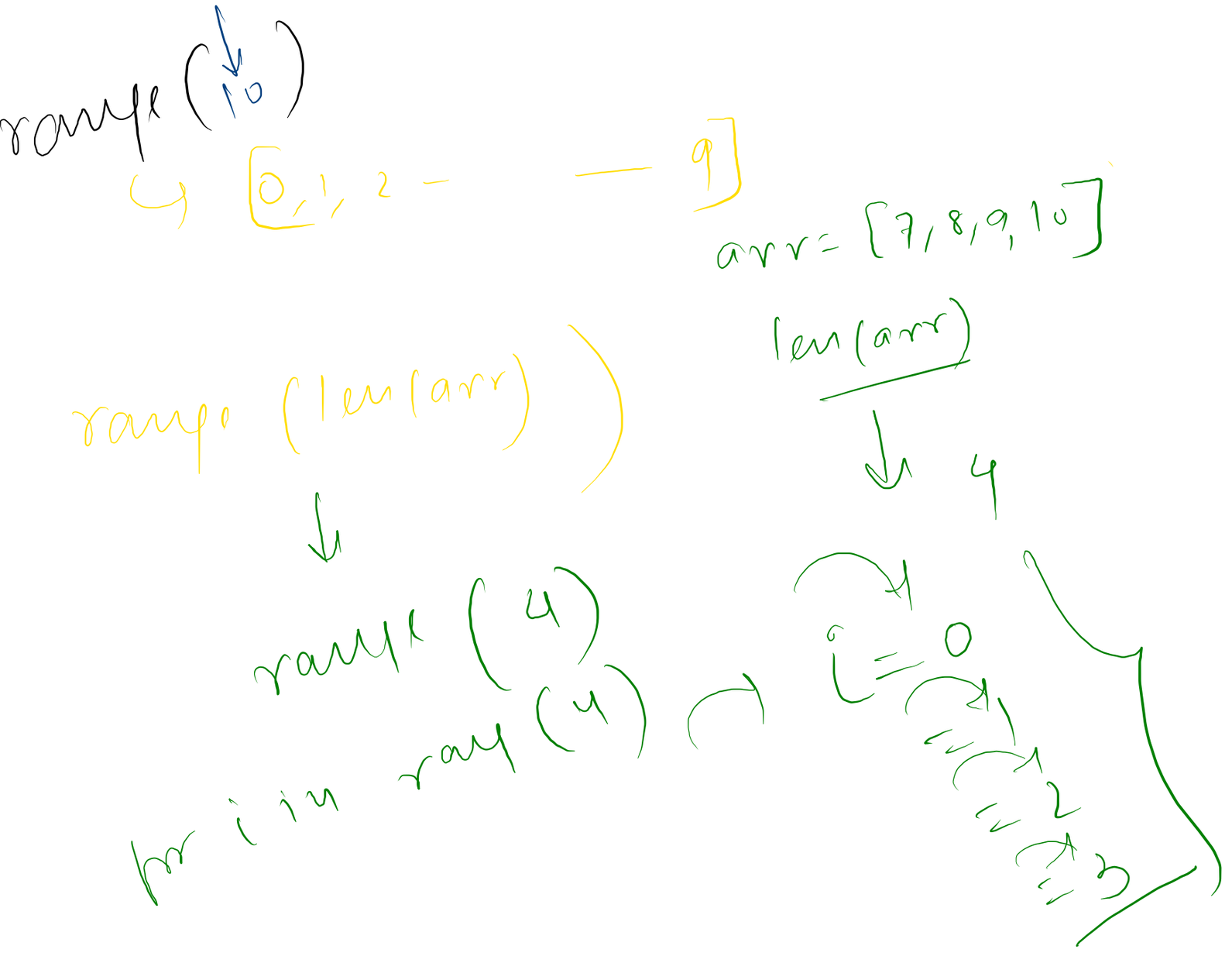
n = 1 → 1

n = 100 → 100

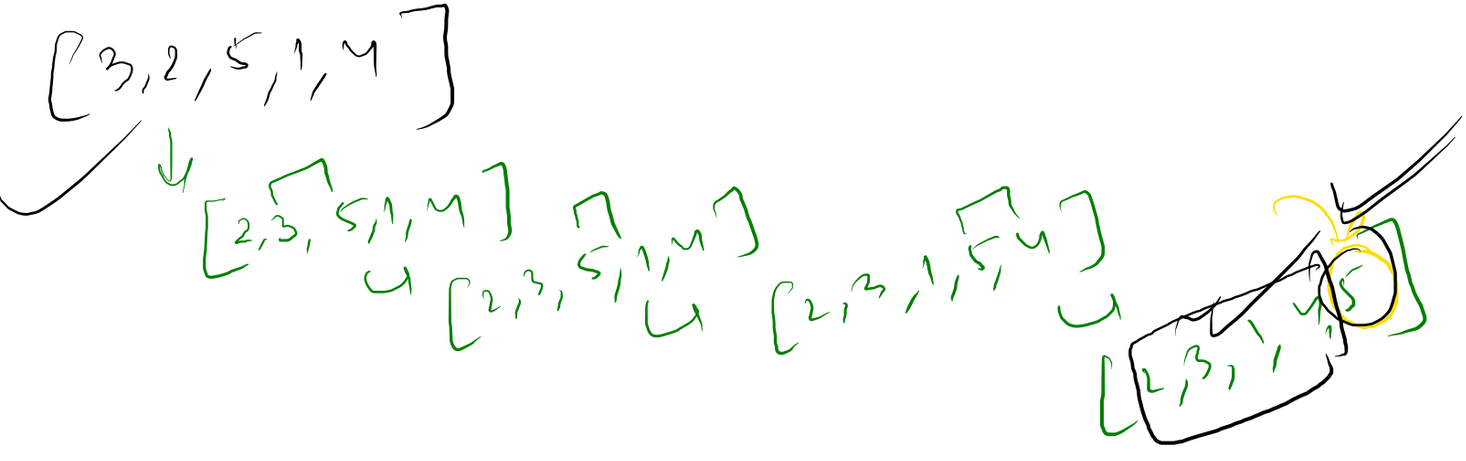
n = 14 → 14

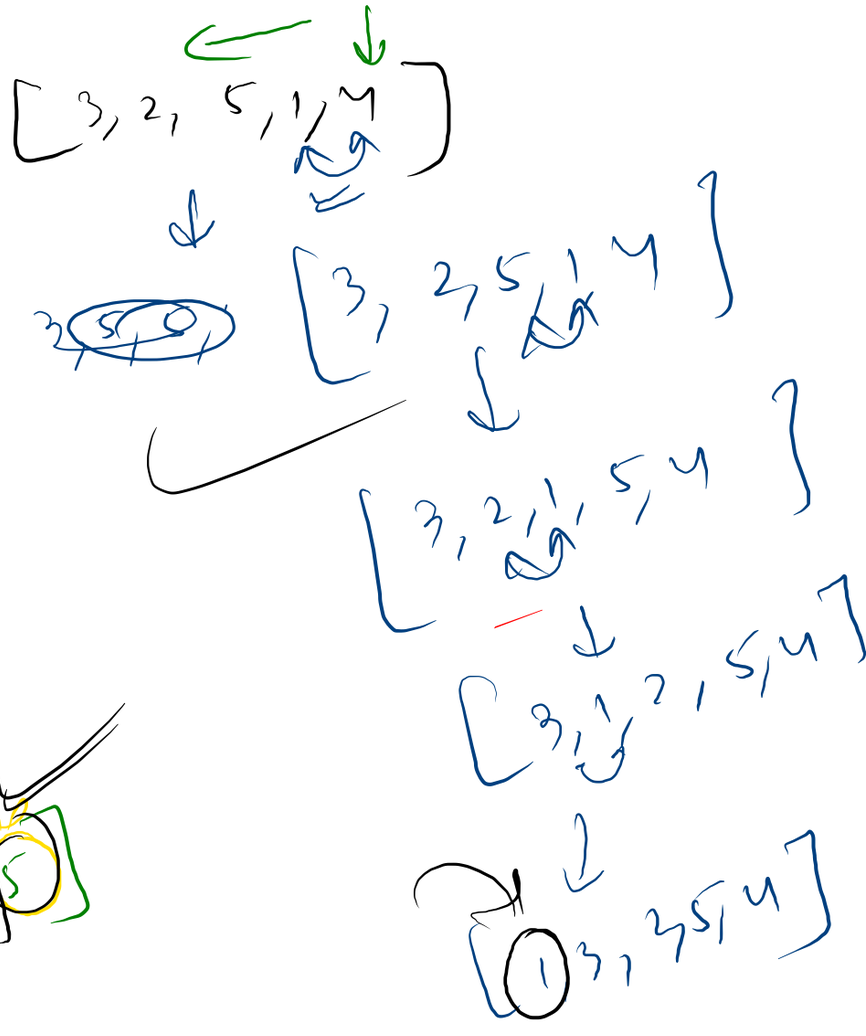
TC => O(n)


Rajesh
Video uploading discipline
Zoom
Recording
uploaded
Hard drive
IMP


range(10)
[0, 1, 2 — — 9]
arr = [7, 8, 9, 10]
len(arr)
4
range(len(arr))
range(4)
for i in range(4)
i = 0
1
2
3

```python
def bubbleSortOptimized(arr):
  for i in range(len(arr)-1, 0 , -1):
    isSorted = True
    for j in range(i):
      if(arr[j]>arr[j+1]):
        isSorted = False
        arr[j],arr[j+1] = arr[j+1],arr[j]


    if isSorted :
      print("Array is already sorted")
      break
```





range(len(arr))

↓

range(5)

0, 1, 2, 3, 4

arr = [7, 8, 9, 10, 11]

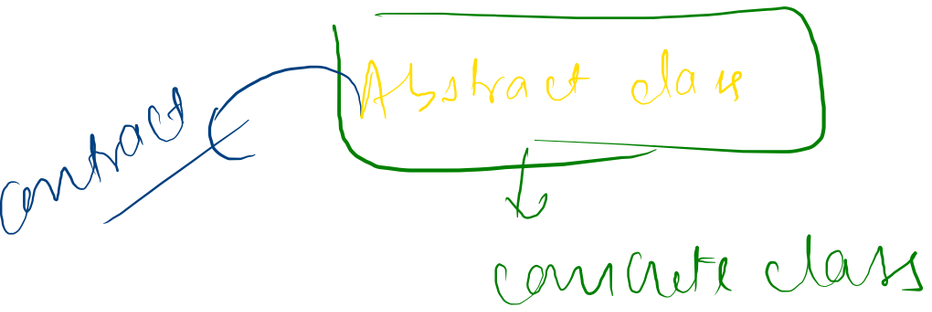
range(len(arr)-1, 0, -1)

start

end

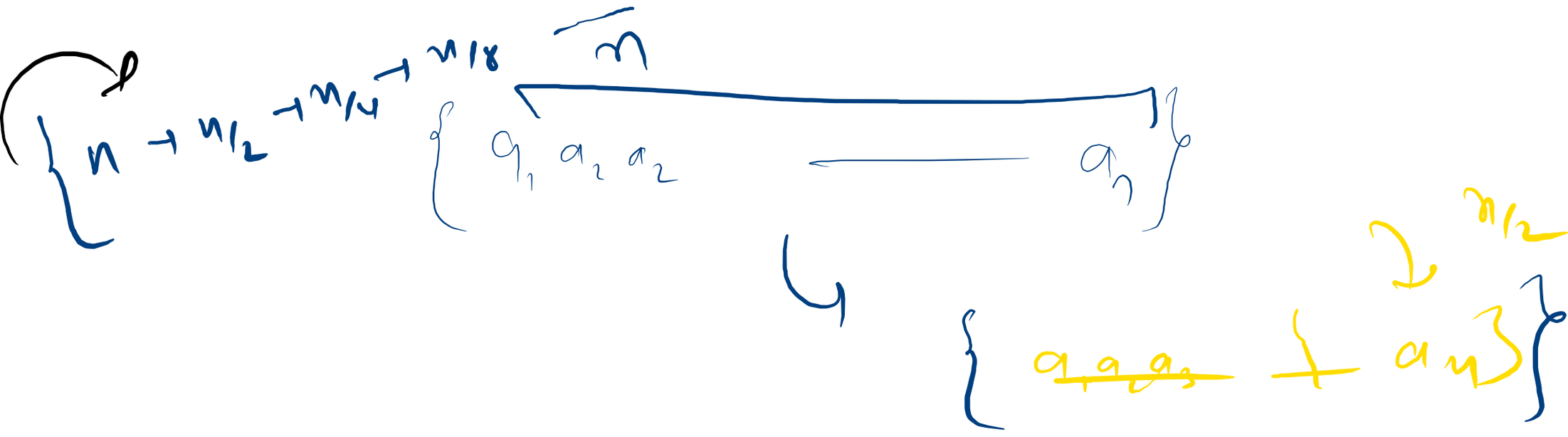
how many jump (→ reverse direction)

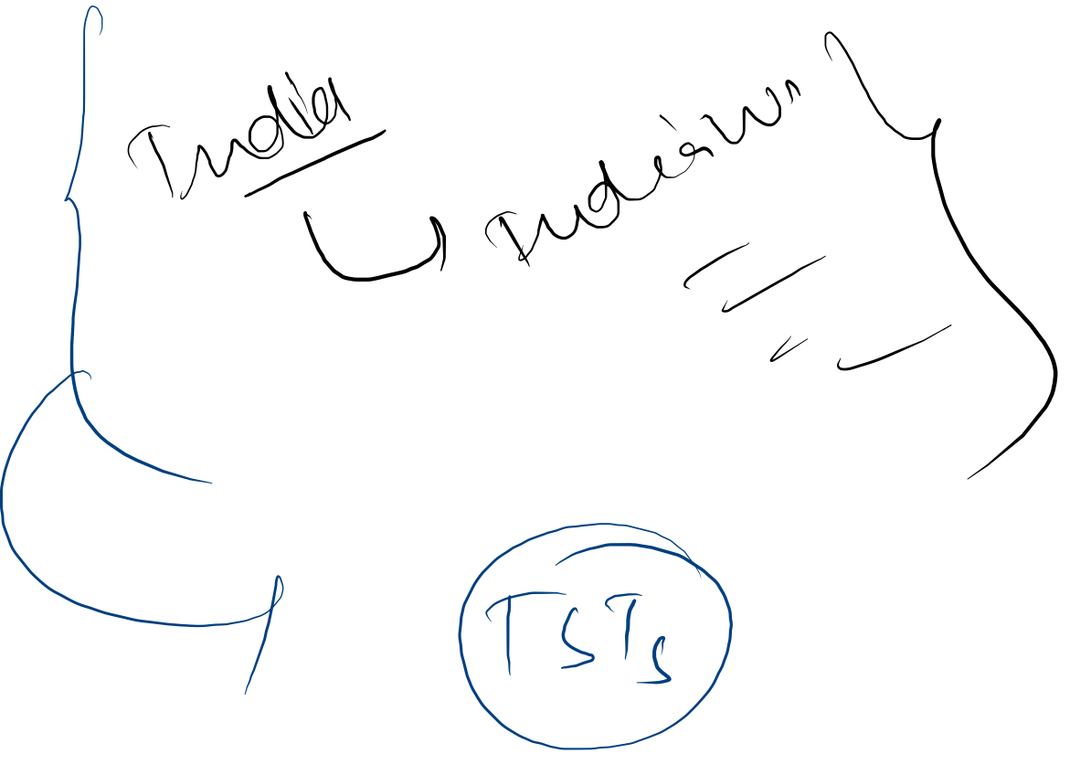
4 → 3 → 2 → 1 → 0


contract
Abstract class
concrete class

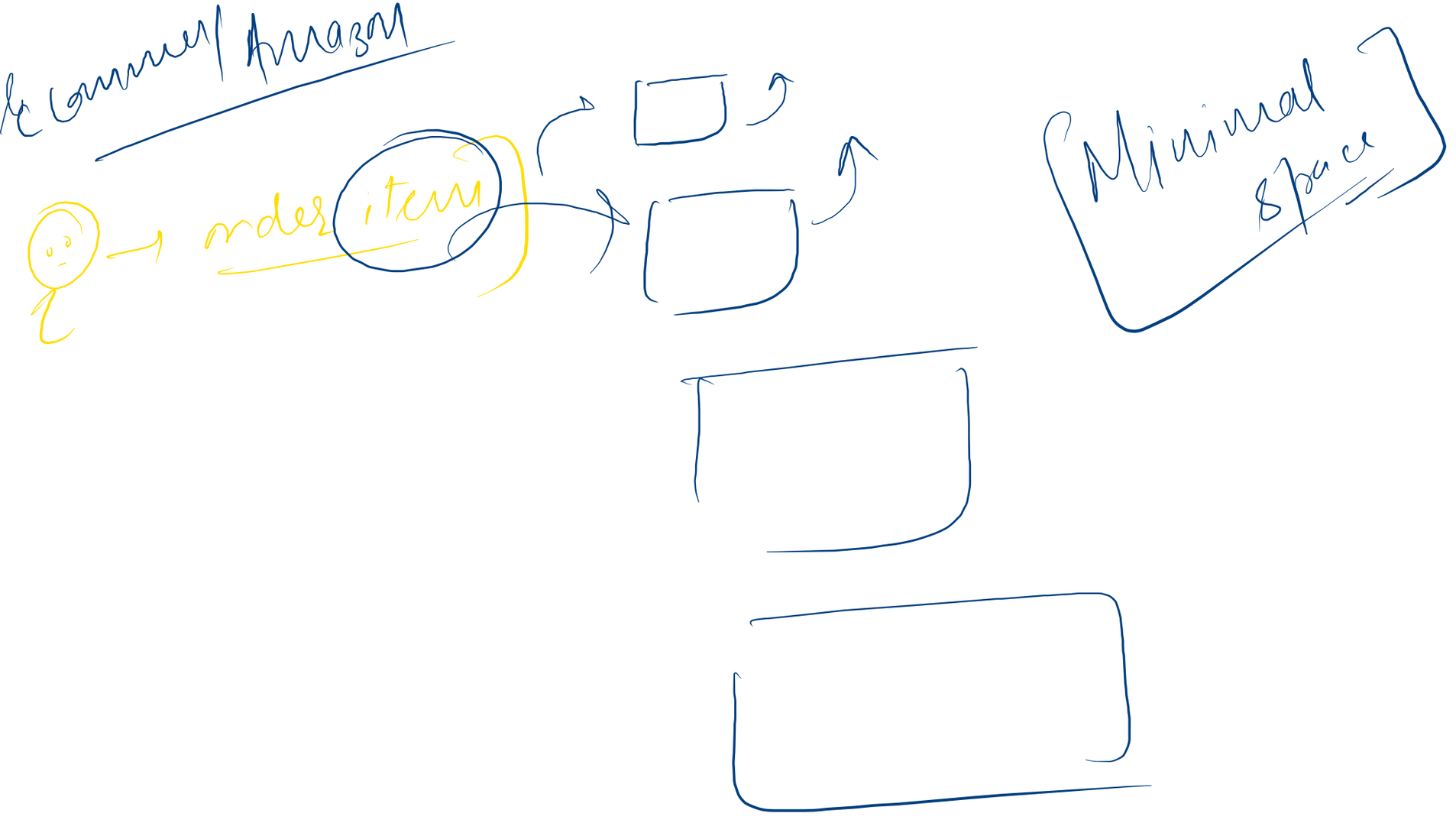
10:45 AM → Break

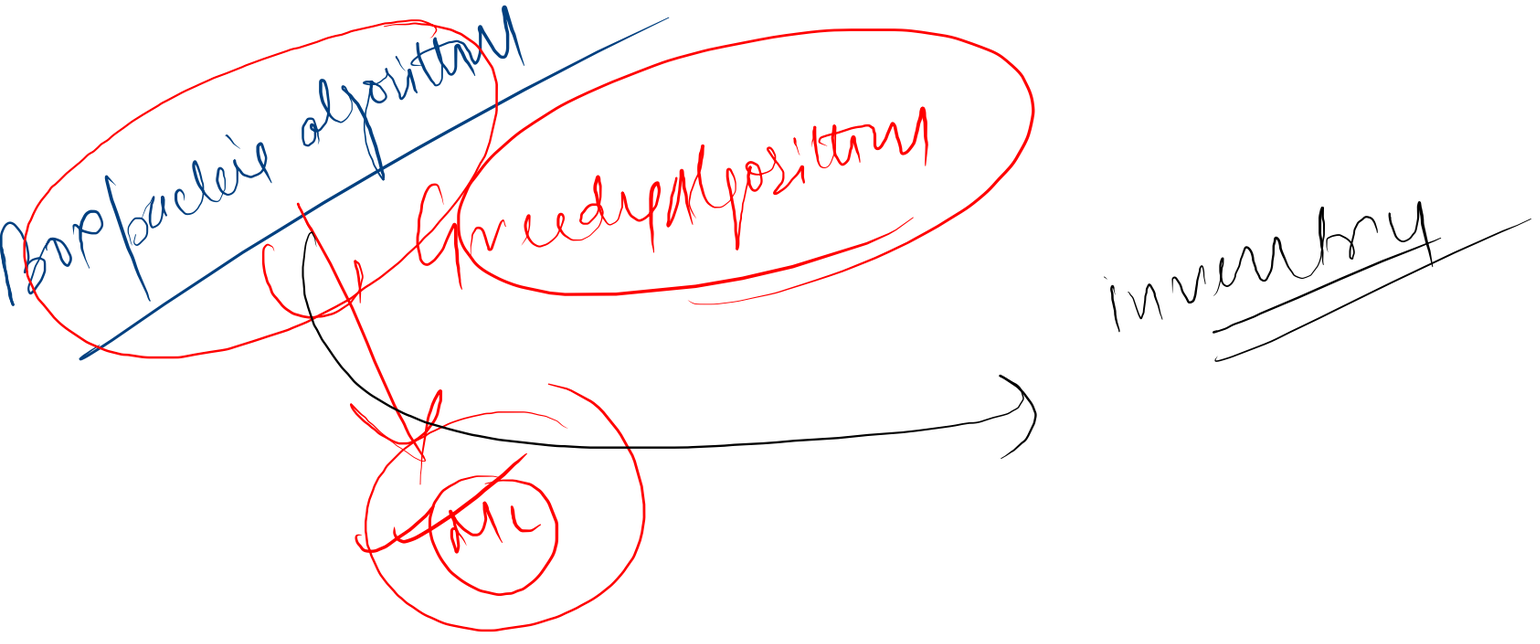
→ Solve more problems


n → n/2 → n/4 → n/8
m
a1 a2 a2 ——— an
n/2
a a a3 ⊥ an

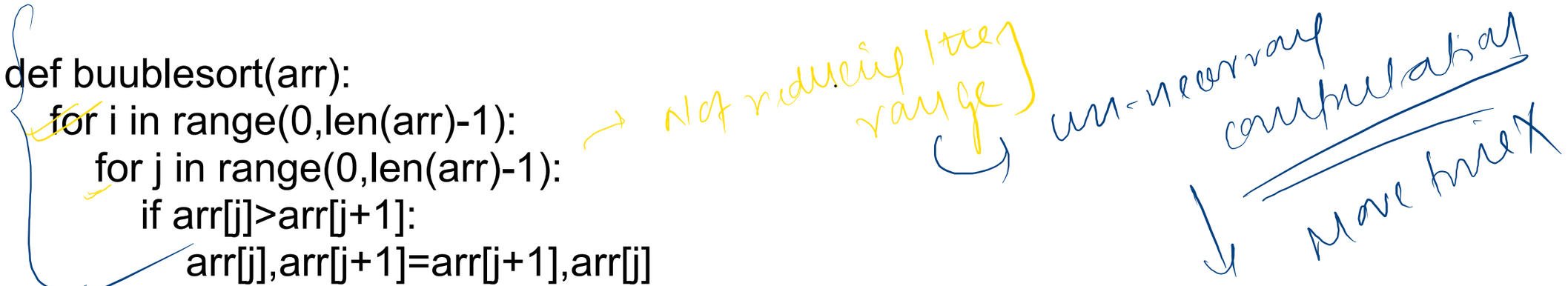

Trolley
Trolley
TSTs

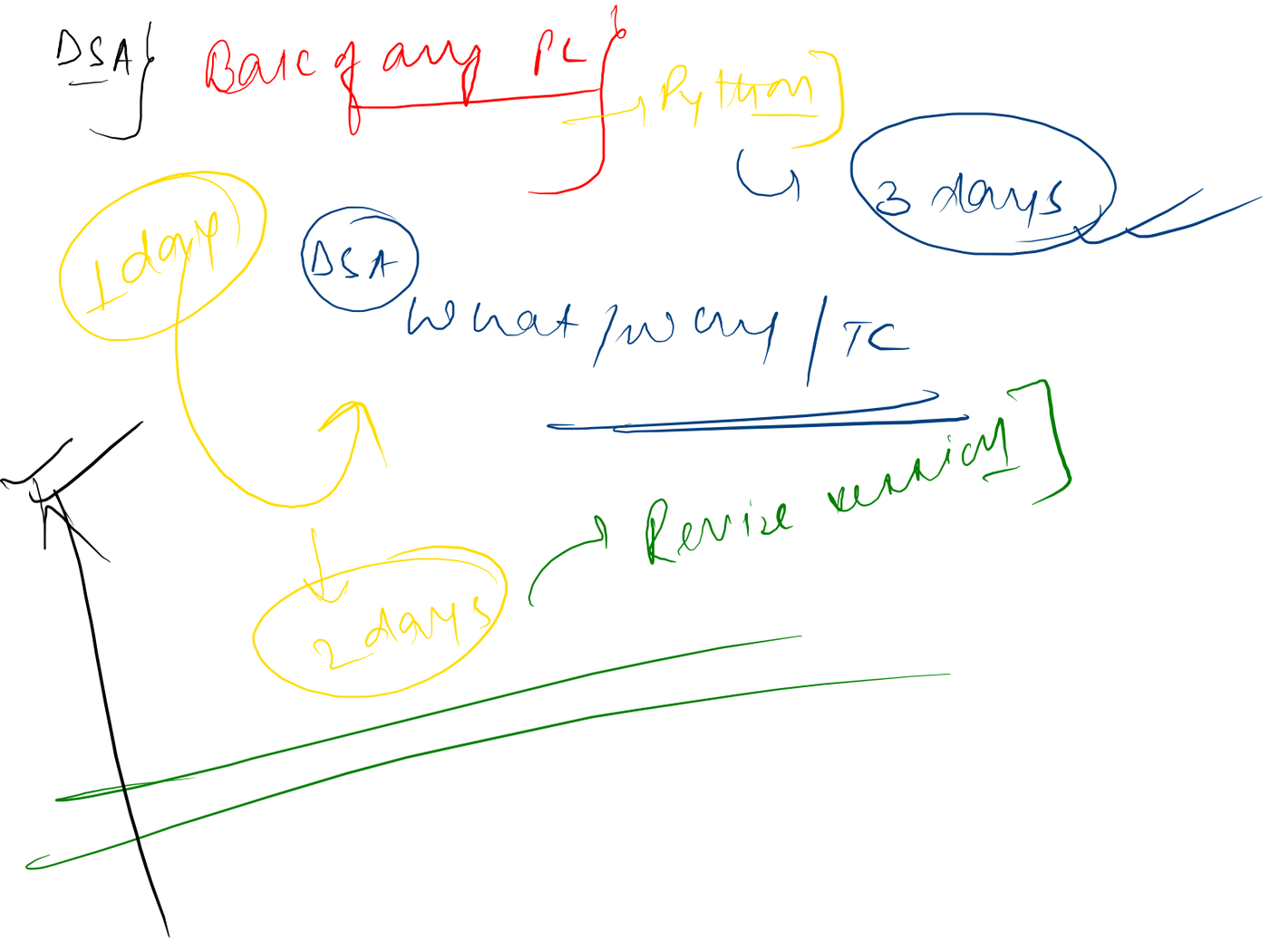

Python
DSA
Revision
Recording
MC


E commerce Amazon
order item
Minimal space


Вероятностный алгоритм
Детерминированный
ML
инвентарь

```
def buublesort(arr):
    for i in range(0,len(arr)-1):
        for j in range(0,len(arr)-1):
            if arr[j]>arr[j+1]:
                arr[j],arr[j+1]=arr[j+1],arr[j]
```



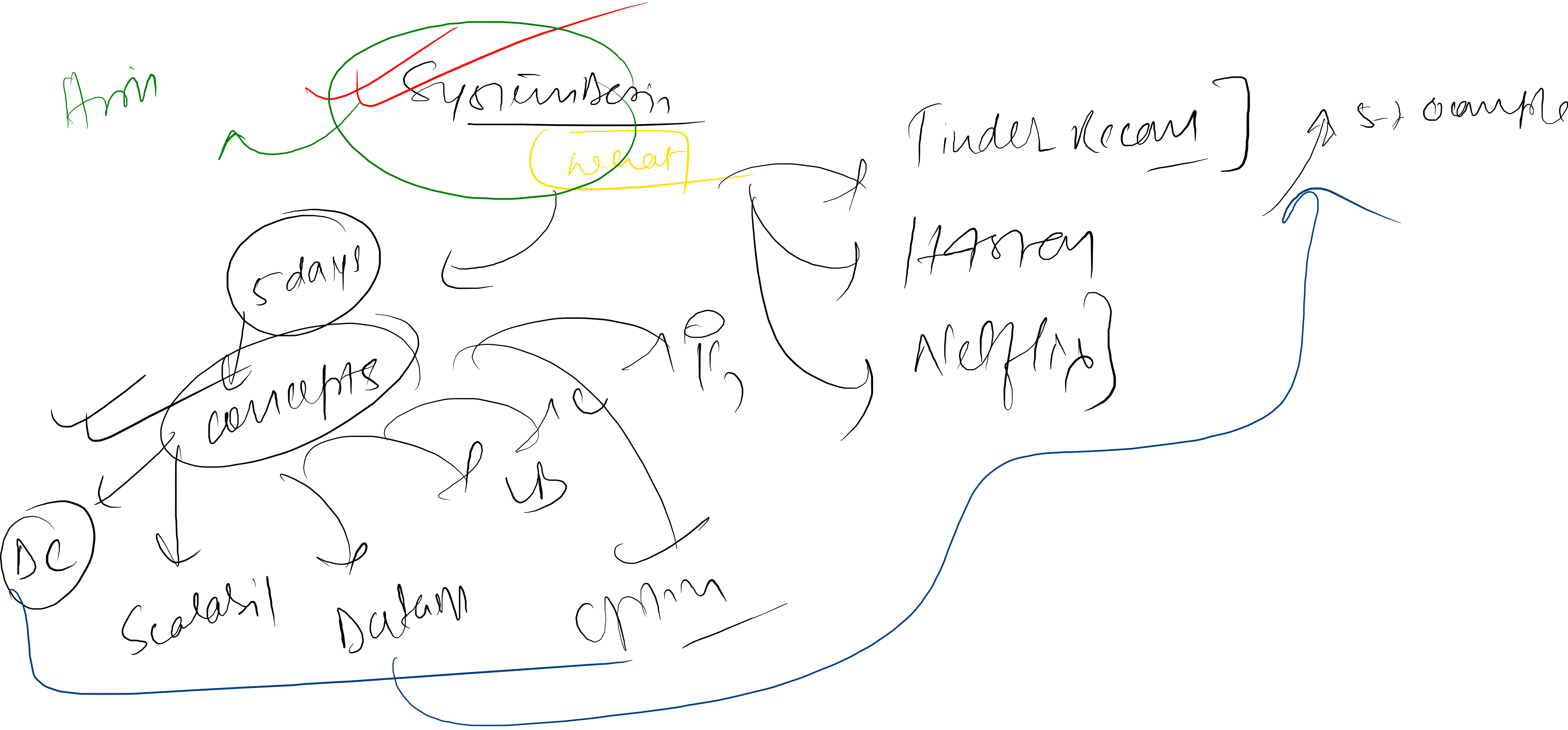

DSA
Basic of any PL
Python
3 days
1 day
DSA
What / Why / TC
2 days
Revise version

System Design

↳ codeX

↓

Product is designed

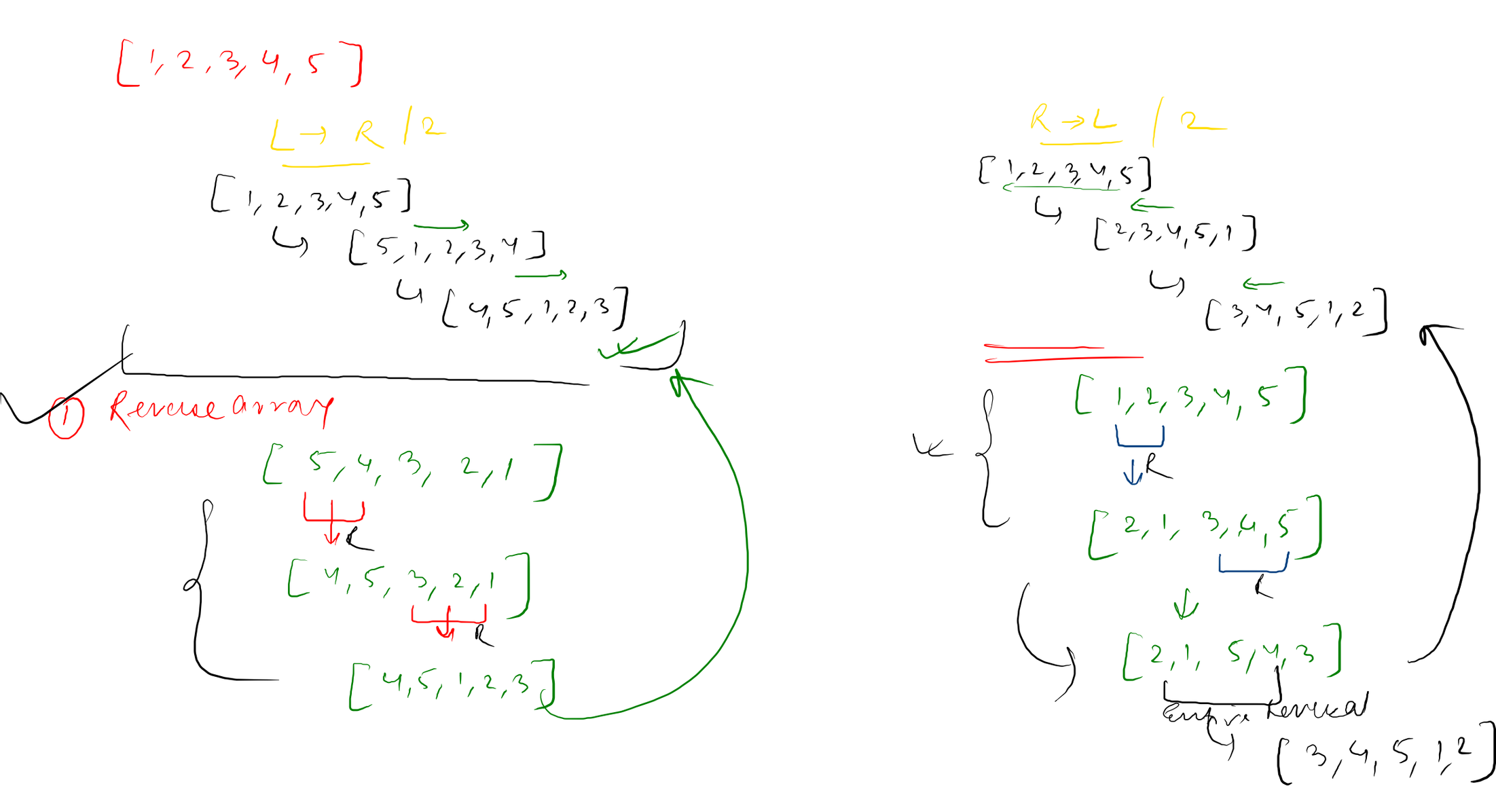


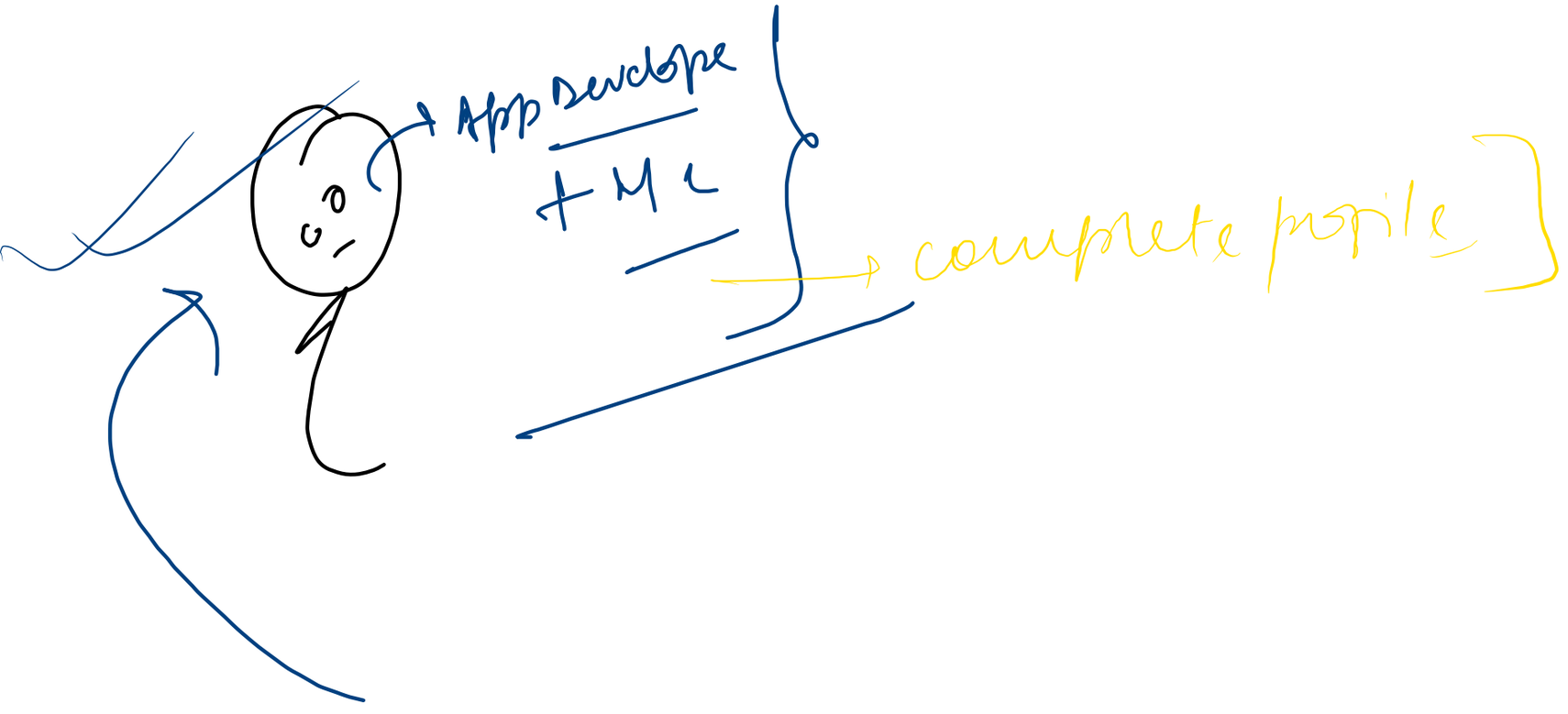
System Design
what
Tinder
Netflix
5 days
concepts
DC
example

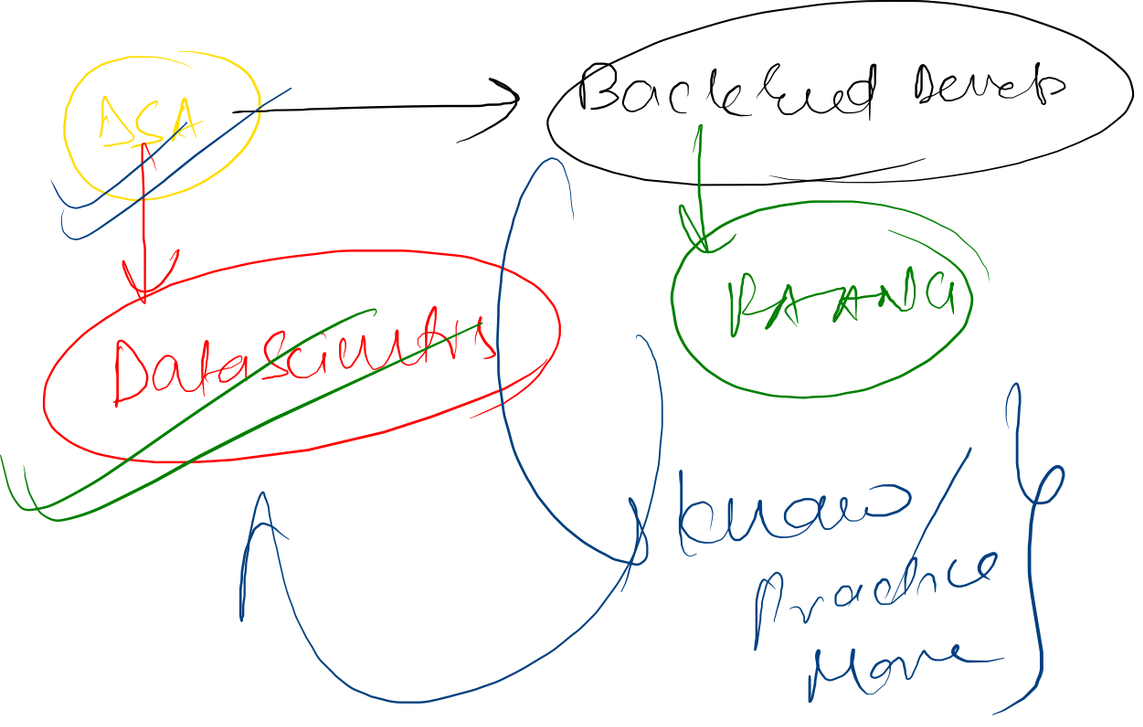
% → Modulous / Remainder

5 % 3 → 2

$$\begin{array}{r|l} 3 & 5 \\ & 3 \\ \hline & 2 \end{array}$$


[1, 2, 3, 4, 5]
L → R / 2
[1, 2, 3, 4, 5]
[5, 1, 2, 3, 4]
[4, 5, 1, 2, 3]
① Reverse array
[5, 4, 3, 2, 1]
R
[4, 5, 3, 2, 1]
R
[4, 5, 1, 2, 3]
R → L / 2
[1, 2, 3, 4, 5]
[2, 3, 4, 5, 1]
[3, 4, 5, 1, 2]
[1, 2, 3, 4, 5]
R
[2, 1, 3, 4, 5]
R
[2, 1, 5, 4, 3]
Reverse
[3, 4, 5, 1, 2]


App Developer
+ ML
complete profile


DSA
Backend Devel
RANG
Data Scientist
Learn
Practice
More