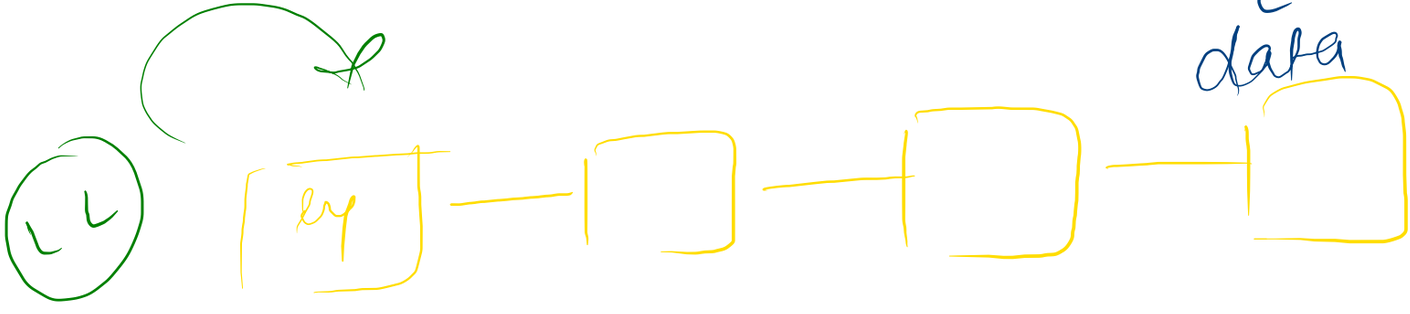
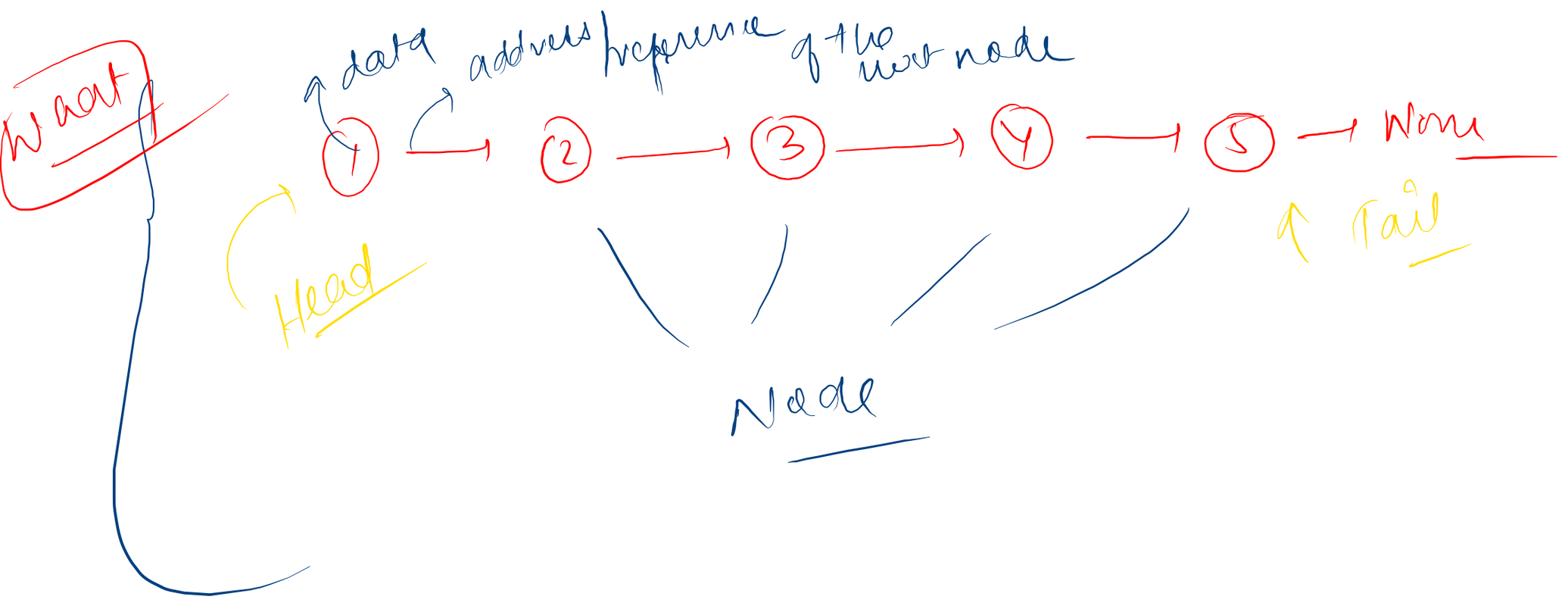
Linked List → Linear?

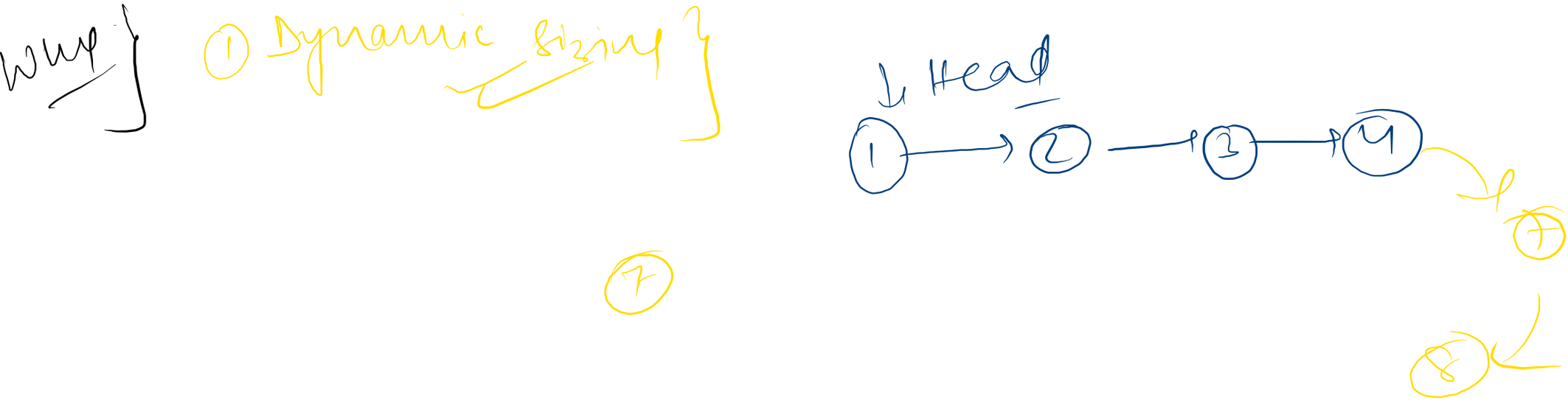
(DS) → Linear data structure

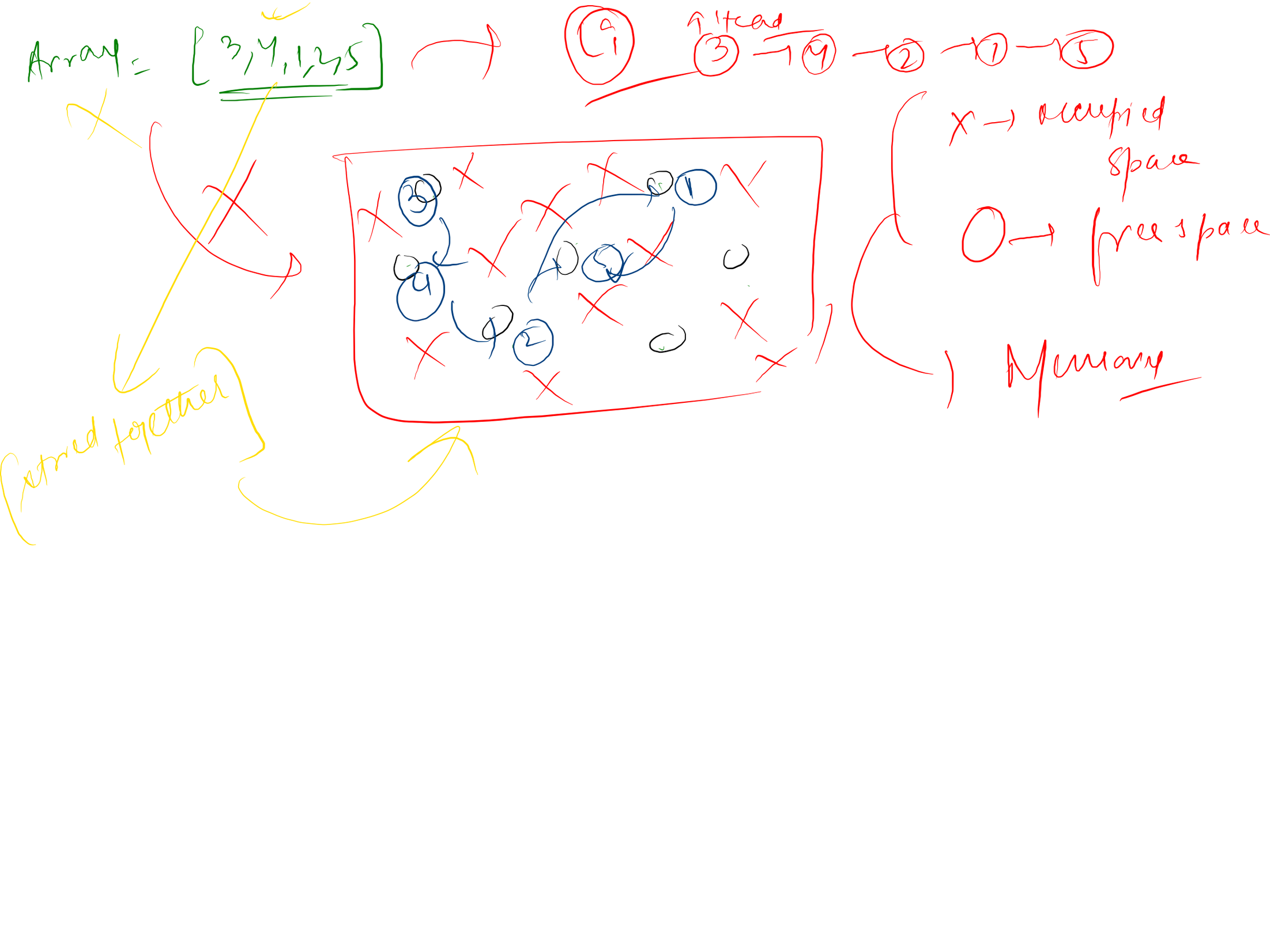
LL → Node → next Node

data

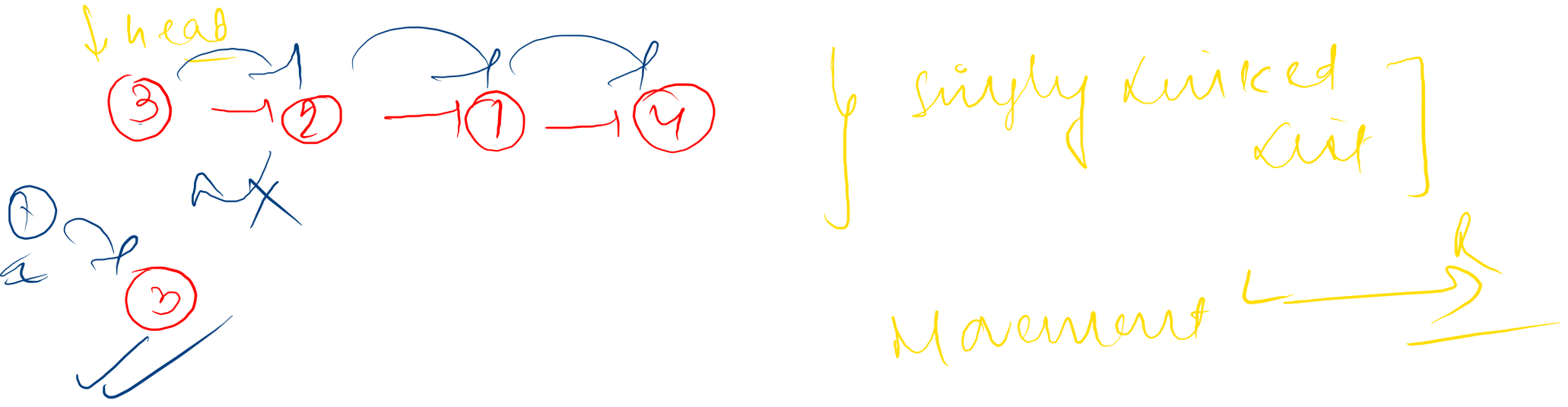



data
address reference of the next node
1
2
3
4
5
None
Head
Tail
Node

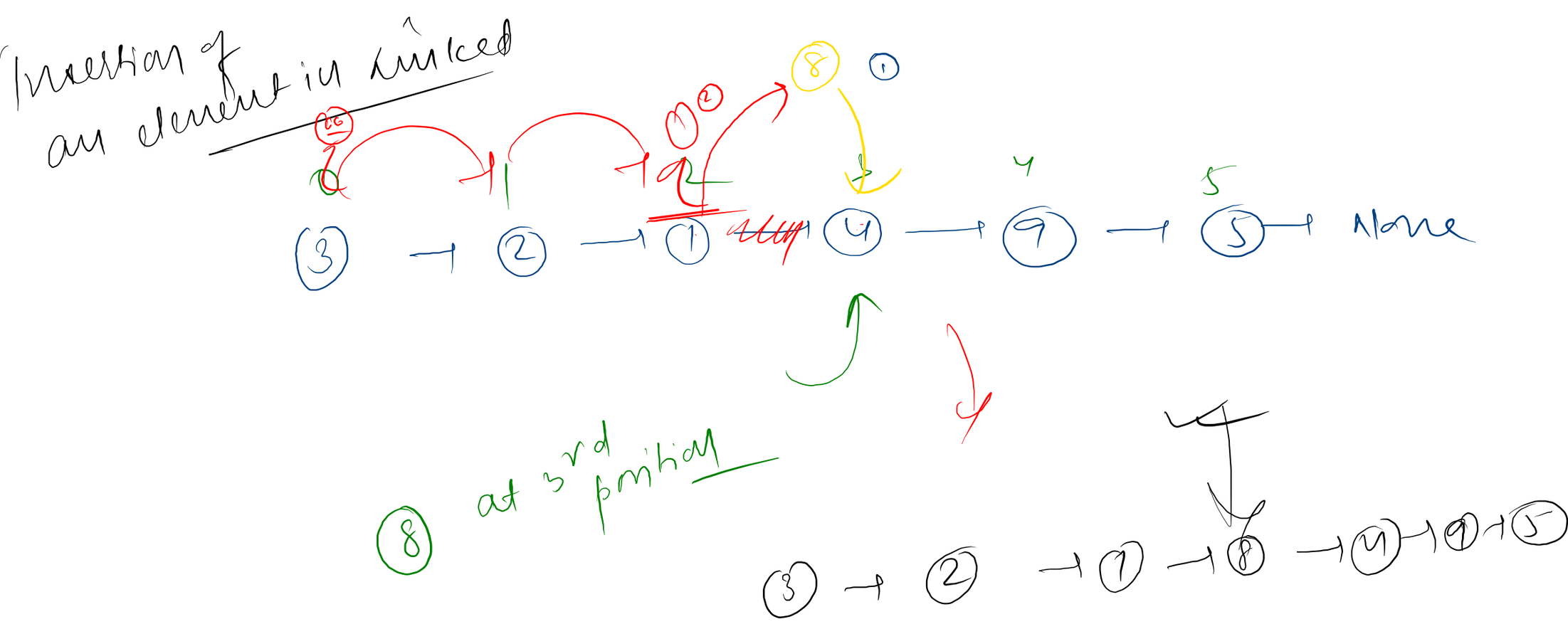

Dynamic Sizing
Head
1
2
3
4
7
8


Array :- [3,4,1,2,5]
head
X → occupied space
O → free space
Memory
stored together

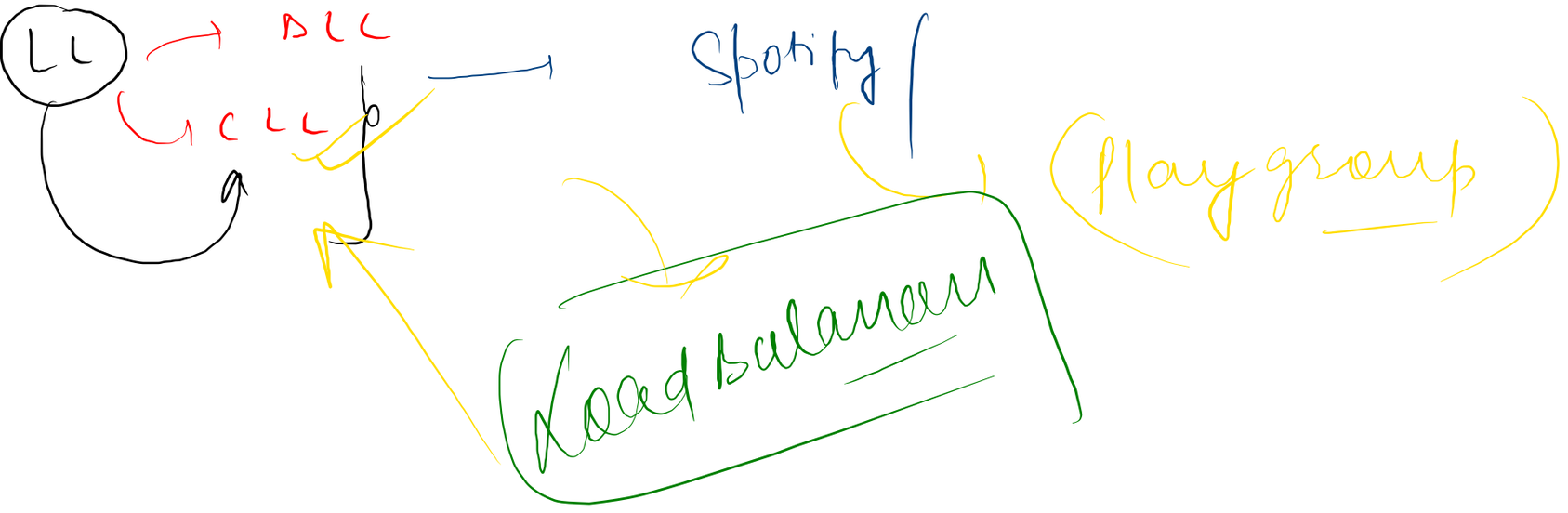
```
class Node :

  def __init__(self,data=None, next=None):
    self.data = data
    self.next = next

  #Method to set the data value
  def setData(self,data):
    self.data = data


  #method to get the data value
  def getData(self):
    return self.data


  #Method of set the next
  def setNext(self, next):
    self.next = next


  #Method to get the next
  def getNext(self):
    return self.next
```

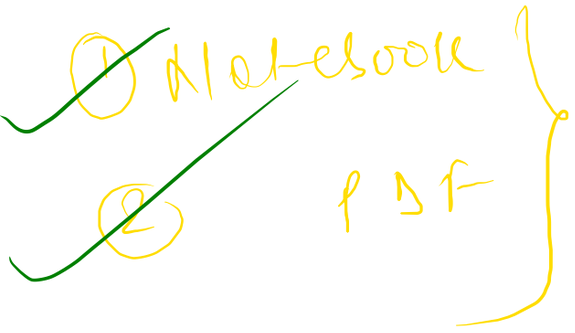

head
3
2
1
4
1
3
Singly Linked List
Movement

Insertion of an element in Linked

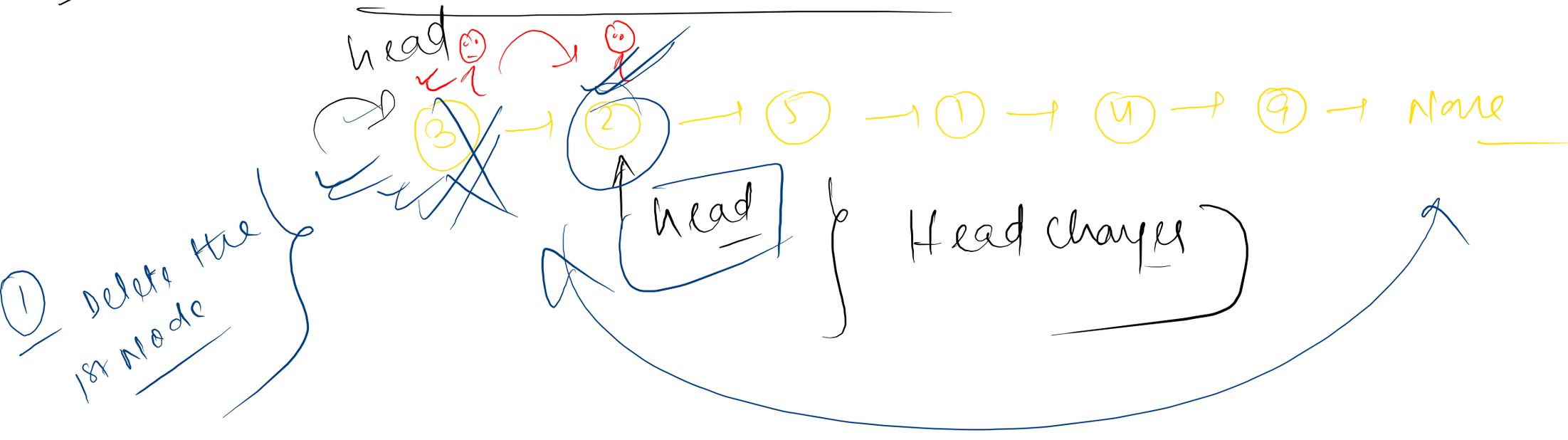


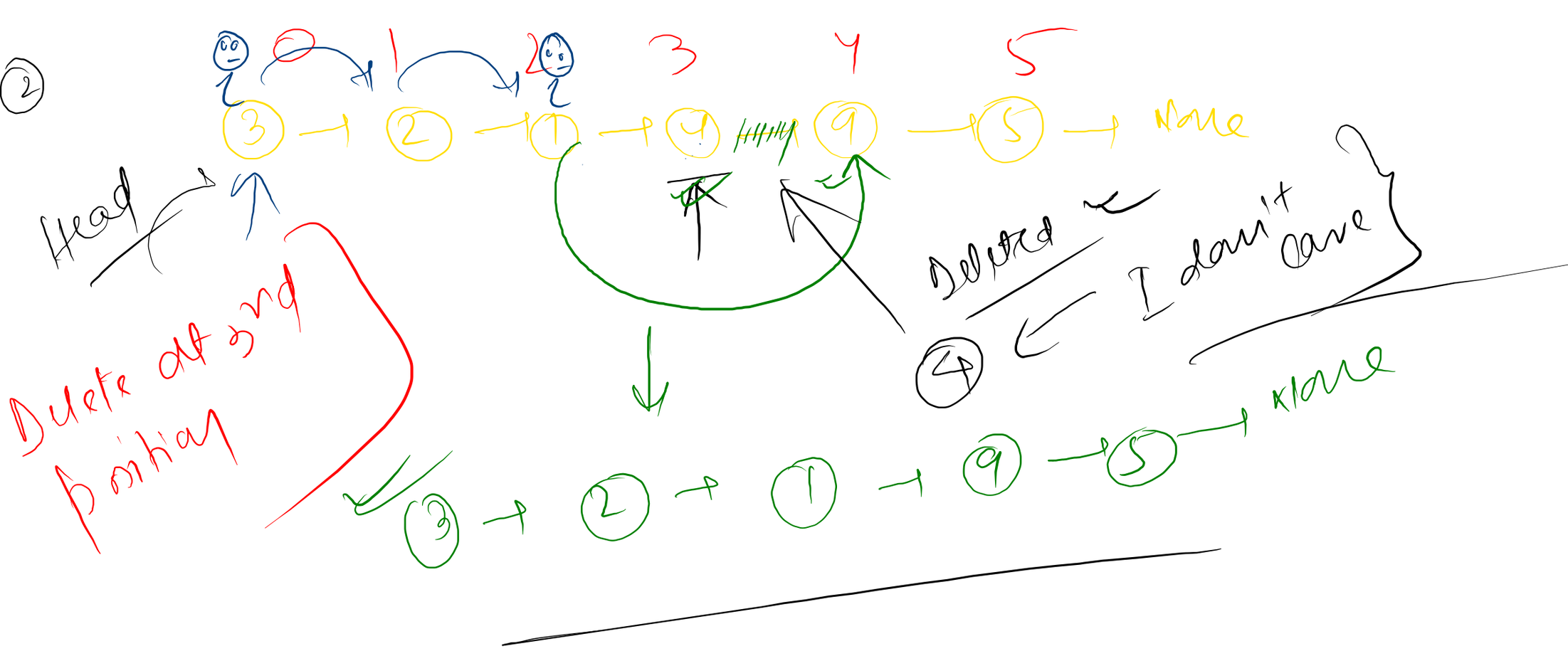
8 at 3rd position


LL
DLC
CLLP
Spotify
Load balancer
Playgroup


1 Notebook
2
pdf

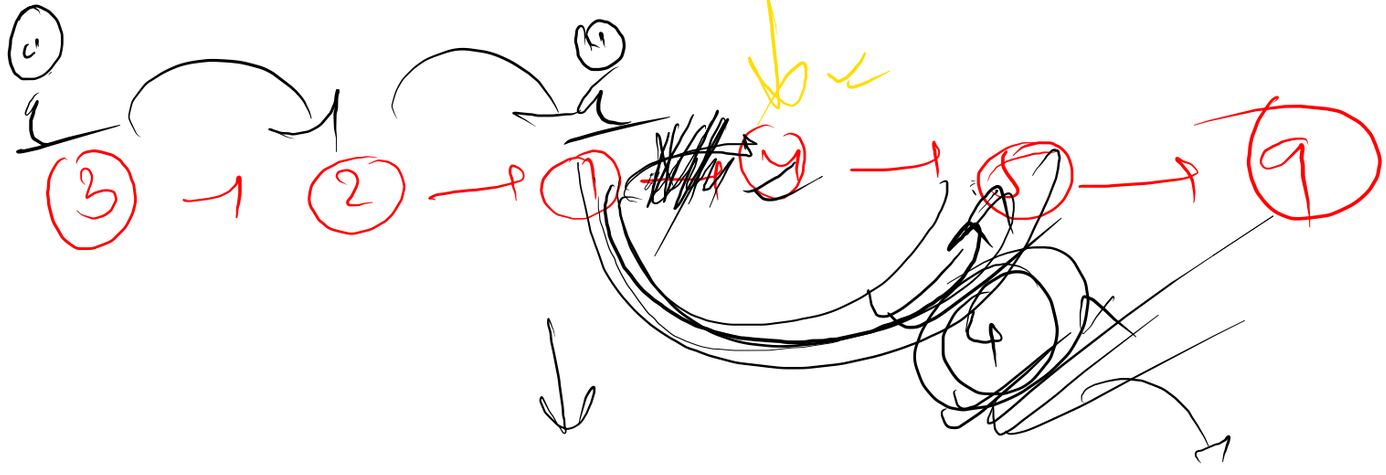
Delete a node at k^{th} position

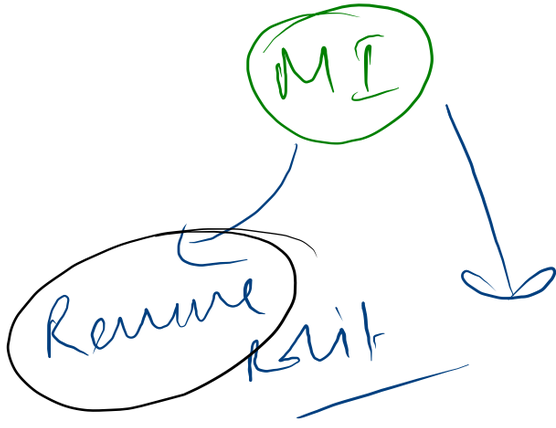



2
0
1
2
3
4
5
3 → 2 → 1 → 9 → 9 → 5 → None
Head
Delete at 3rd position
Deleted
I don't care
4
3 → 2 → 1 → 9 → 5 → None

Delete a node from LL

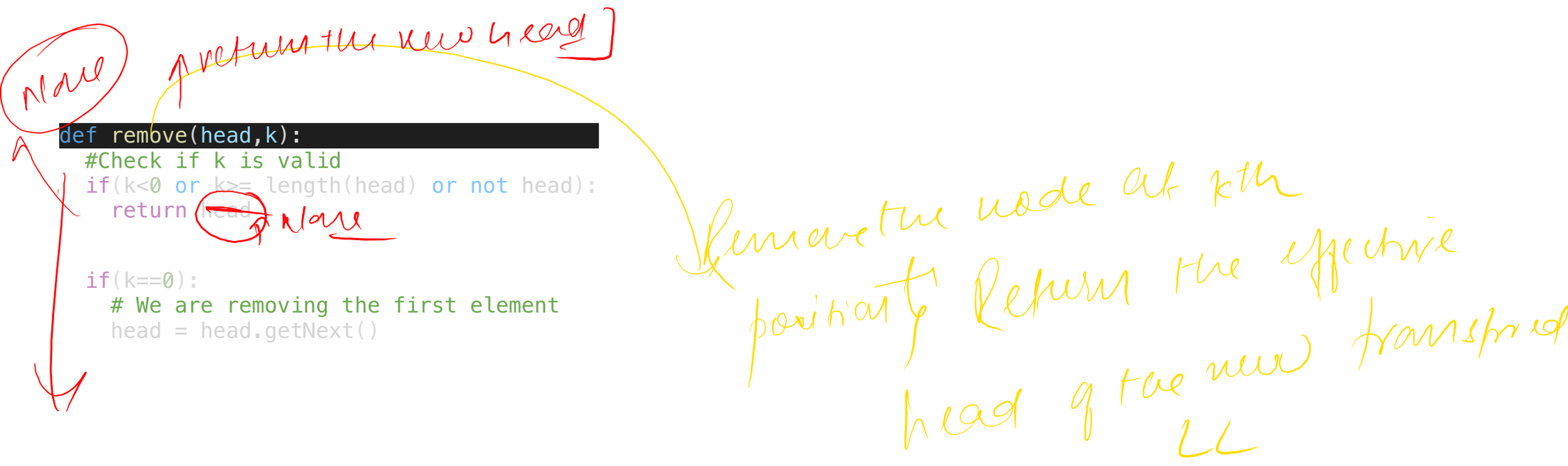
Remove that node from LL



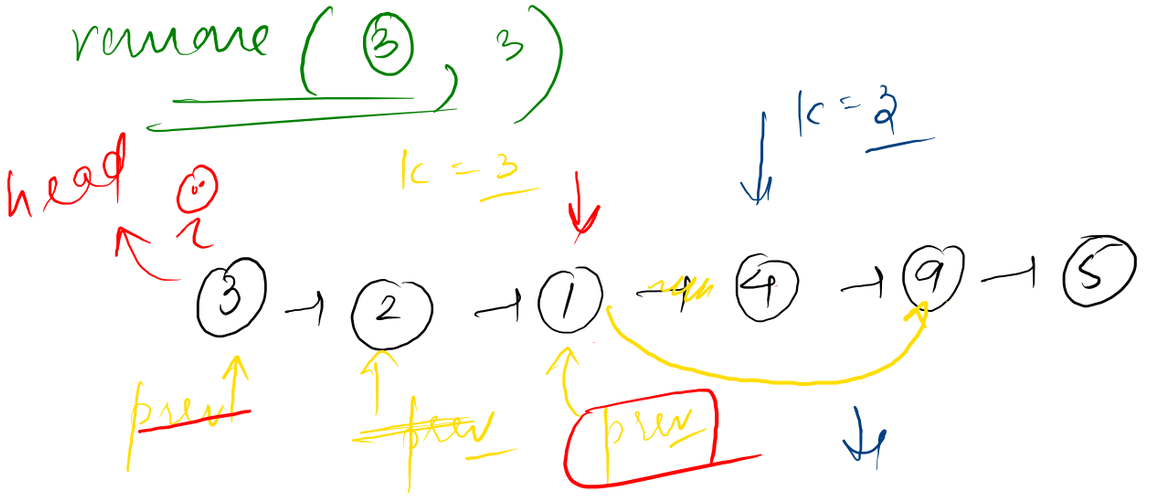


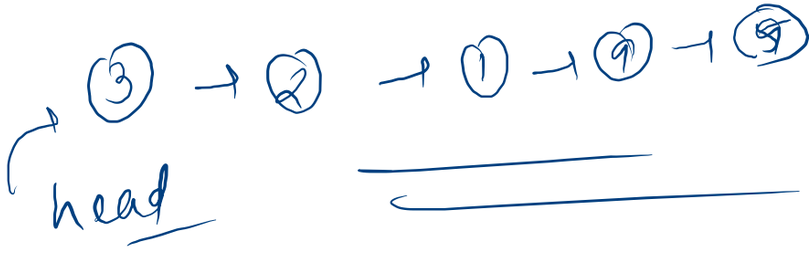
3 → 2 → 1 → 5 → 9

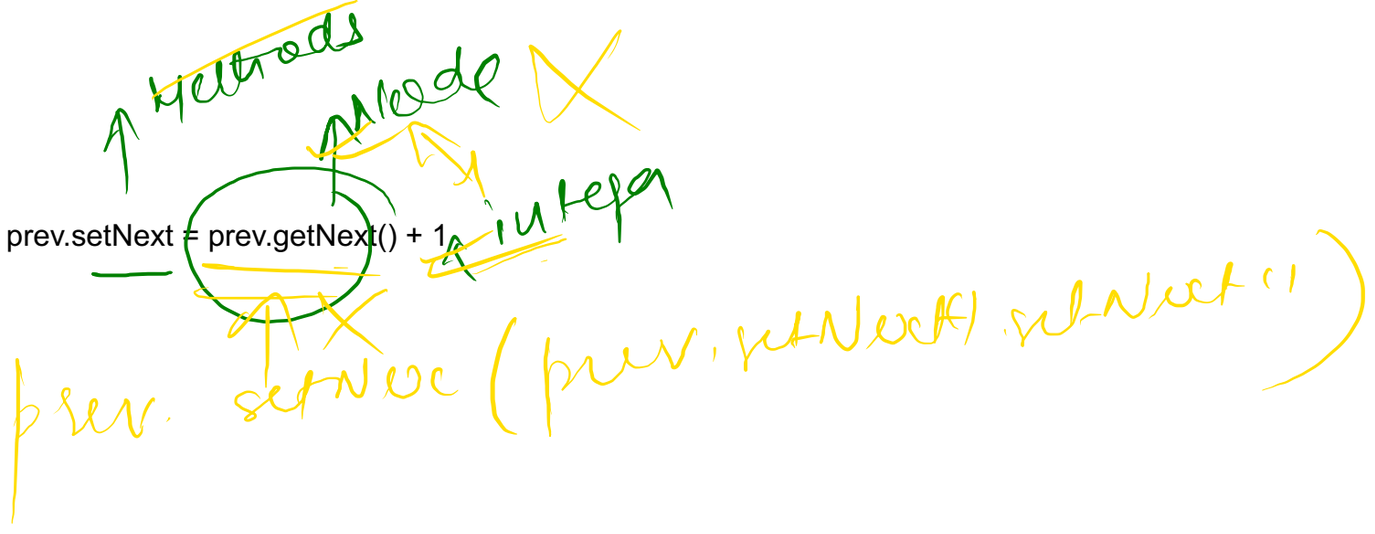
```
def rembve(head,k):
  #Check if k is valid
  if(k<0 or k>= length(head) or not head):
    return head

  if(k==0):
    # We are removing the first element
    head = head.getNext()
```



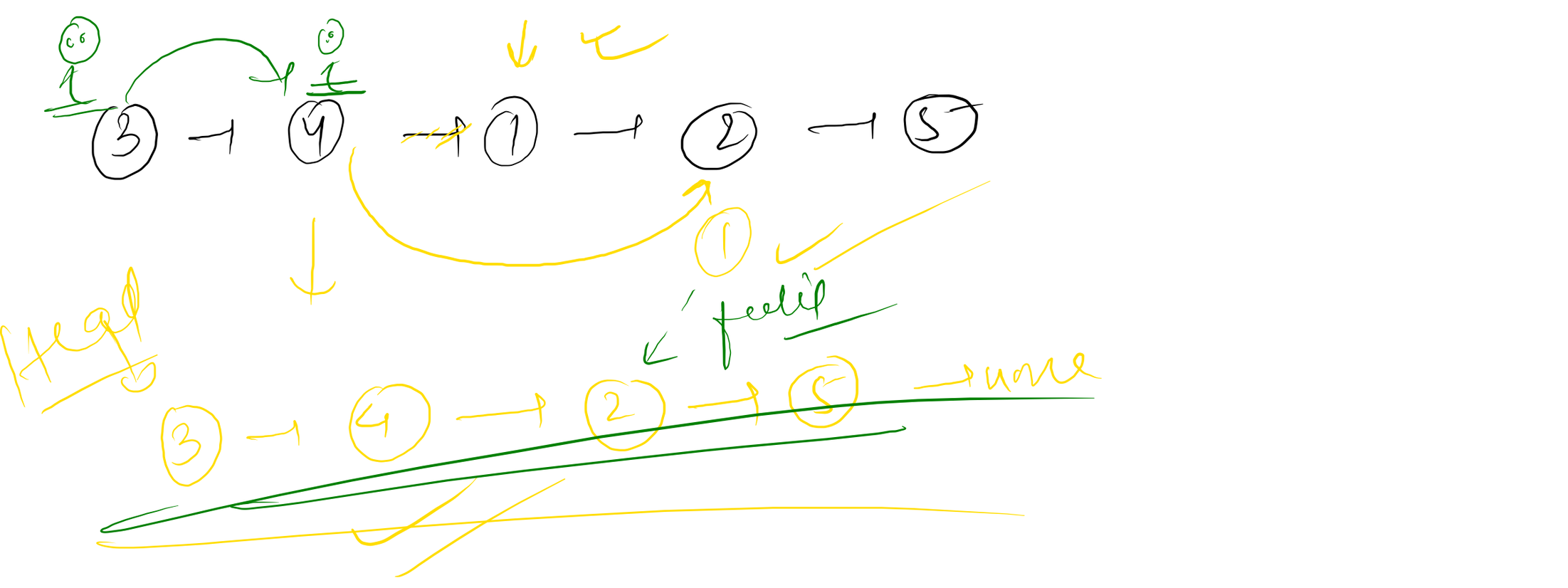
```
def remove(head,k):  #Supposed to return the new head of the transformed LL ( by removing kth
node)
  #Check if k is valid
  if(k<0 or k>= length(head) or not head):
    return head


  if(k==0):
    # We are removing the first element
    head = head.getNext()
  else :


    # We jump to k-1th position
    i=0
    prev = head
    while(i< k-1):
      prev = prev.getNext()
      i +=1
    #prev will be pointing to the node left of the kth position
    prev.setNext(prev.getNext().getNext())

  return head
```



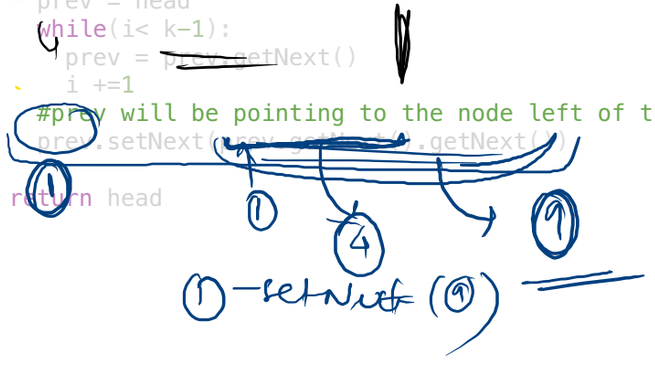


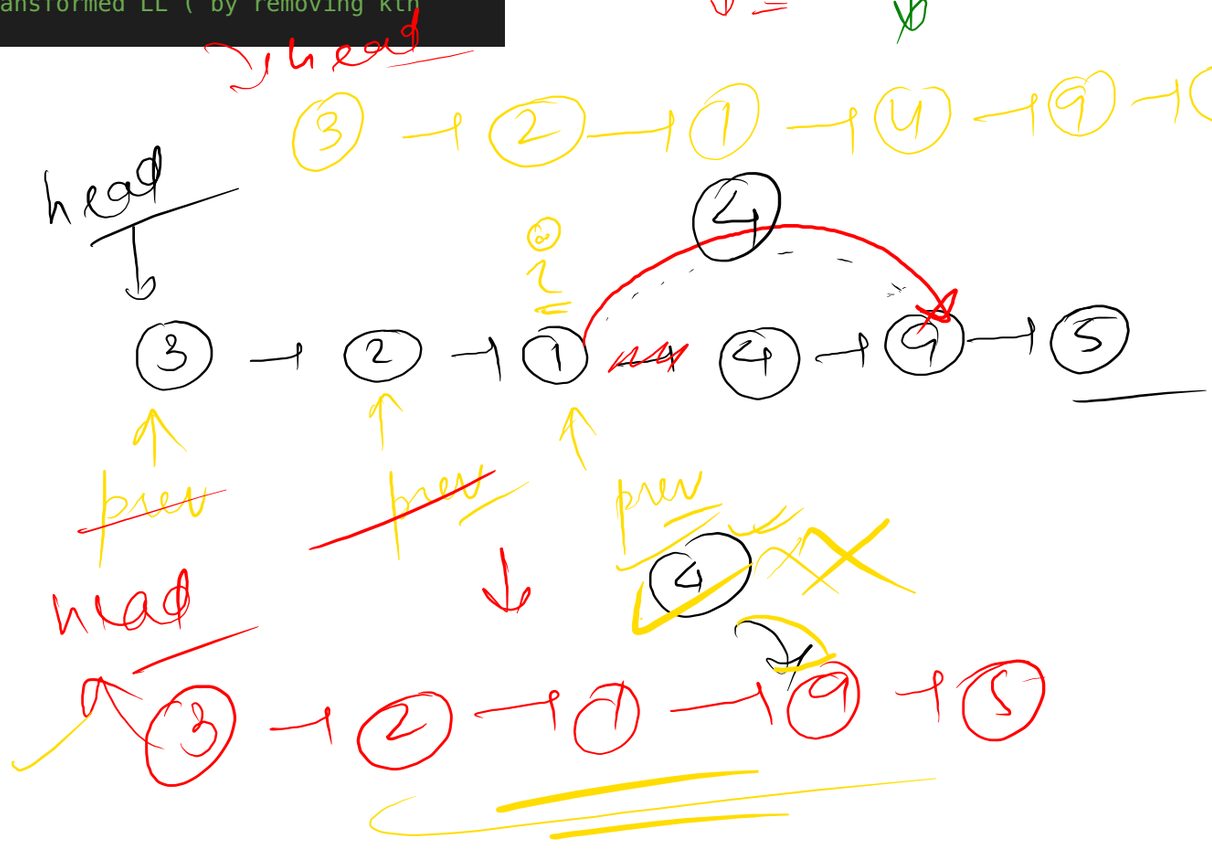

prev.setNext = prev.getNext() + 1;

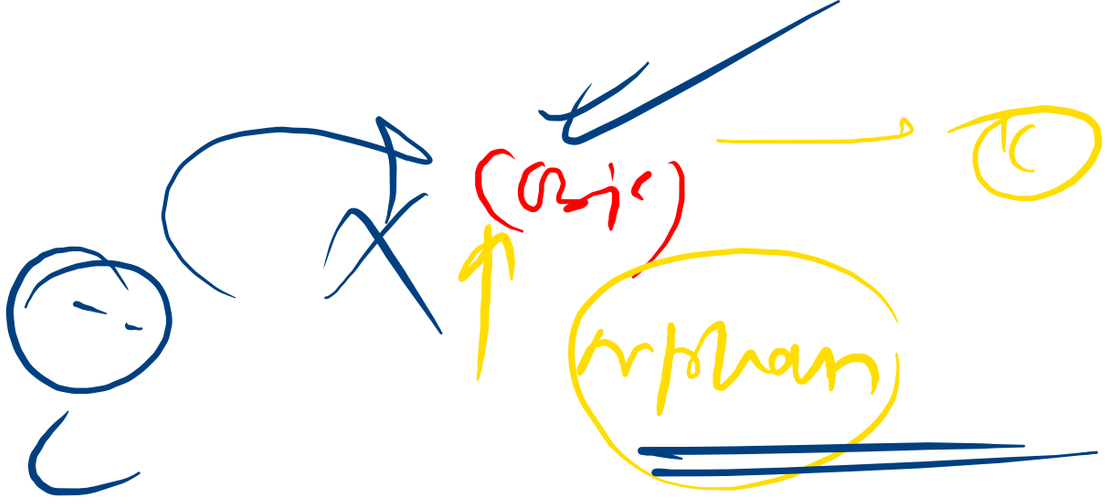
```
prev.setNext(prev.getNext().getNext())
```

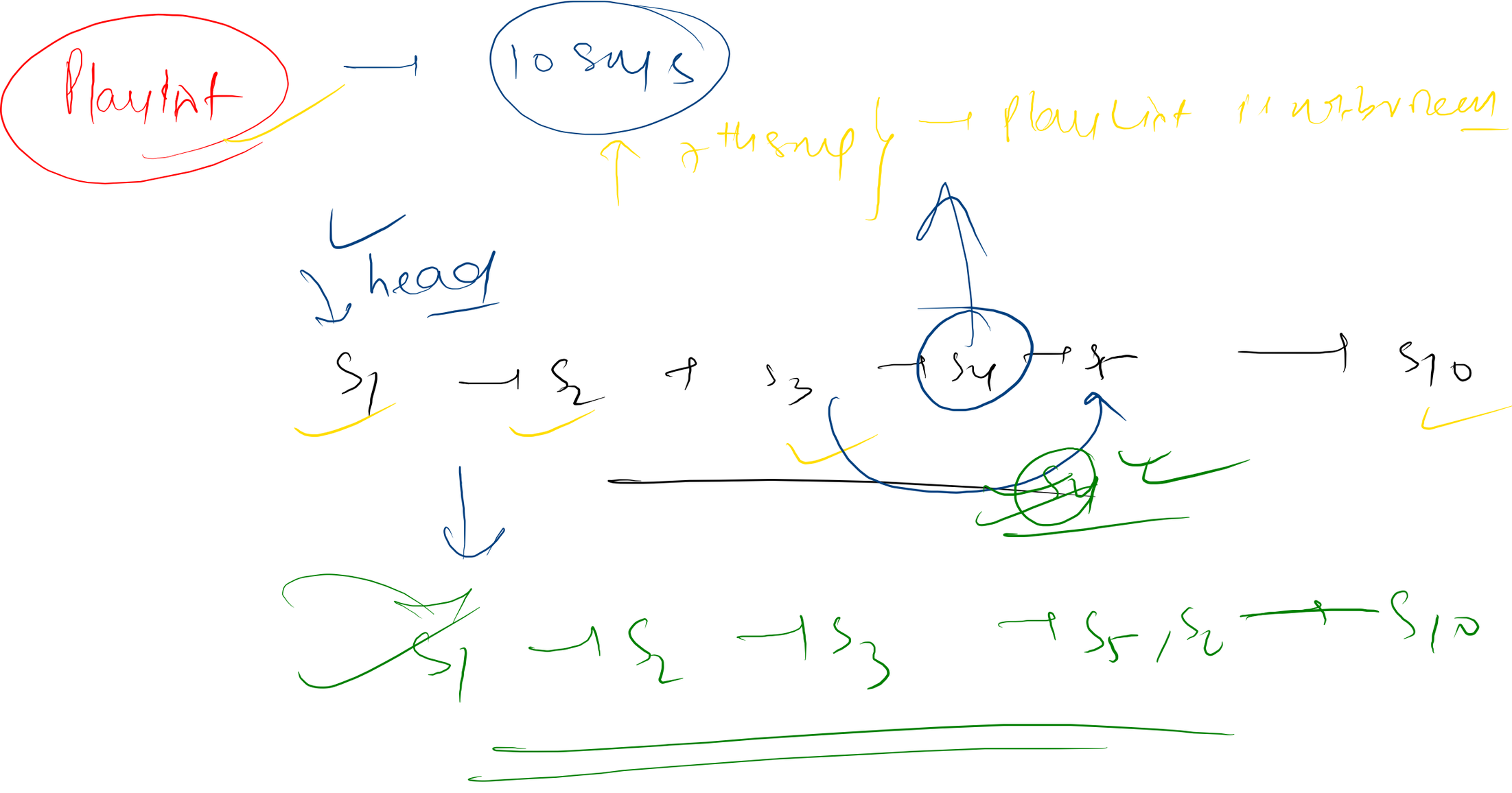

3 → 4 → 1 → 2 → 5
1
feelif
3 → 4 → 2 → 5 → none

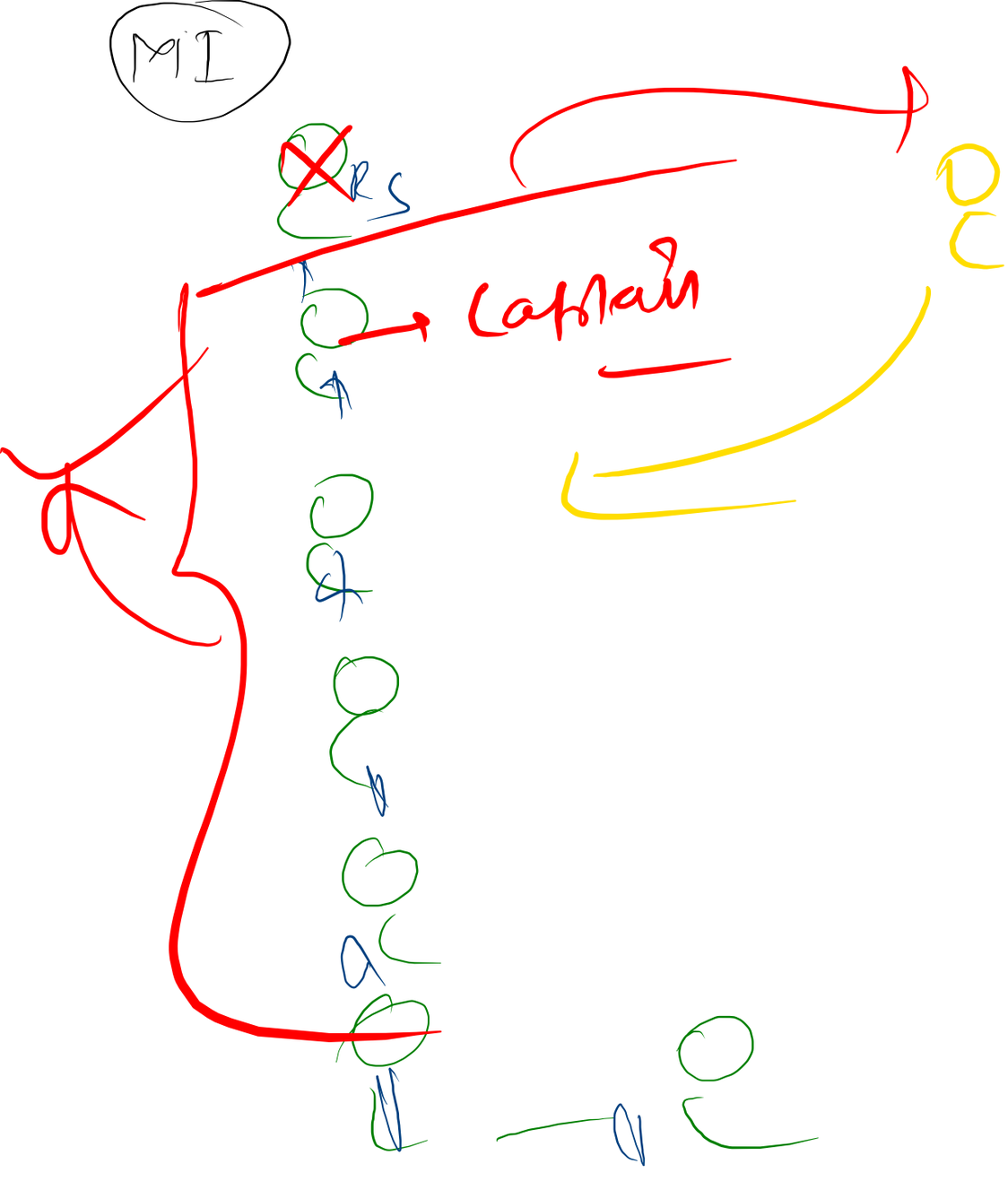
```
def remove(head,k):  #Supposed to return the new head of the transformed LL ( by removing kth node)
  #Check if k is valid
  if(k<0 or k>= length(head) or not head):
    return head


  if(k==0):
    # We are removing the first element
    head = head.getNext()
  else :

    # We jump to k-1th position
    i=0
    prev = head
    while(i< k-1):
      prev = prev.getNext()
      i +=1
    #prev will be pointing to the node left of the kth position
    prev.setNext(prev.getNext().getNext())

  return head
```

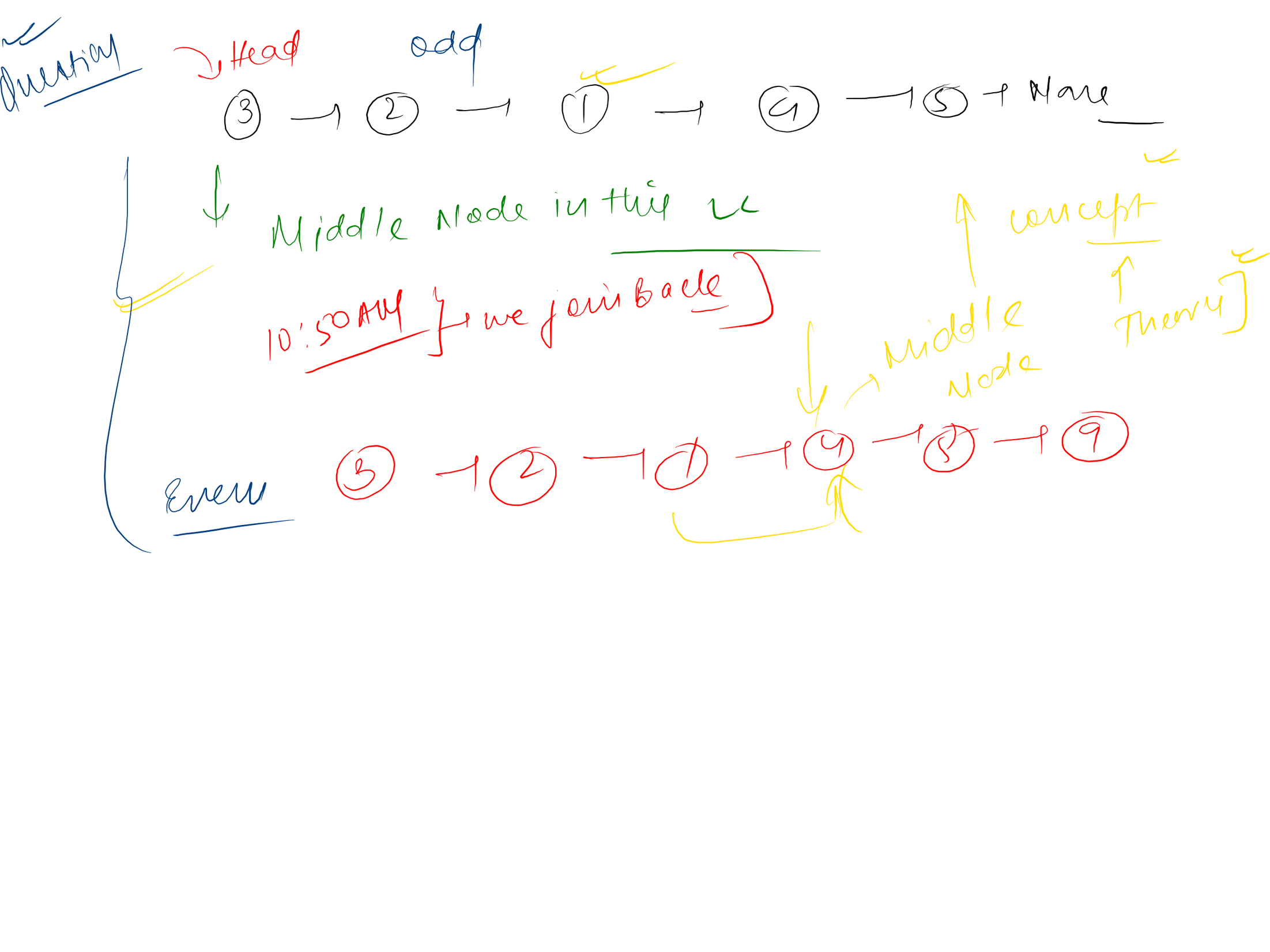


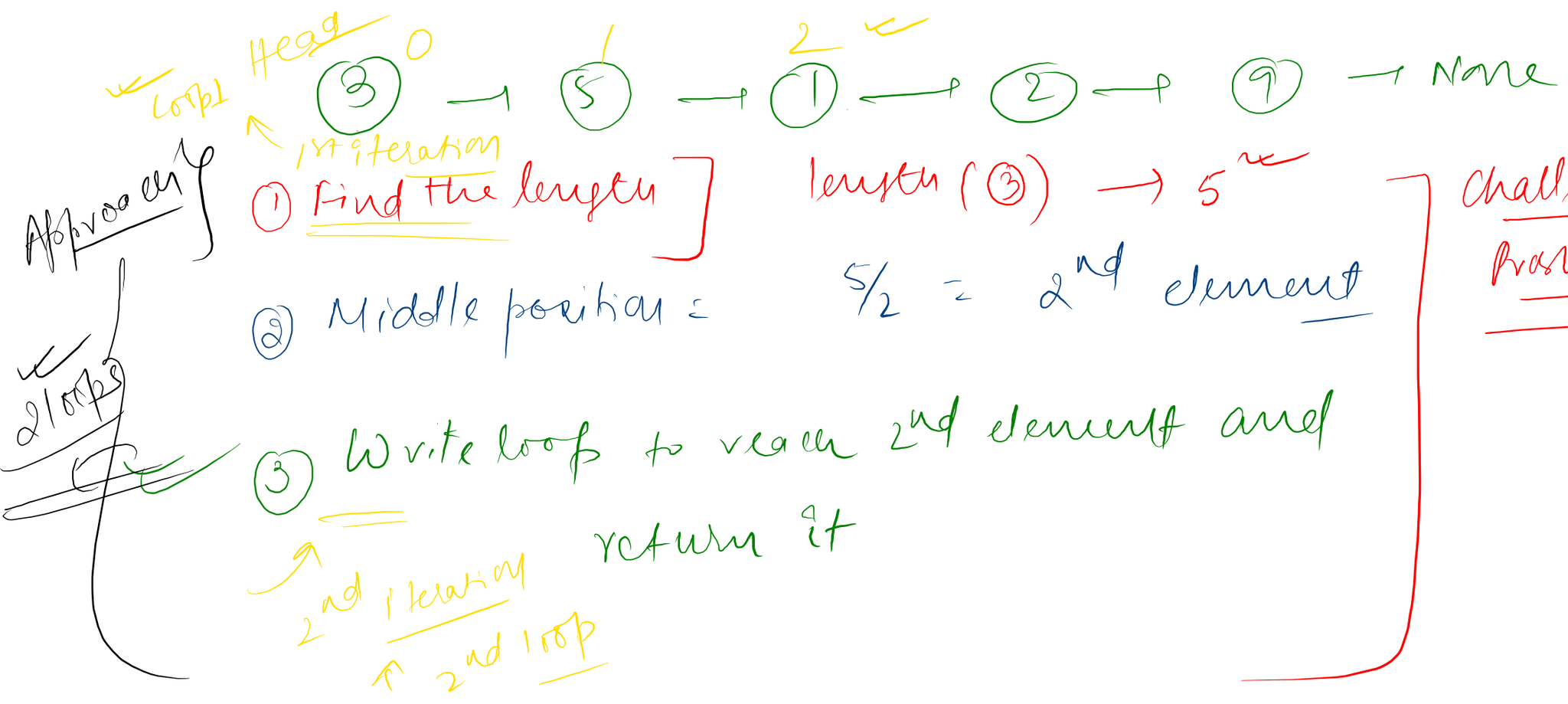





Playlist
10 songs
3 threads
Playlist is interleaved
head
s_1
s_2
s_3
s_4
s_{10}
s_1
s_2
s_3
s_5 / s_2
s_{10}

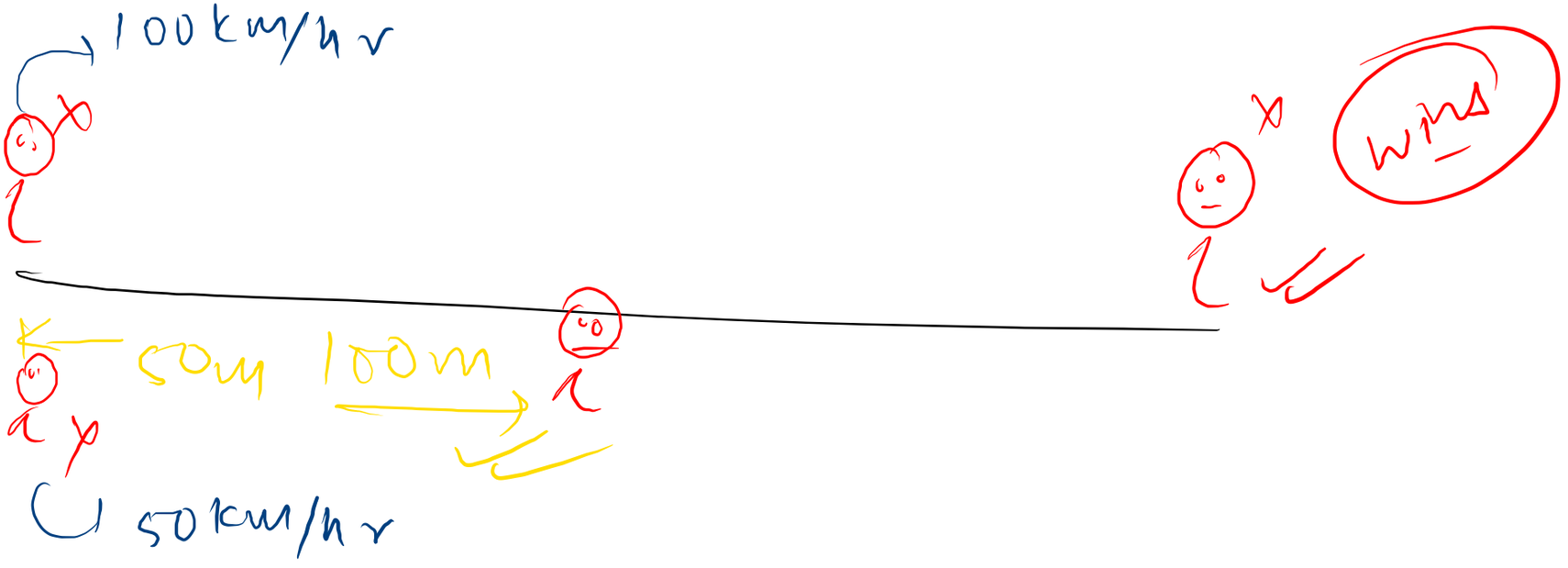

MI
RS
Capital


Question
Head
odd
3 → 2 → 1 → 4 → 5 → Null
Middle Node in this
10:50 AM } we join back
concept
Theory
Middle Node
Even
3 → 2 → 1 → 4 → 5 → 9

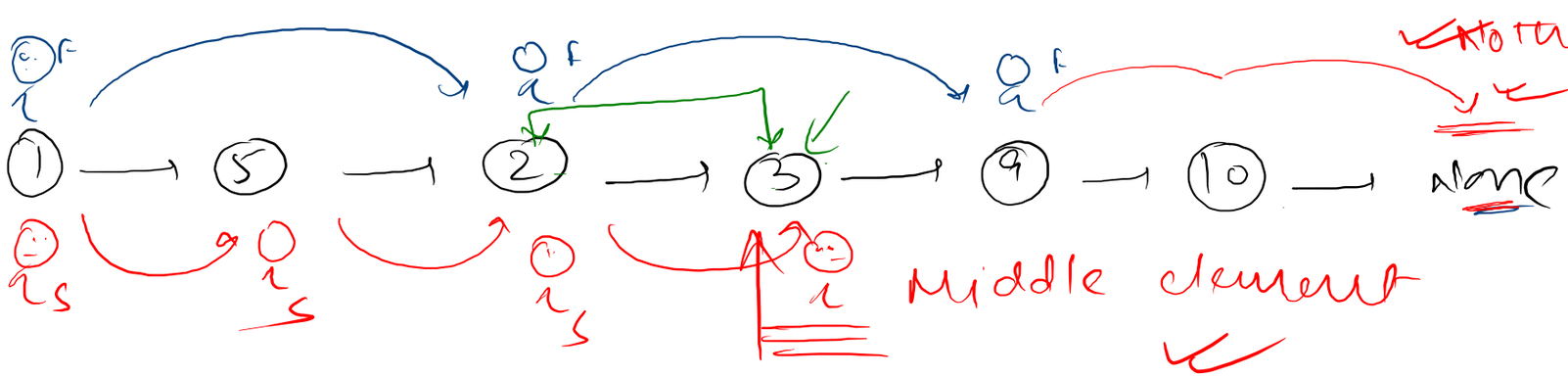

Head
0
2
3
5
1
2
9
None
Loop1
1st iteration
Approach
2 loops
1 Find the length
length (3) → 5
2 Middle position = 5/2 = 2nd element
3 Write loop to reach 2nd element and return it
2nd iteration
2nd loop
Challenge Problem

optimized solution → Run only 1 loop → single iteration

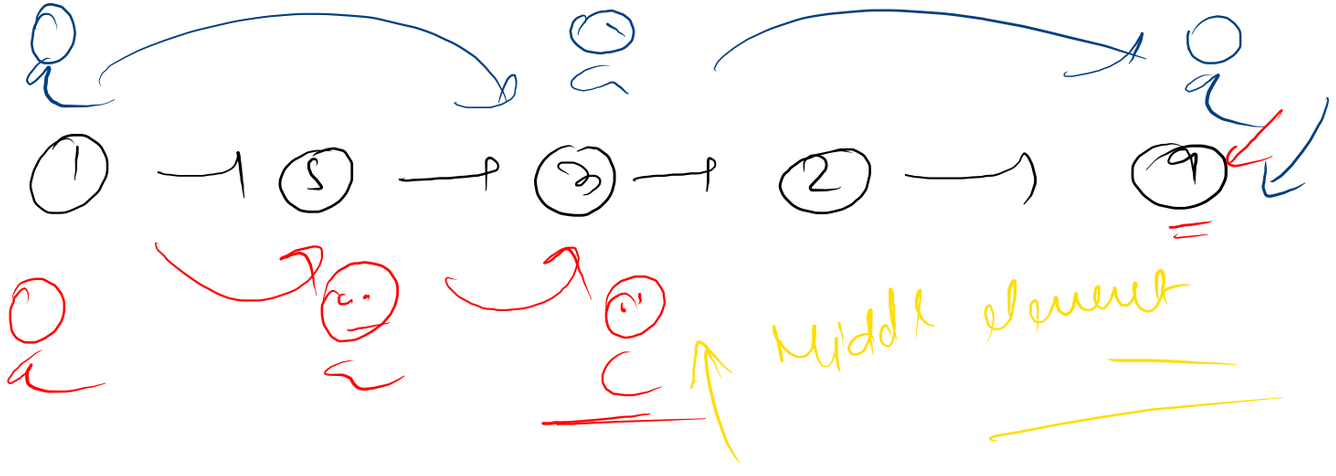
2 pointer approach


100 km/hr
wind
50m
100m
50 km/hr

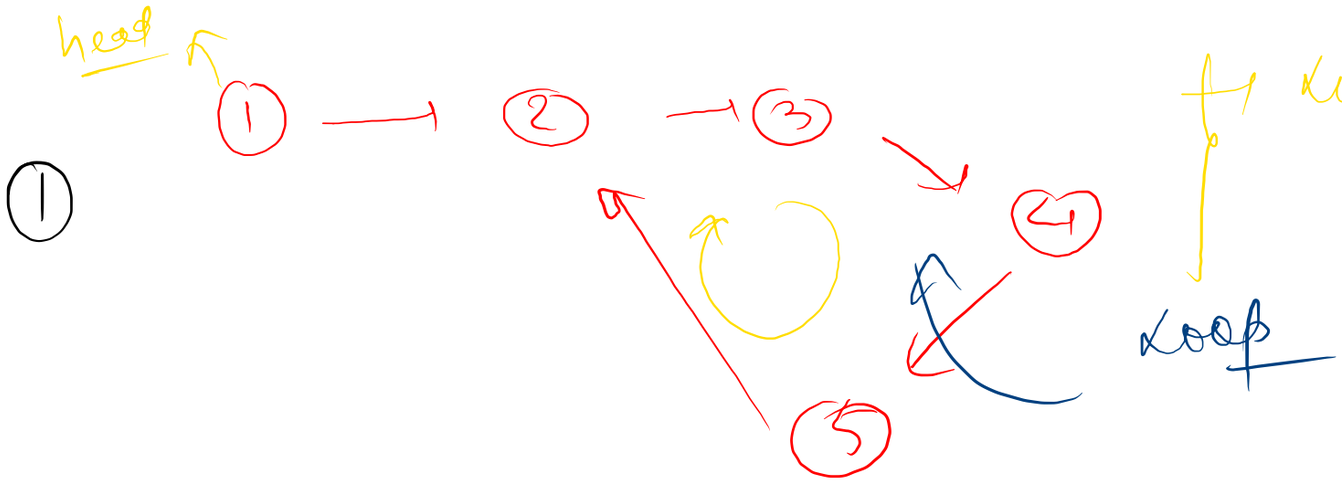
Even case



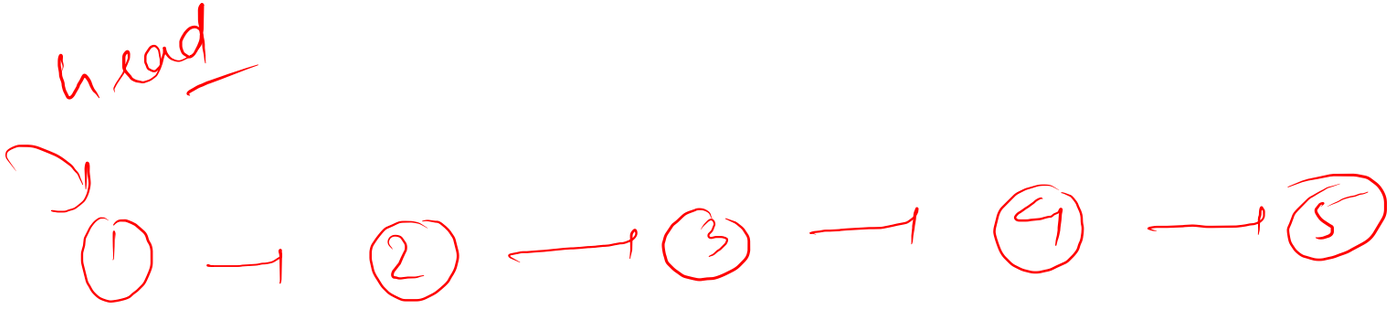
Odd case



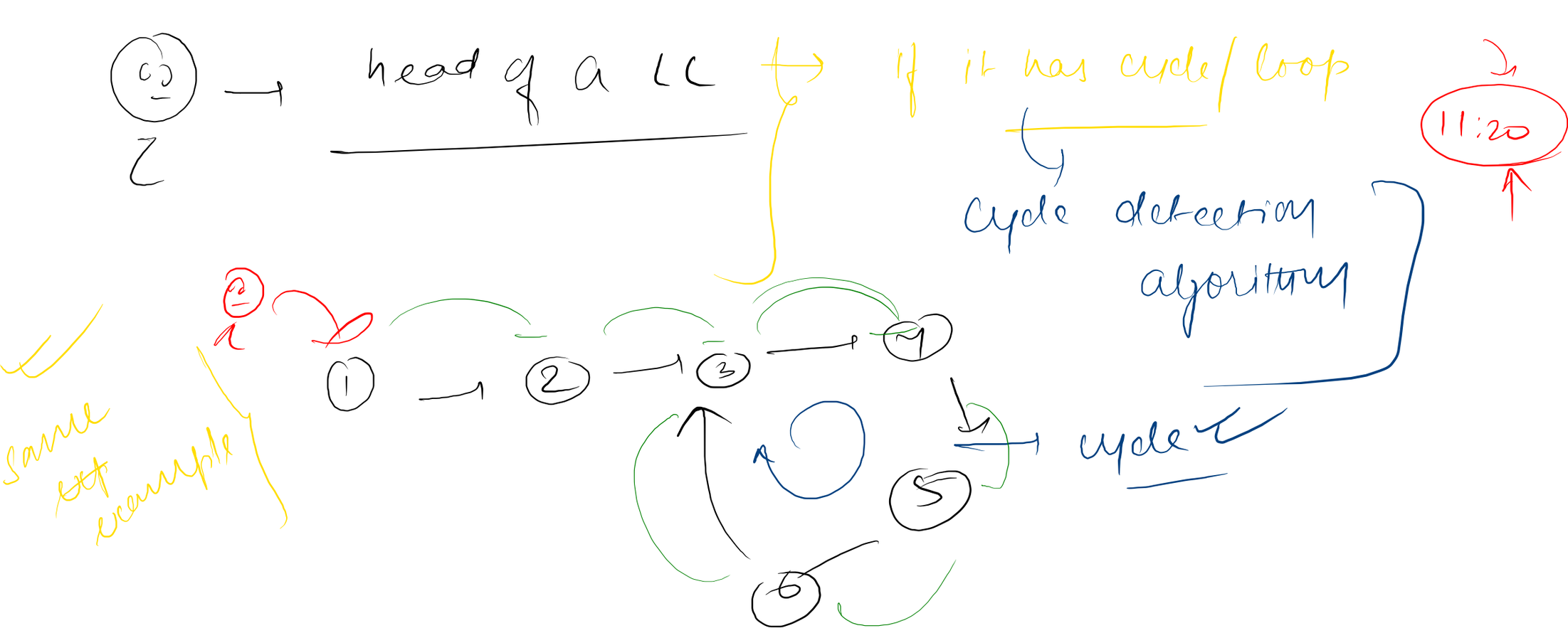
Problem } (5) I will ask this question

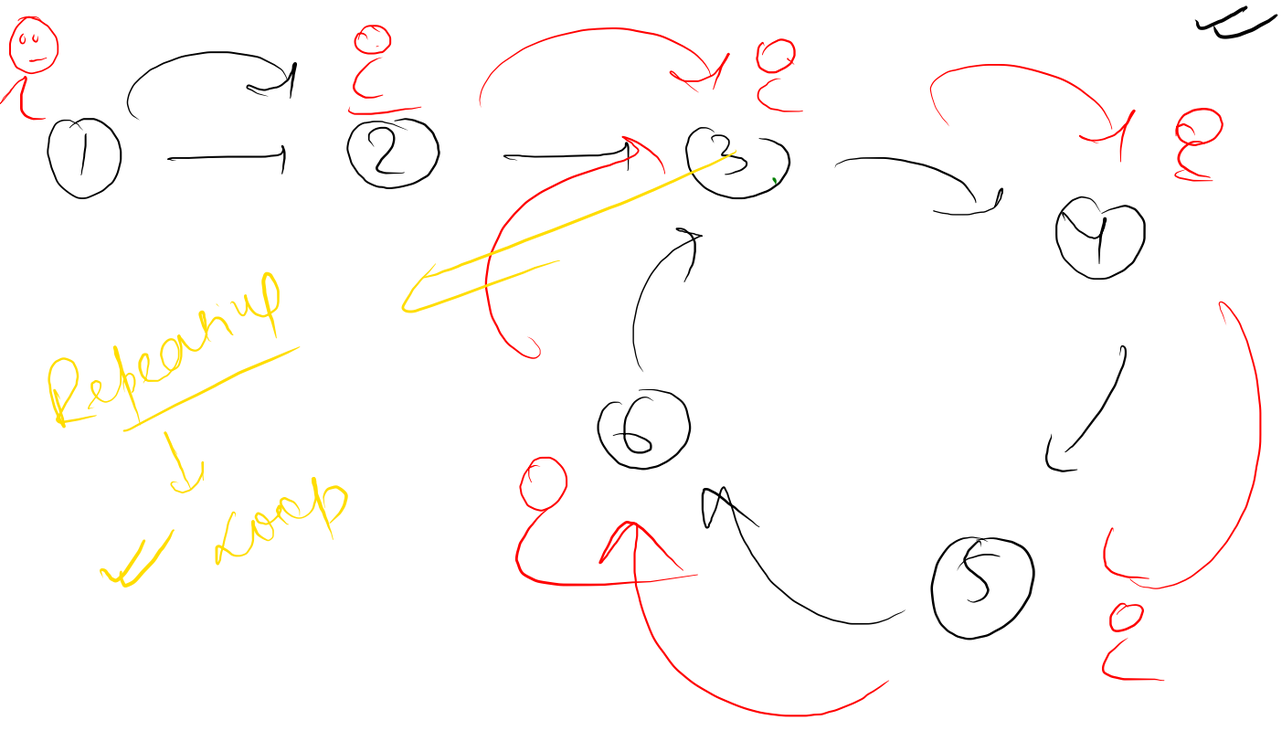


① ↑ Linked list contains a loop [cycle]



② ↑ Doesn't have a loop


head of a LL
if it has cycle/loop
11:20
Cycle detection algorithm
same set example
cycle



Approach 1

Referring

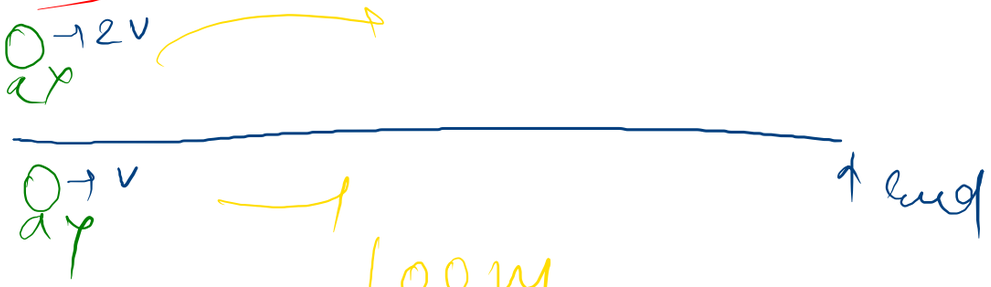
Loop

Variable storage DS

set

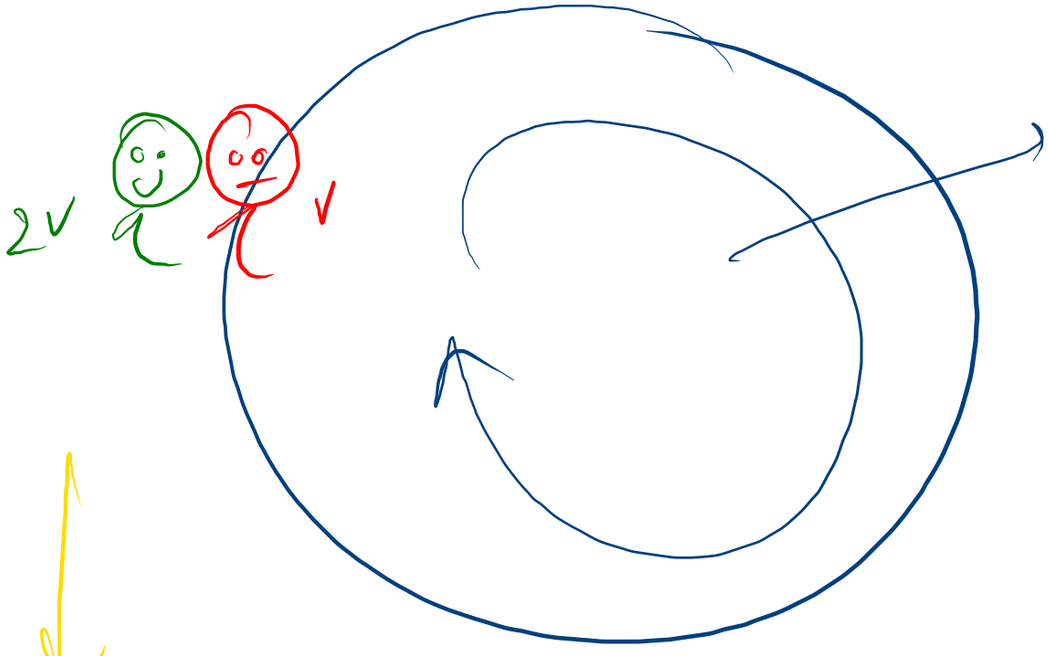

Address
Object
Loop
Points
TC = O(n)
Challenge
Extra Space / Storage
Set

Space Optimized solution



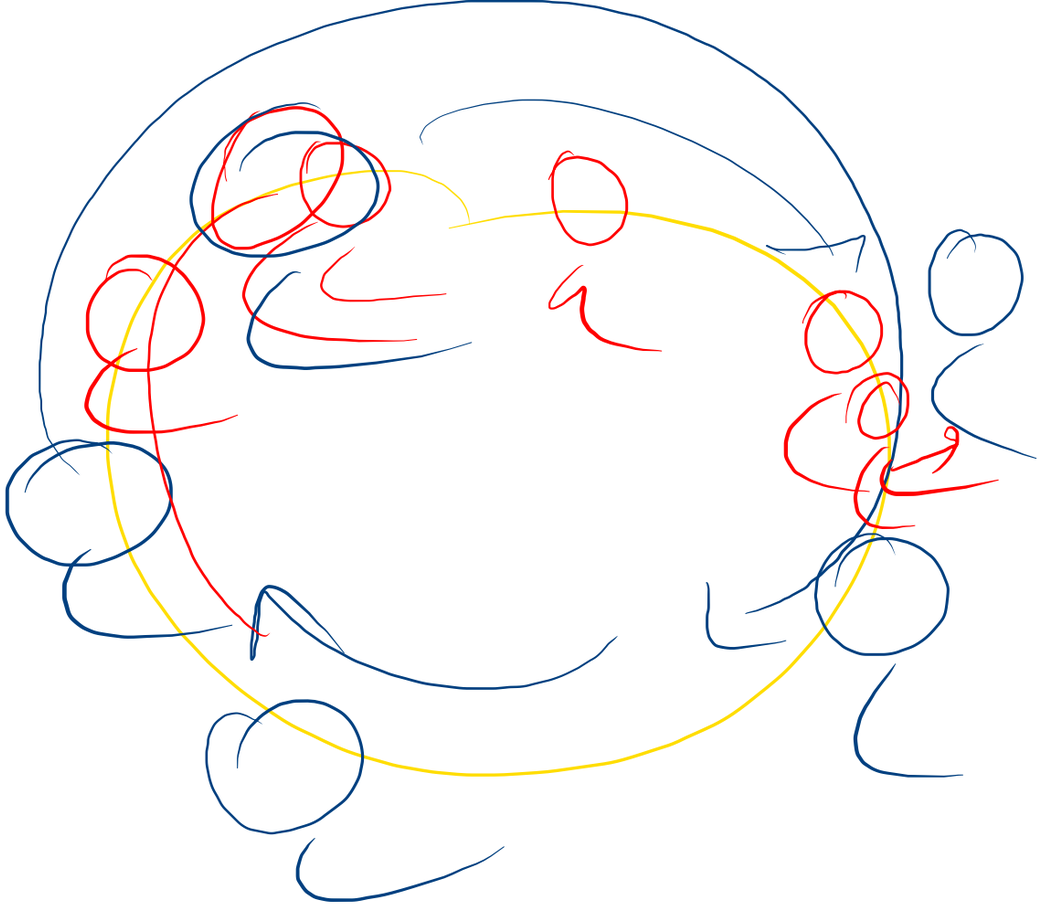
100 m

will X and Y ever meet ? during race



indefinite time

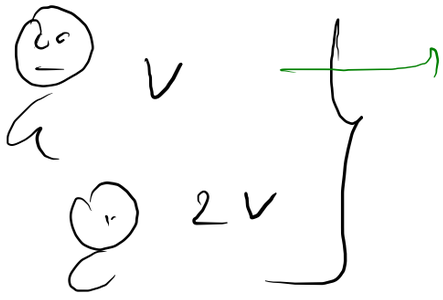
will they meet again ? while running



Fast overtakes slow

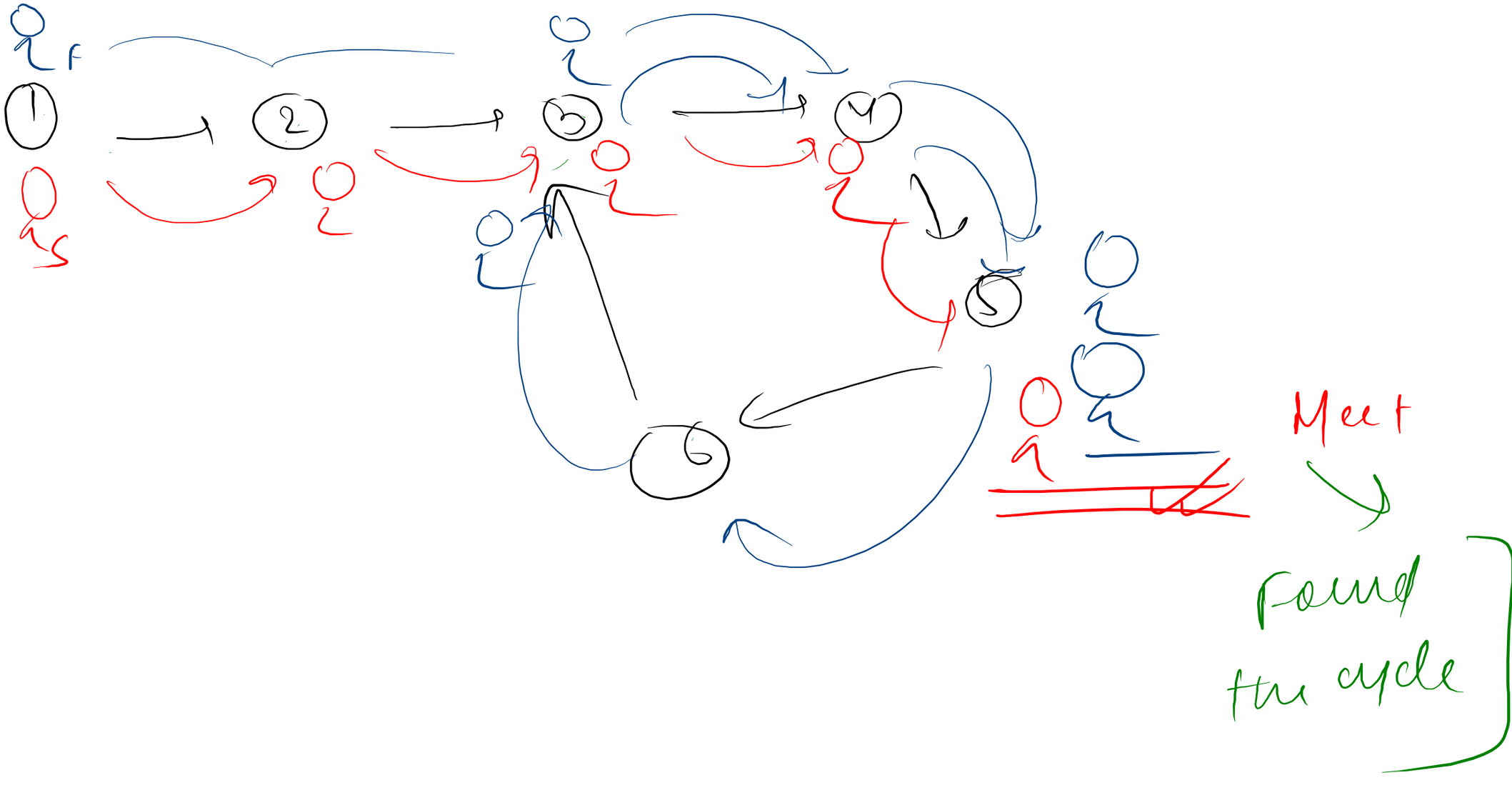
↓

Meet

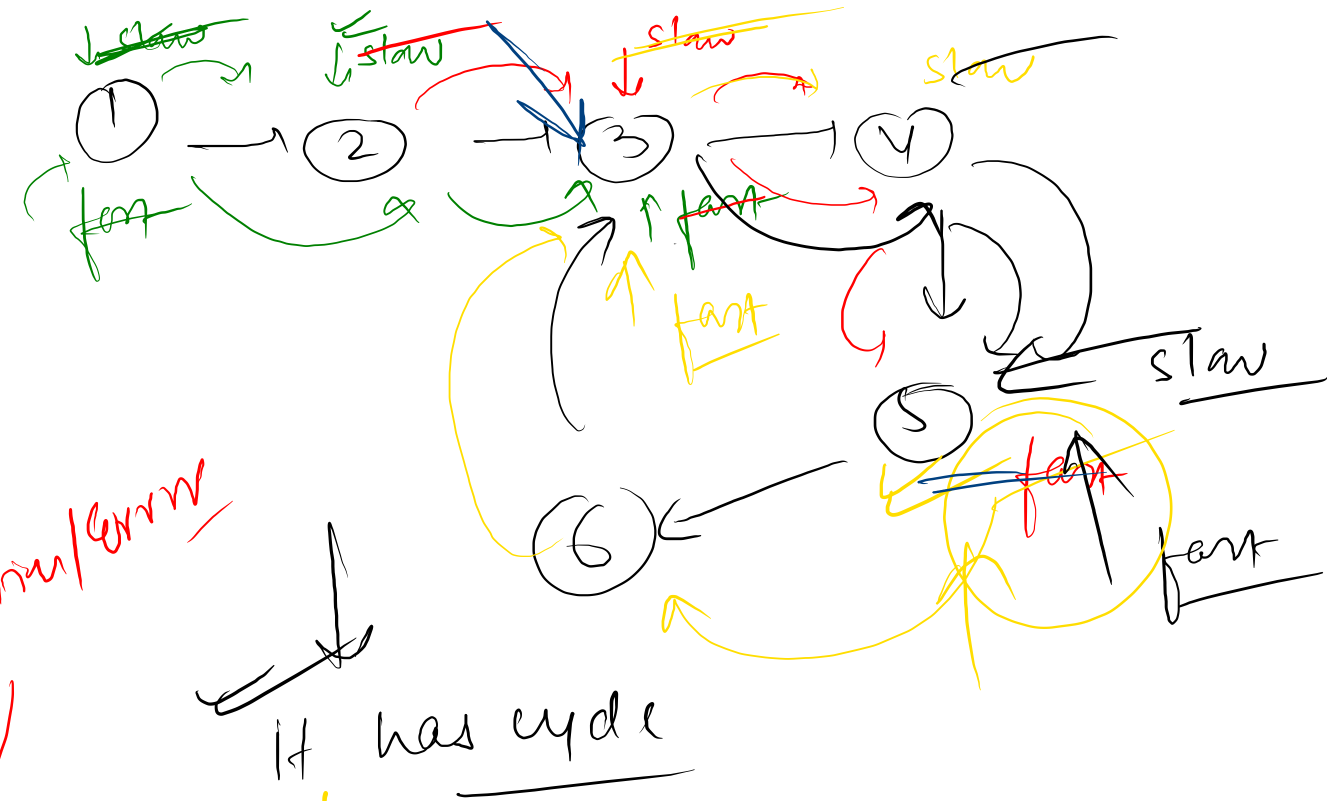


Run → meet → a cycle exists

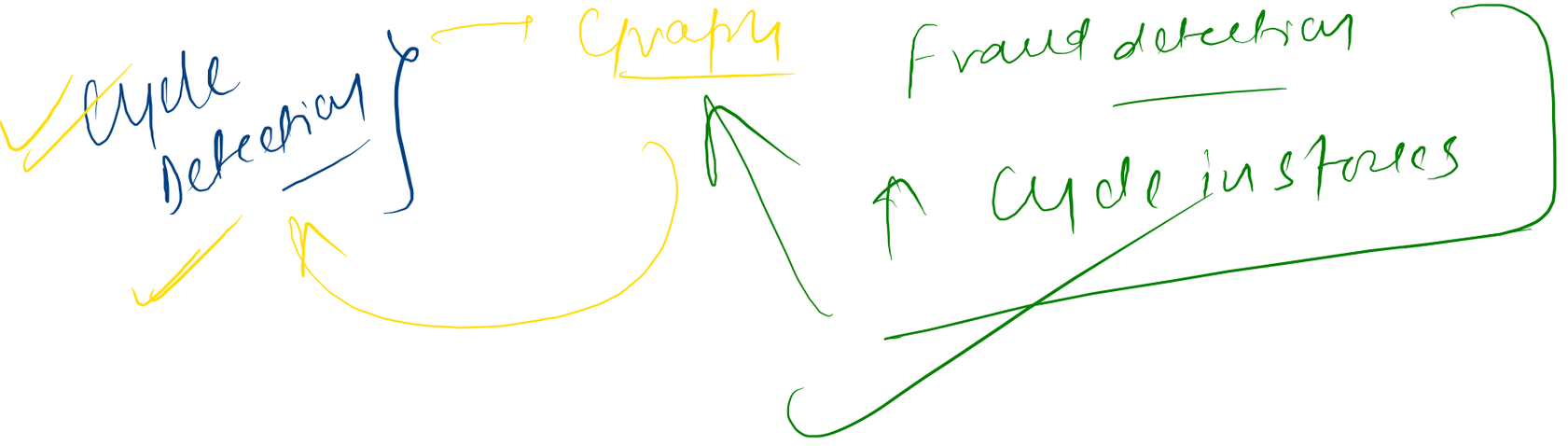
X Doesn't meet → a cycle doesn't exist

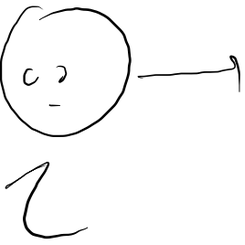

f
s
1
2
3
4
5
6
Meet
found the cycle

```python
def isCyclePresent(head):
    slow = head
    fast = head

    while(fast and fast.getNext()):
        slow = slow.getNext()
        fast = fast.getNext().getNext()

        if fast and slow.getData() == fast.getData():
            return True #cycle exists

    return False #Cycle doesn't exist
```



None · no cycle · exception/error · None · Handle duplicate · Assignments

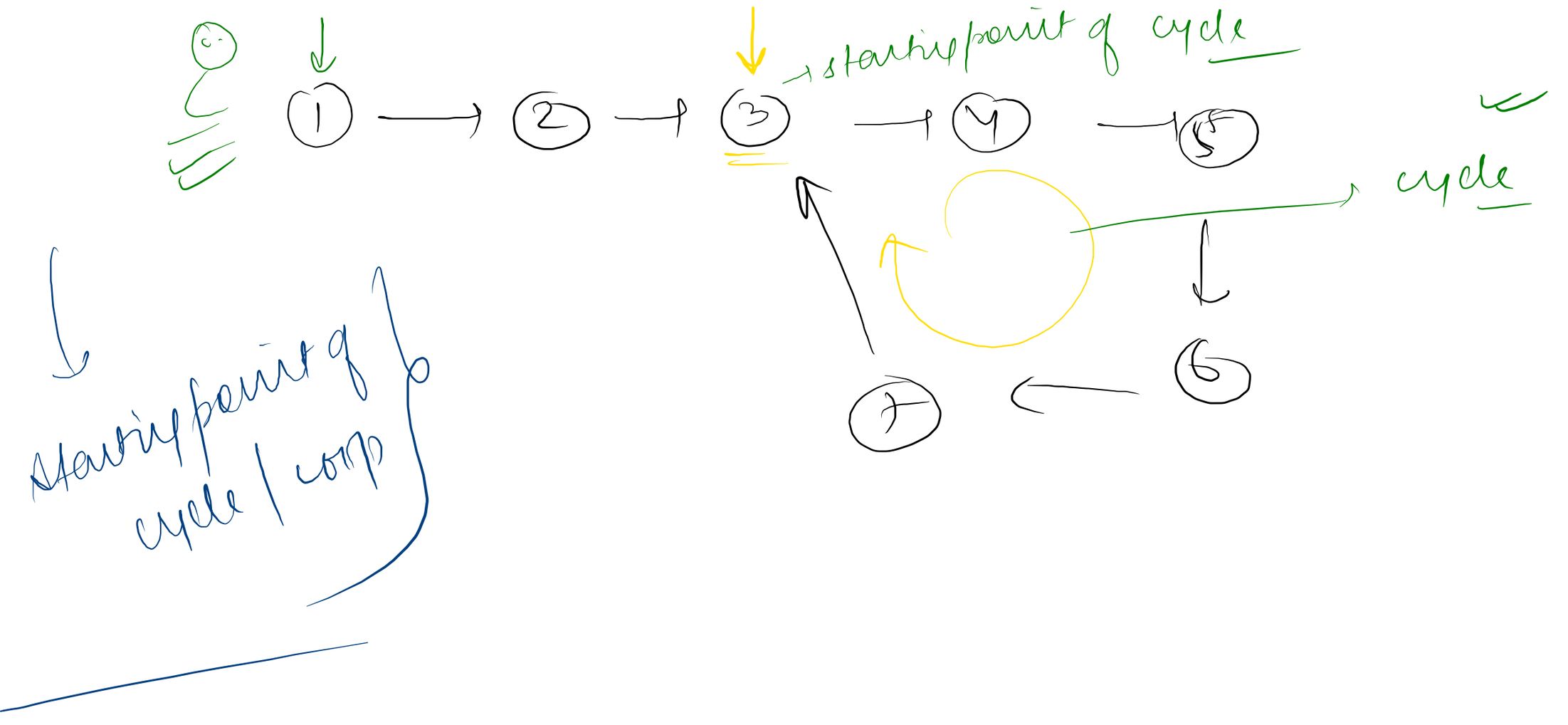

Cycle detection
Graphs
Fraud detection
Cycle in stores

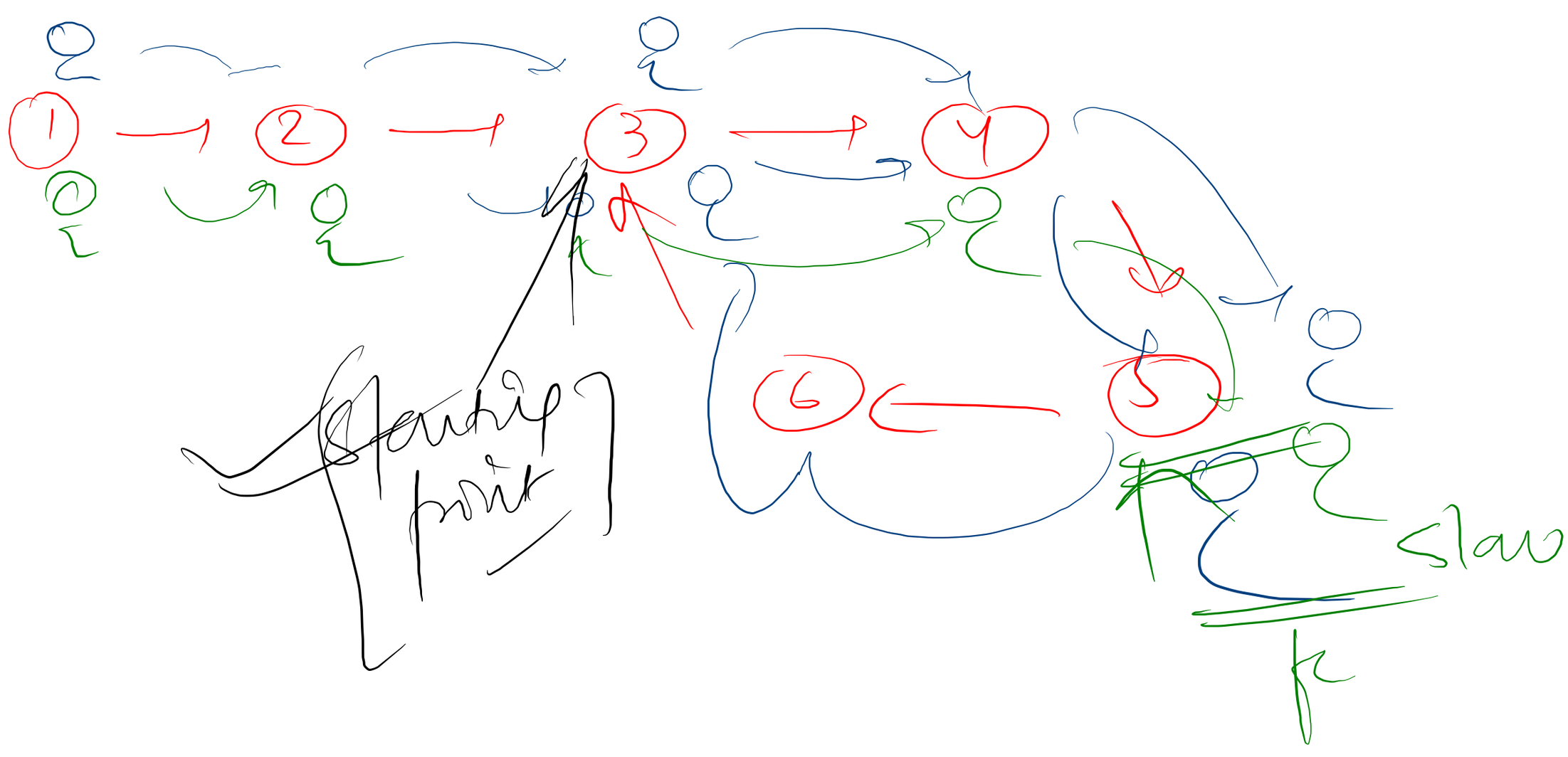


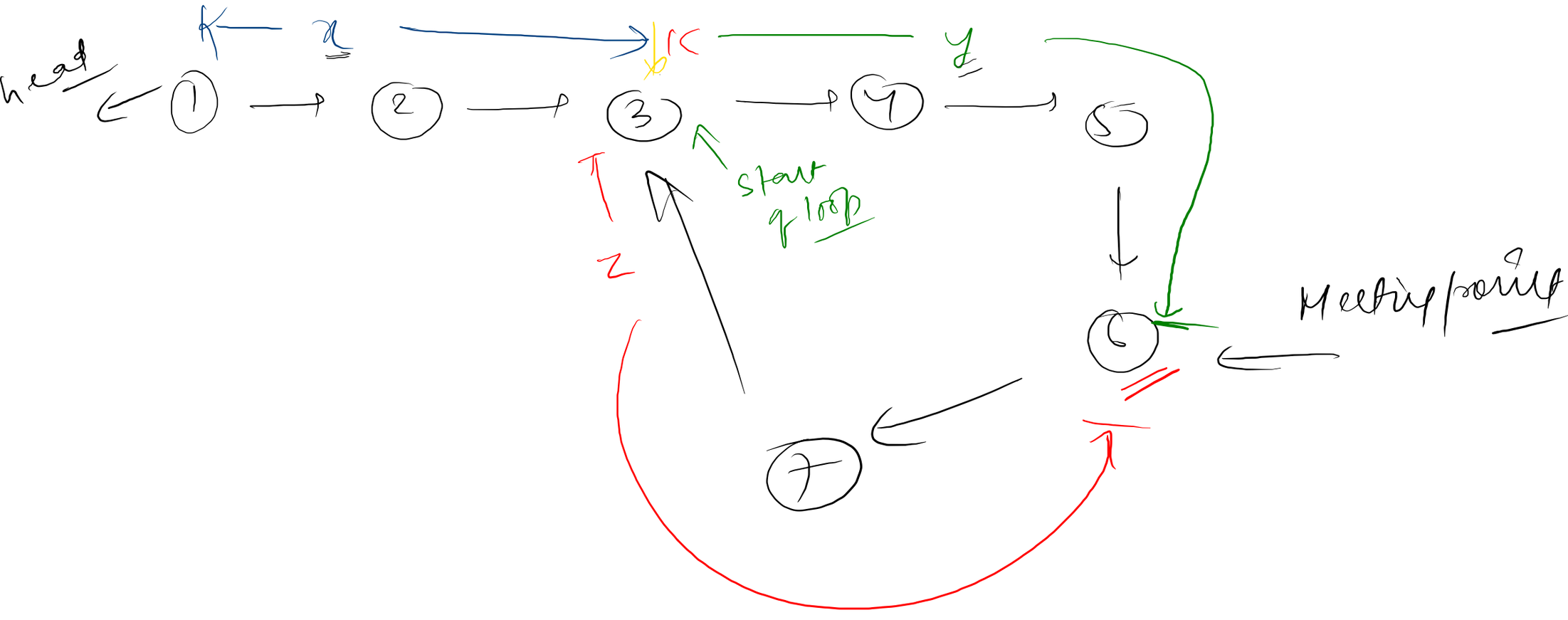
Find the middle

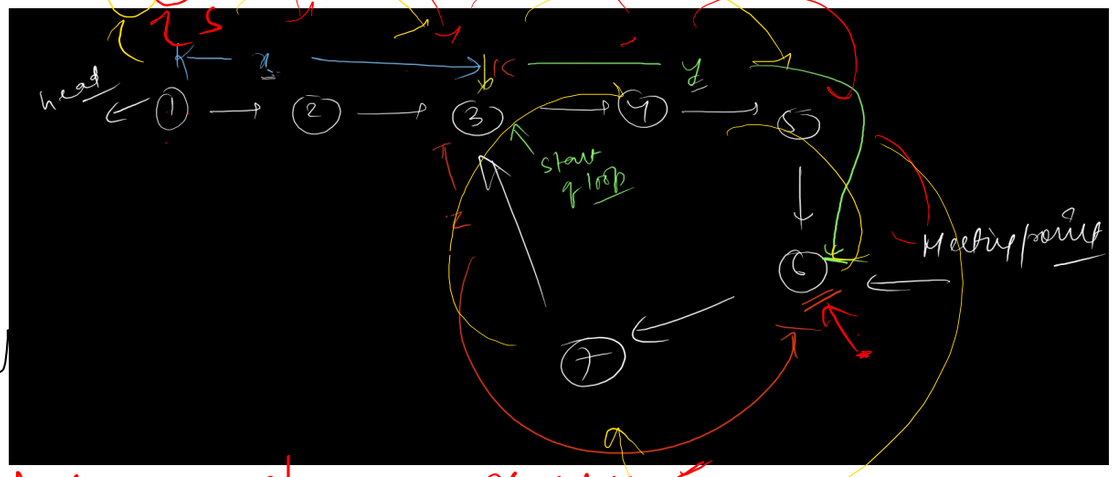
↳ check if the loop is there

↓


starting point of cycle
1
2
3
4
5
6
7
cycle
starting point of cycle / loop


1
2
3
4
5
6
Starting point
Slow


head
1
2
3
4
5
6
7
x
y
z
start of loop
Meeting point



x → distance from start → start of loop

y → distance from start of loop → Meeting point

z → distance from Meeting point → start of loop

$speed_{fast} = 2 \times speed_{slow}$

$D_{fast} = 2 \times D_{slow}$

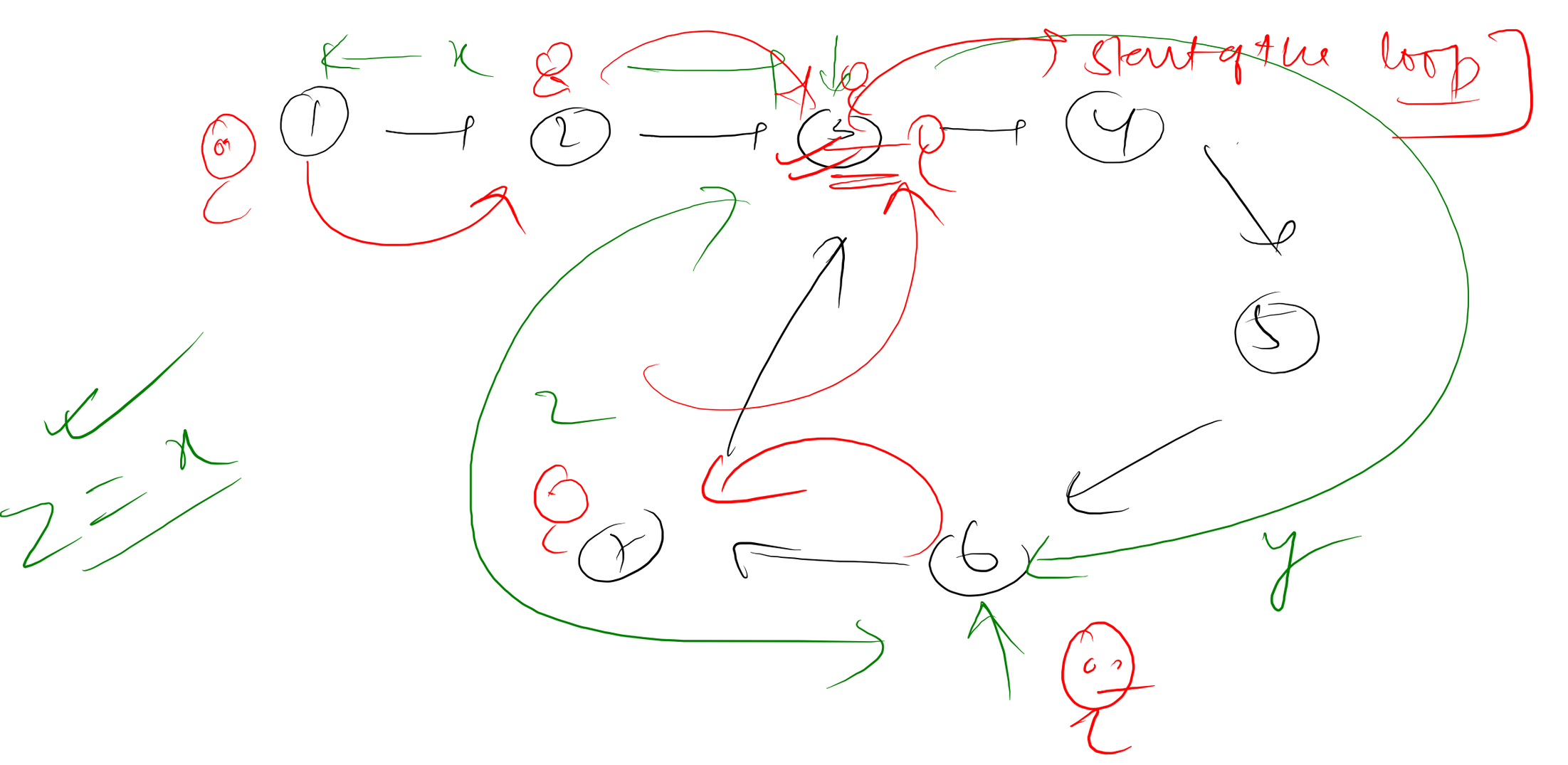
$\Rightarrow x + z + 2y = 2(x + y)$

$\Rightarrow x + z + 2y = 2x + 2y$

$\Rightarrow x = z$ ← proved

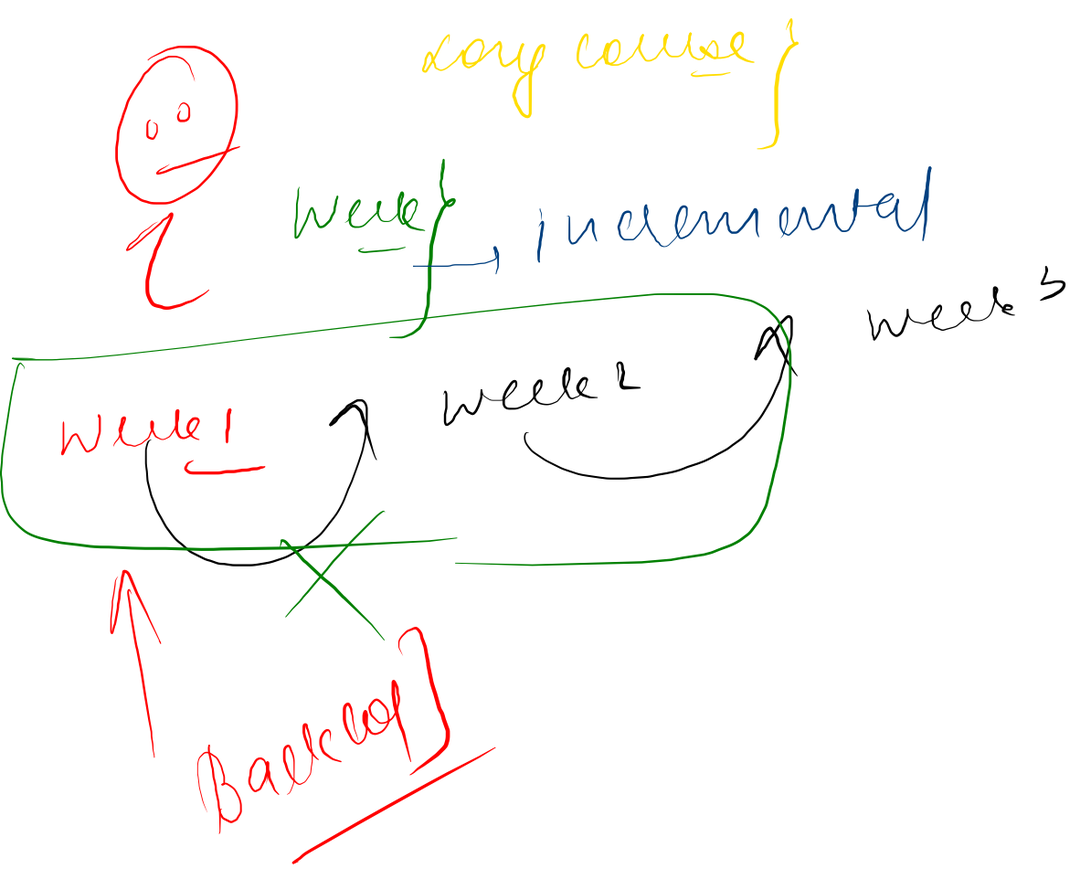
Distance slow = $x + y$

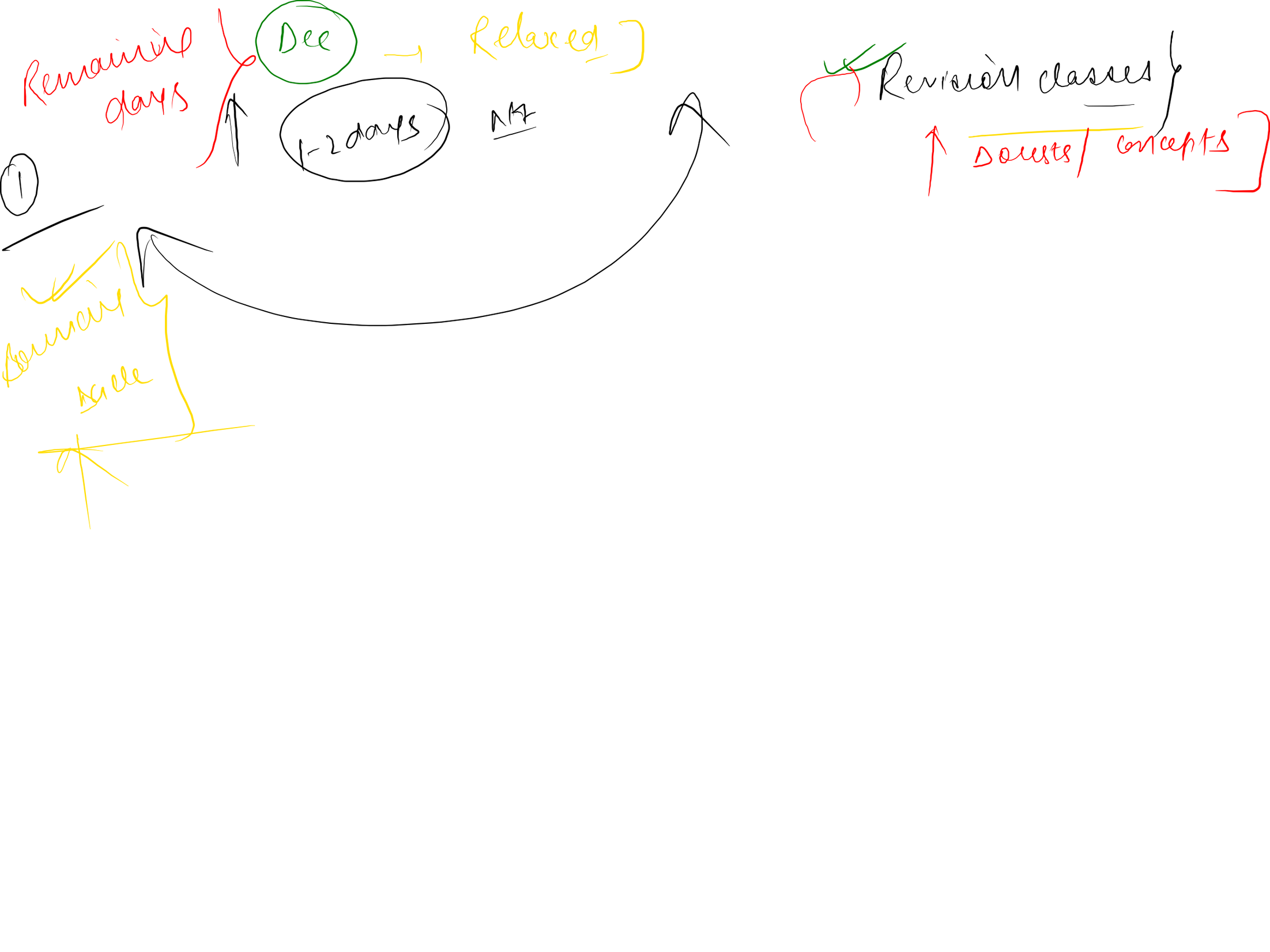
Distance fast = $x + y + z + y = x + z + 2y$


x
start of the loop
z = x
z
y

Break time

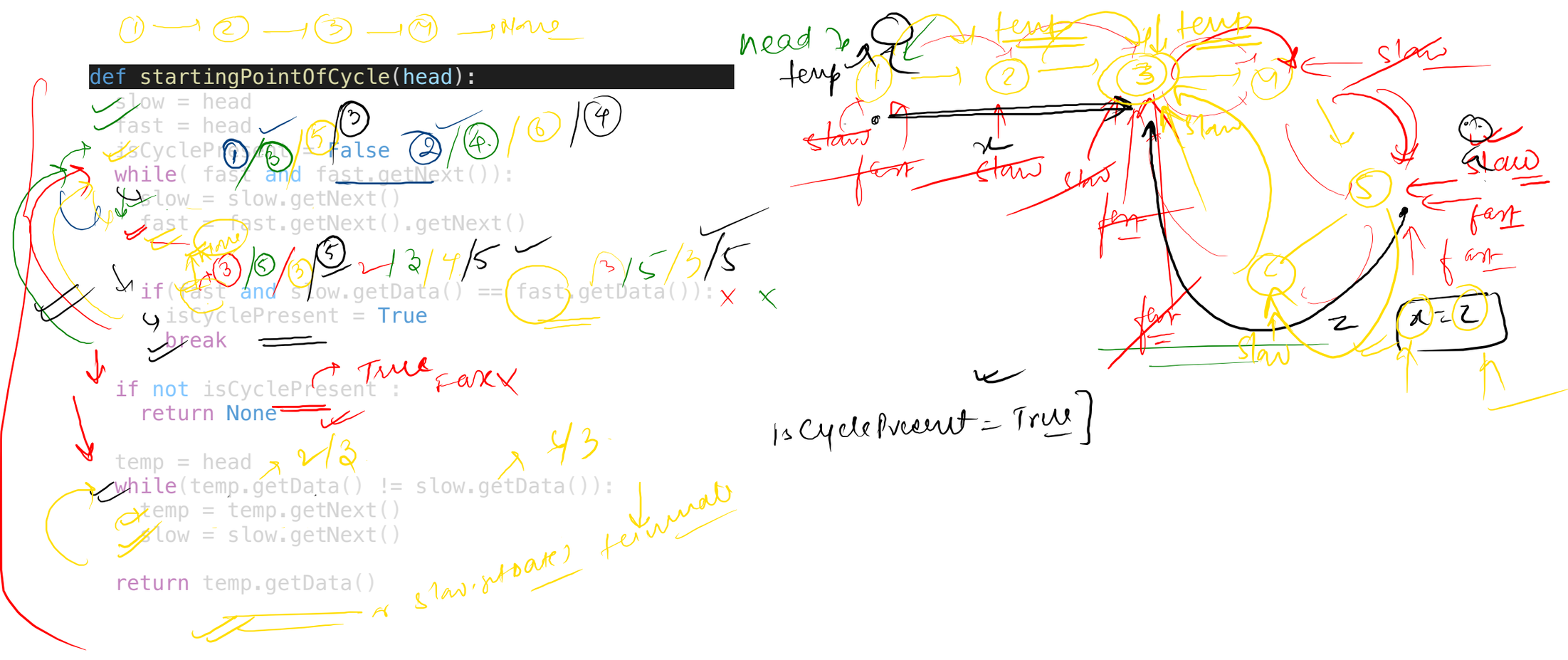
12:33 PM


long course
week
incremental
week 3
week 2
week 1
Backlog

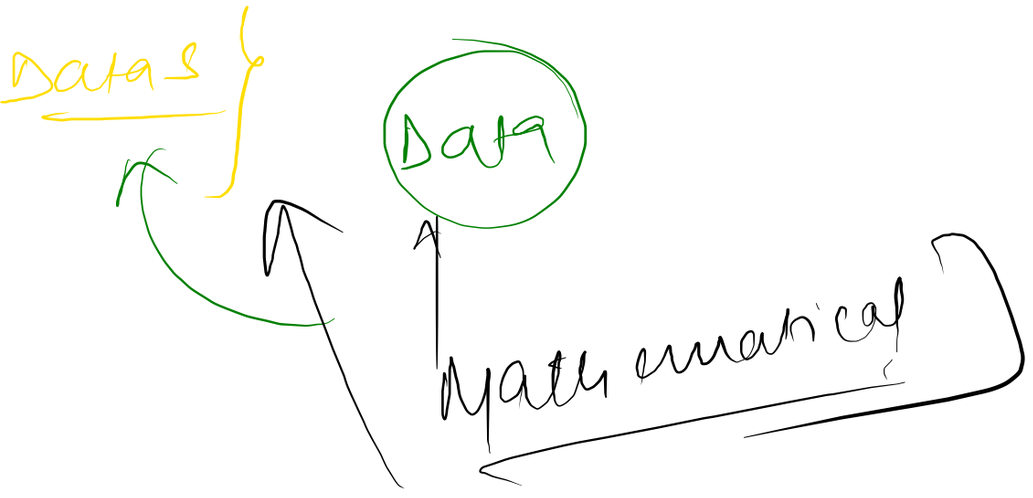

Remaining days
Dec
Relaxed
1-2 days
Revision classes
Doubts / concepts

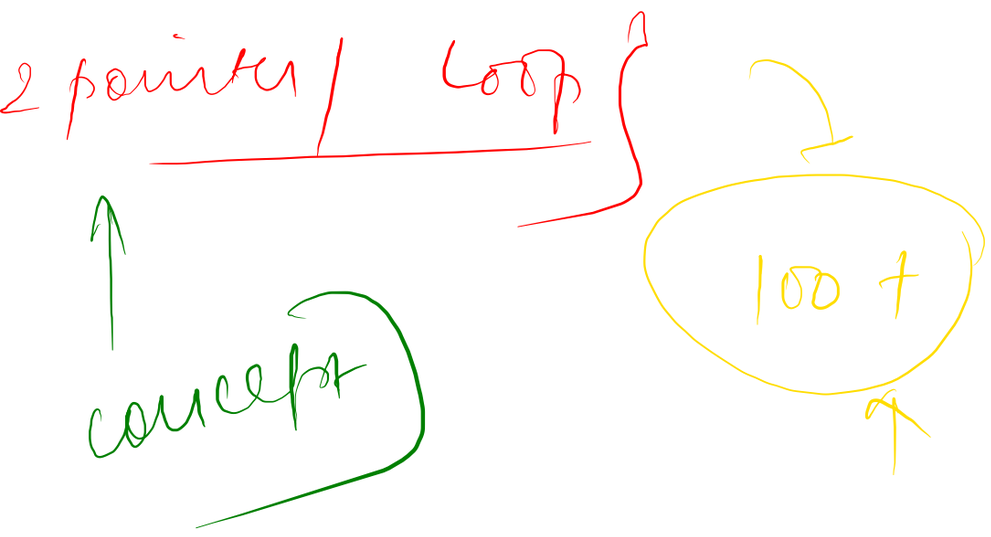
Nancy } Financey]

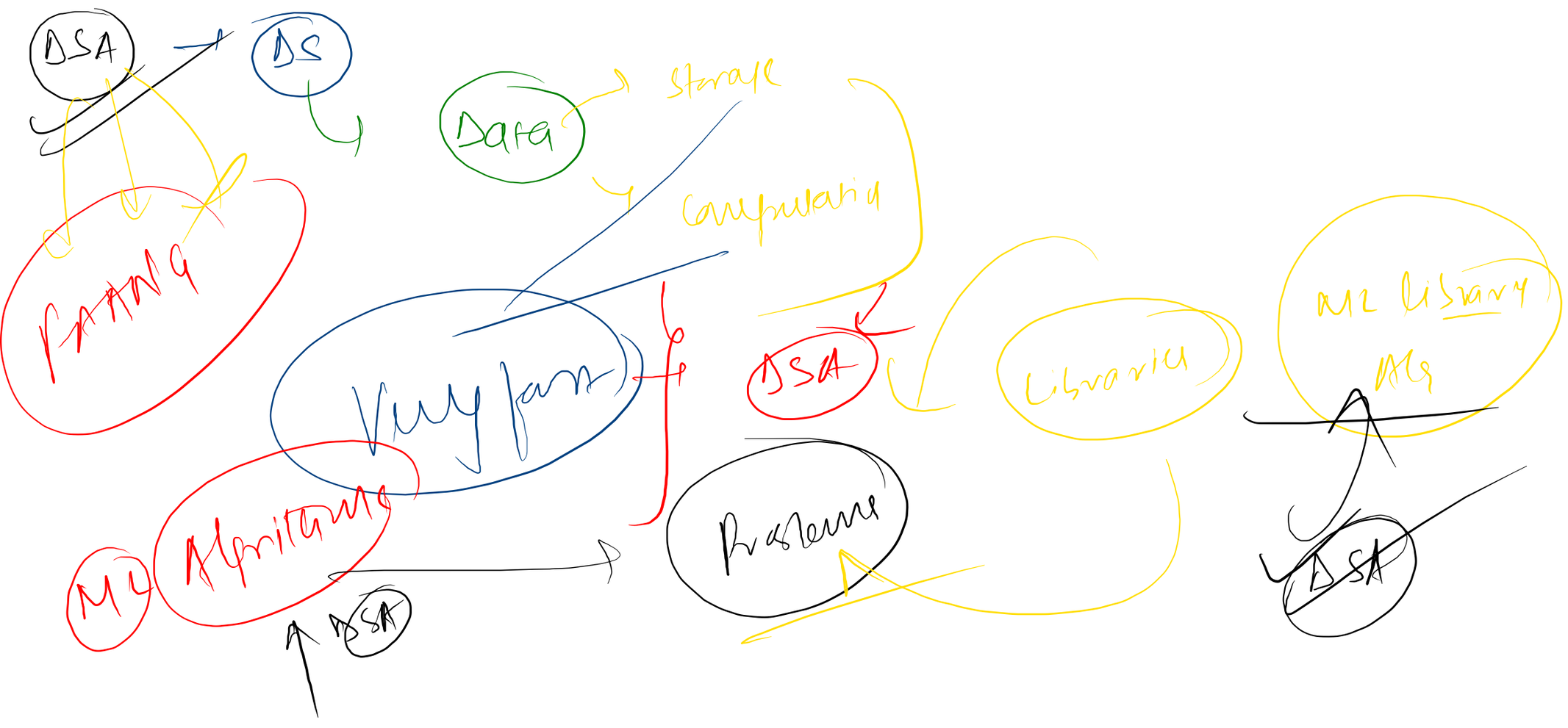
Book mining course]



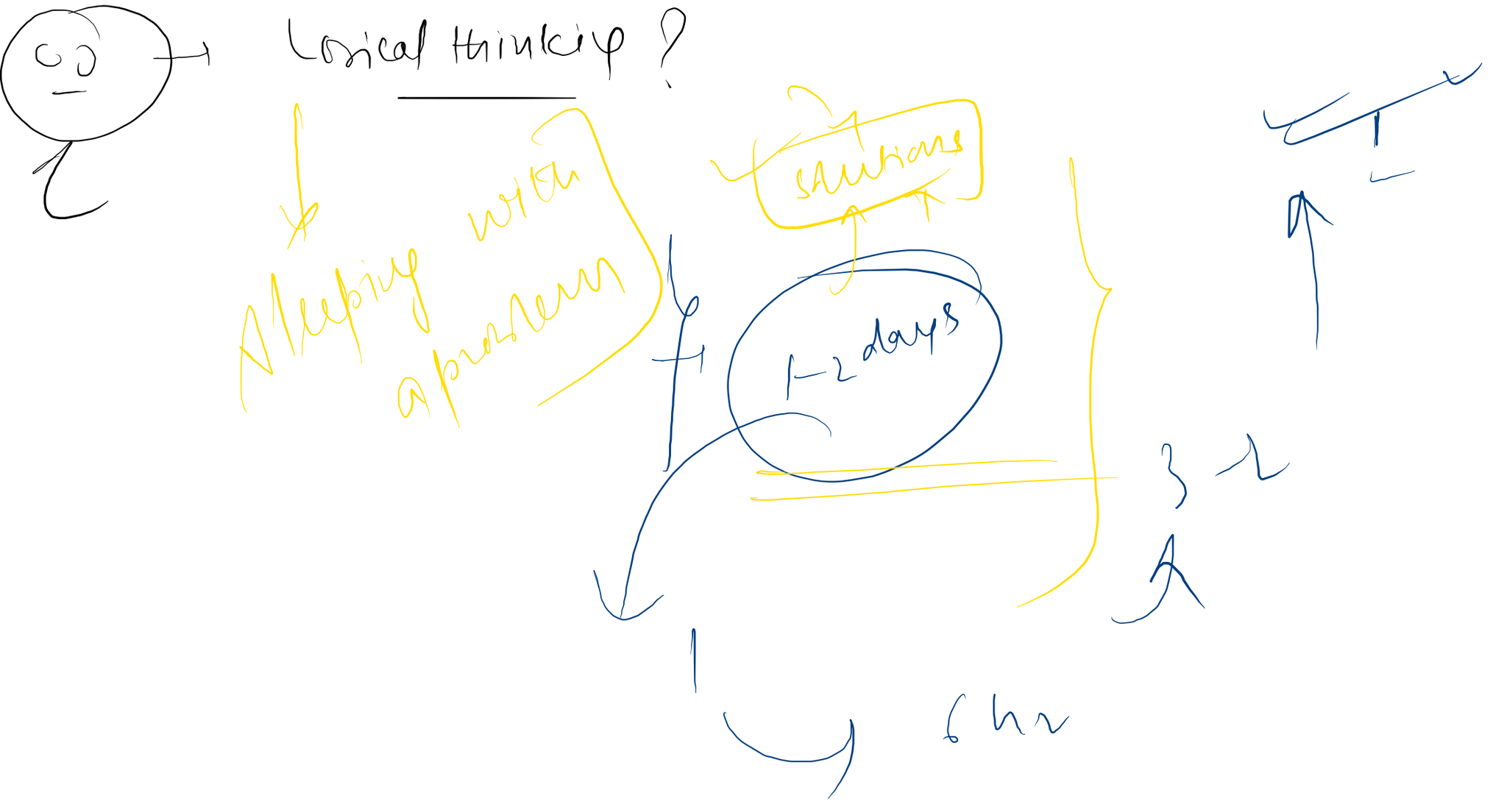
```
def startingPointOfCycle(head):
    slow = head
    fast = head
    isCyclePresent = False
    while( fast and fast.getNext()):
        slow = slow.getNext()
        fast = fast.getNext().getNext()

        if(fast and slow.getData() == fast.getData()):
            isCyclePresent = True
            break

    if not isCyclePresent :
        return None

    temp = head
    while(temp.getData() != slow.getData()):
        temp = temp.getNext()
        slow = slow.getNext()

    return temp.getData()
```

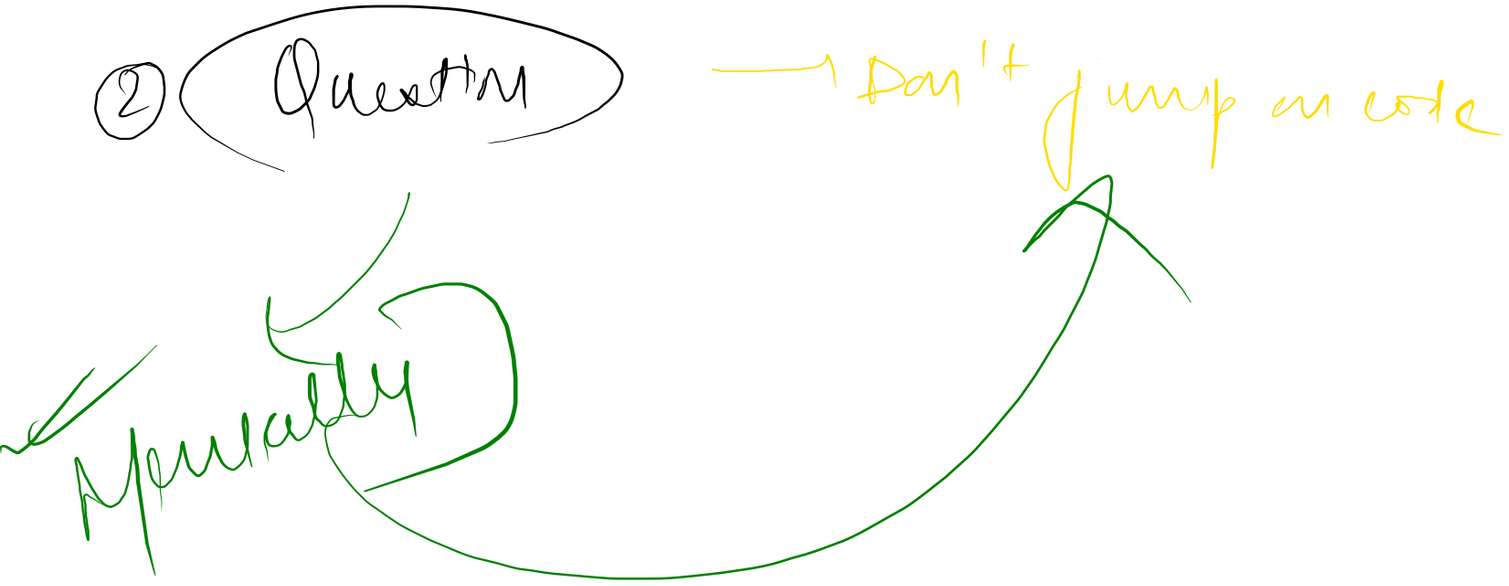

Data
Data
Mathematical


2 pointers / Loop
concept
100 %


DSA
DS
Data
Storage
Computation
Coding
Very fast
DSA
Libraries
ML Library
Algorithms
ML
Problems
DSA
DSA


Revision
Wednesday

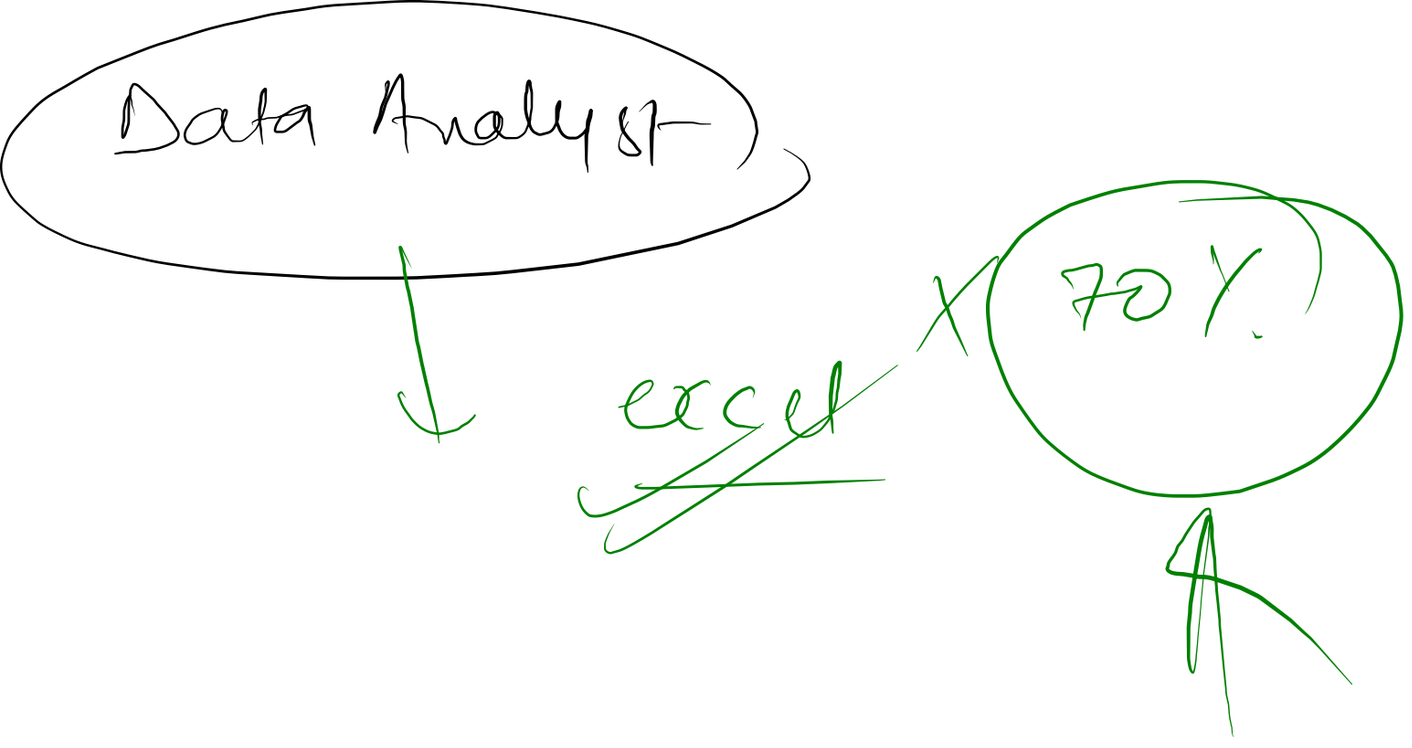

Logical thinking ?
Stuck
Stepping with a problem
1-2 days
1
6 hr
3 hr

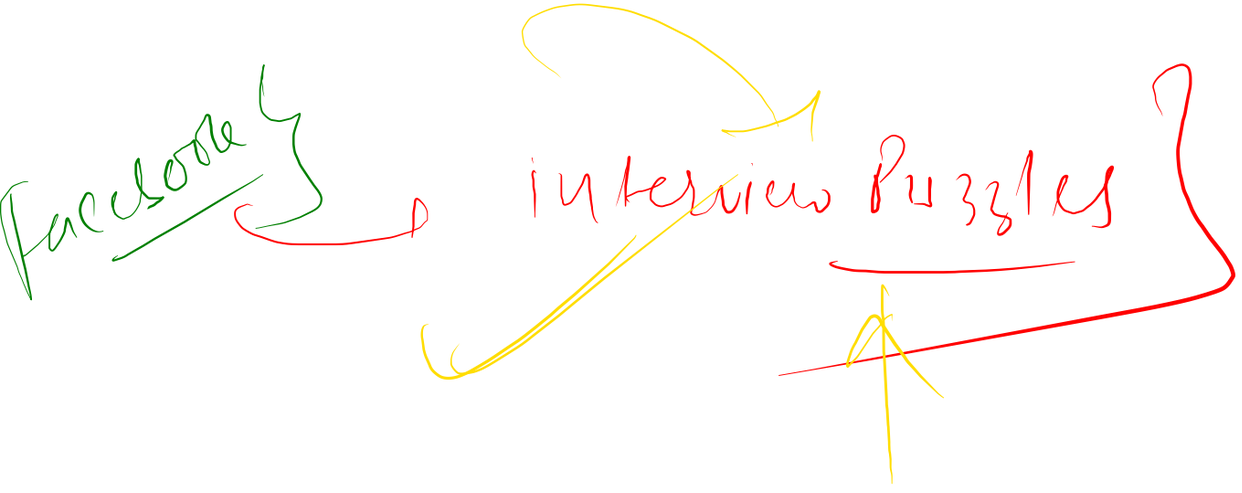

2
Question
Don't jump on code
Manually

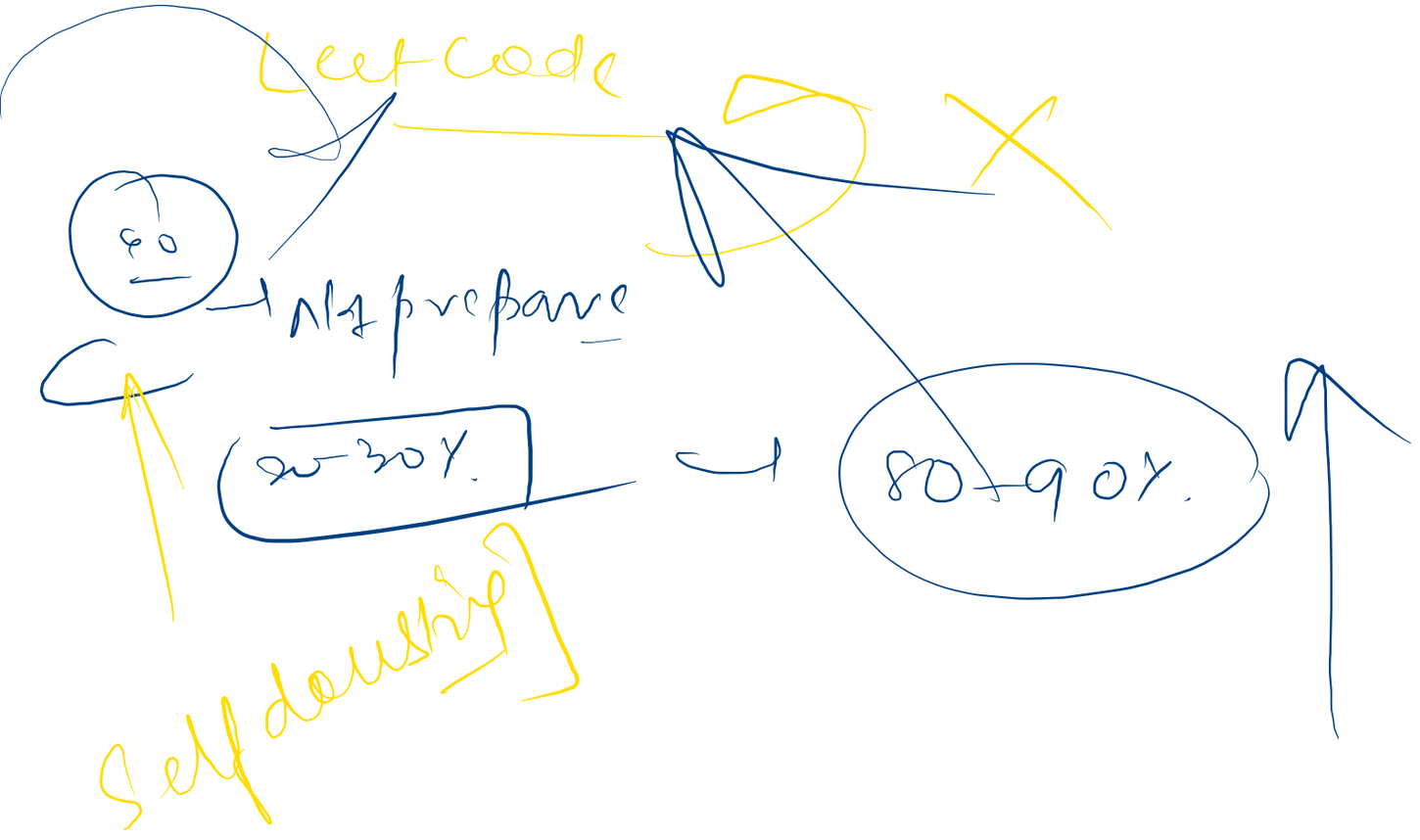
③ Logical Puzzles

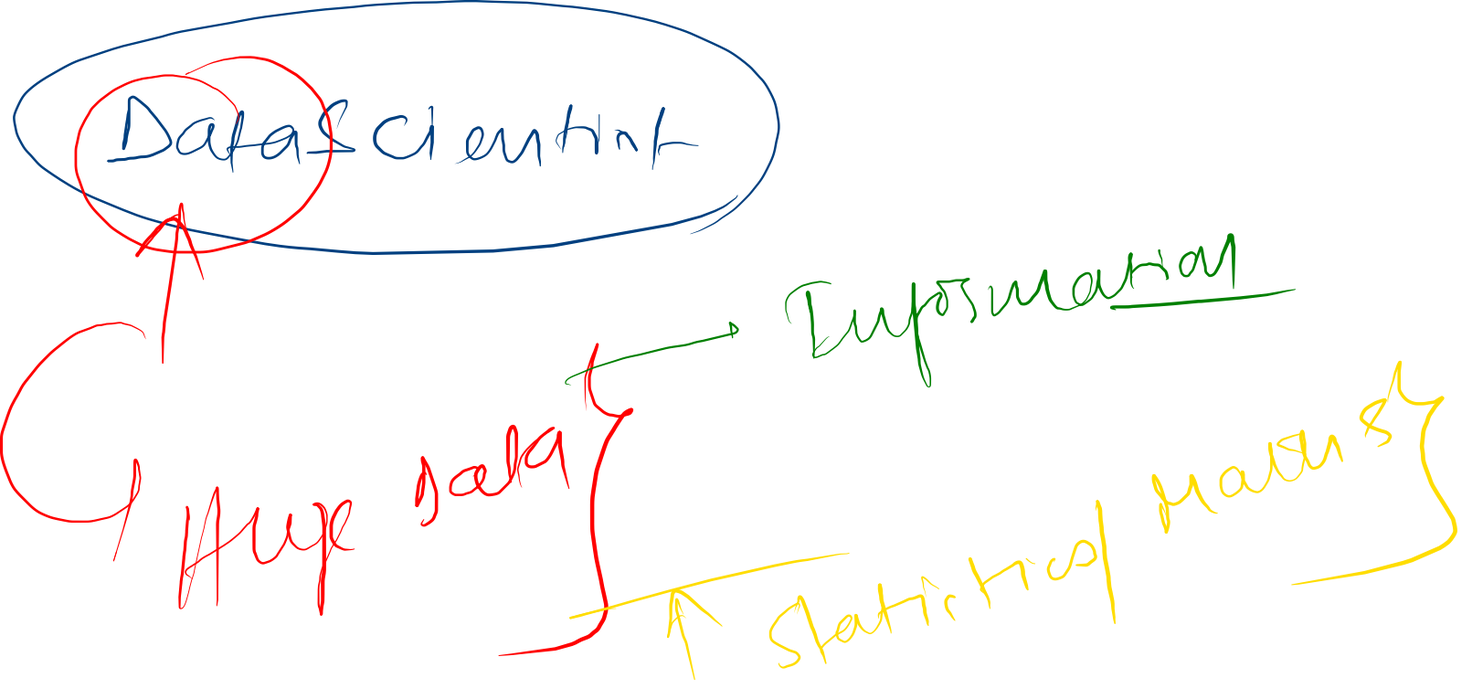
Training yourself

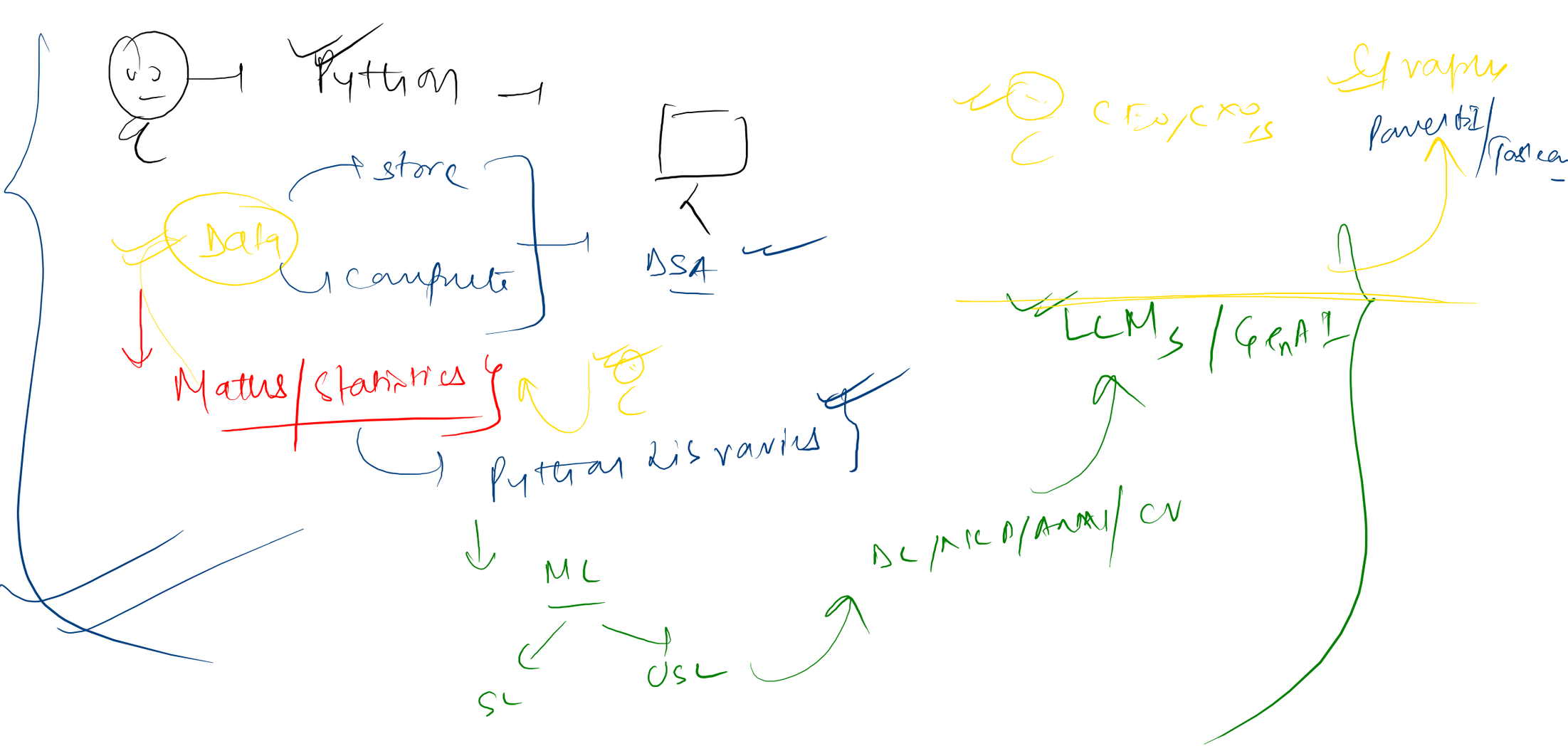
Patience

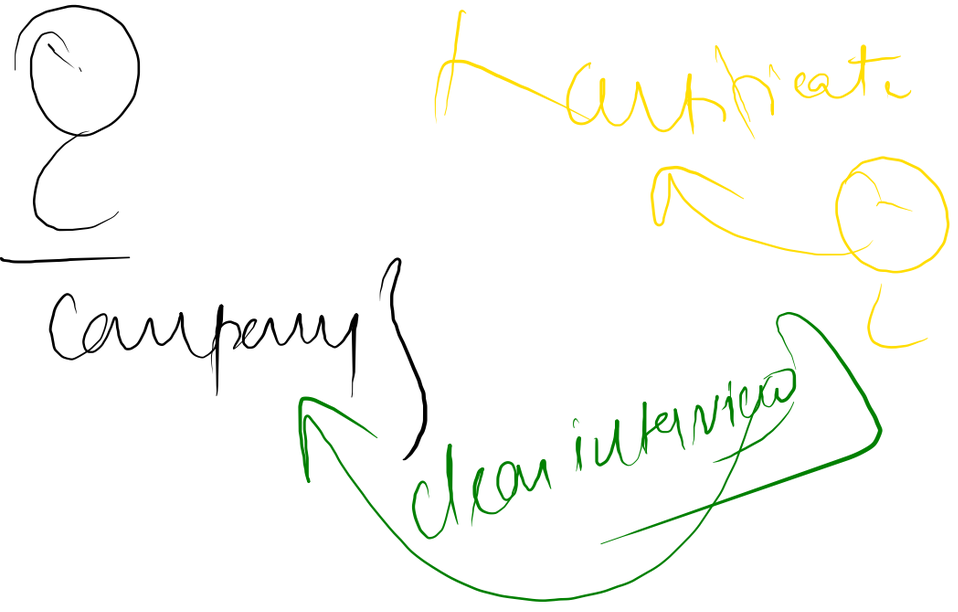

Data Analyst
excel
70%

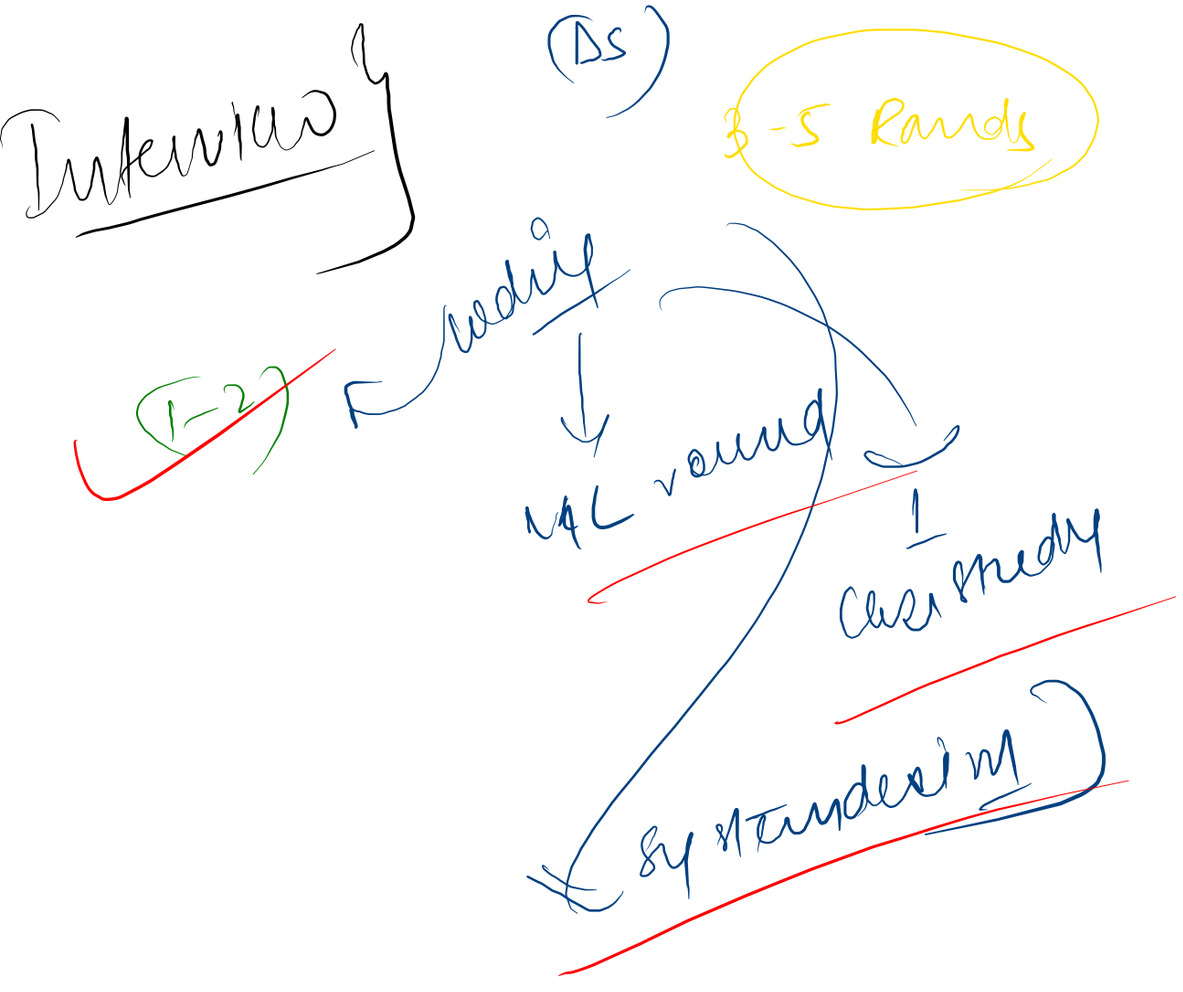

Facebook
interview Puzzles

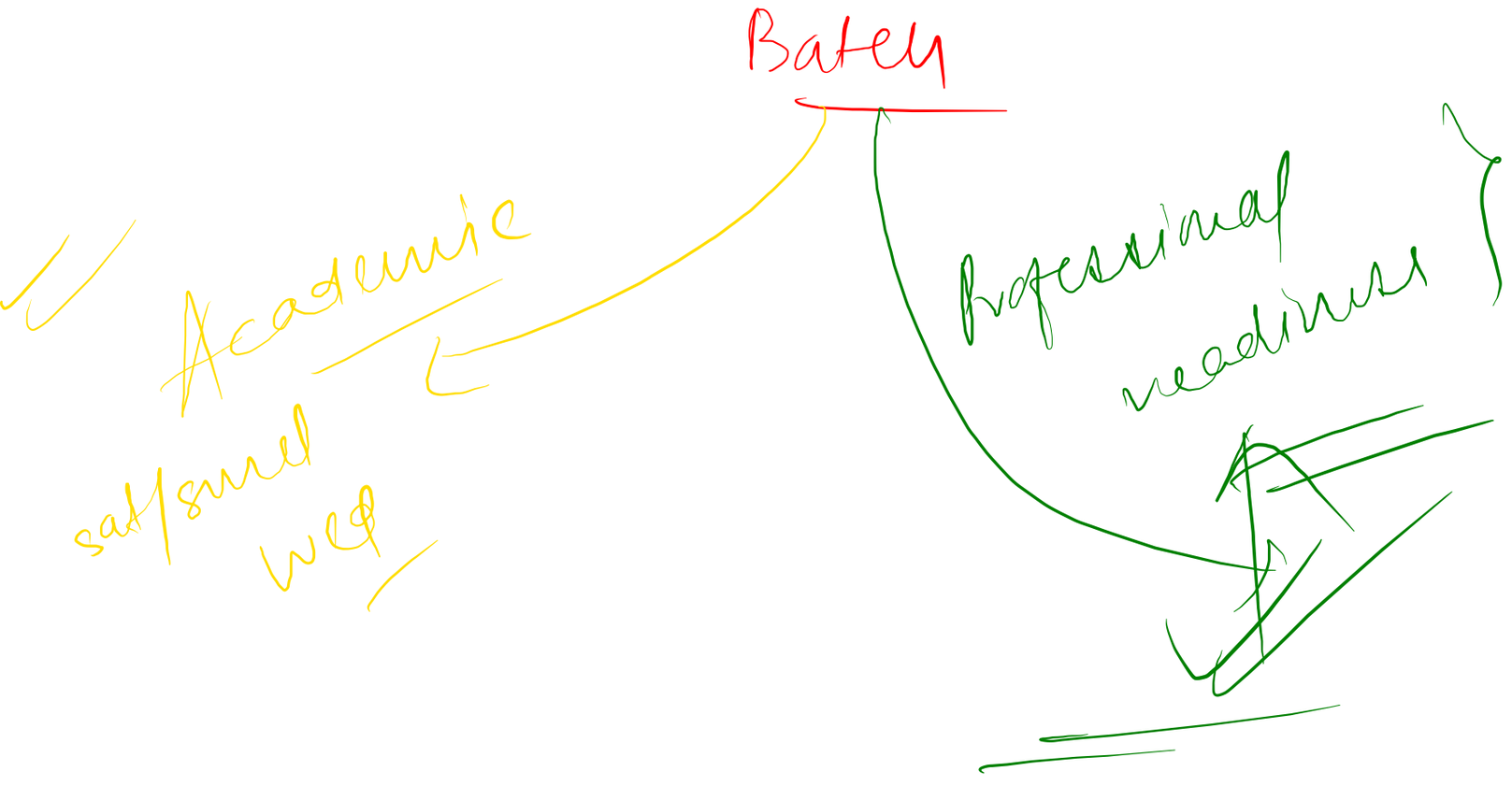

Leet code
X
Alg prepare
20-30%
80-90%
Self doubt

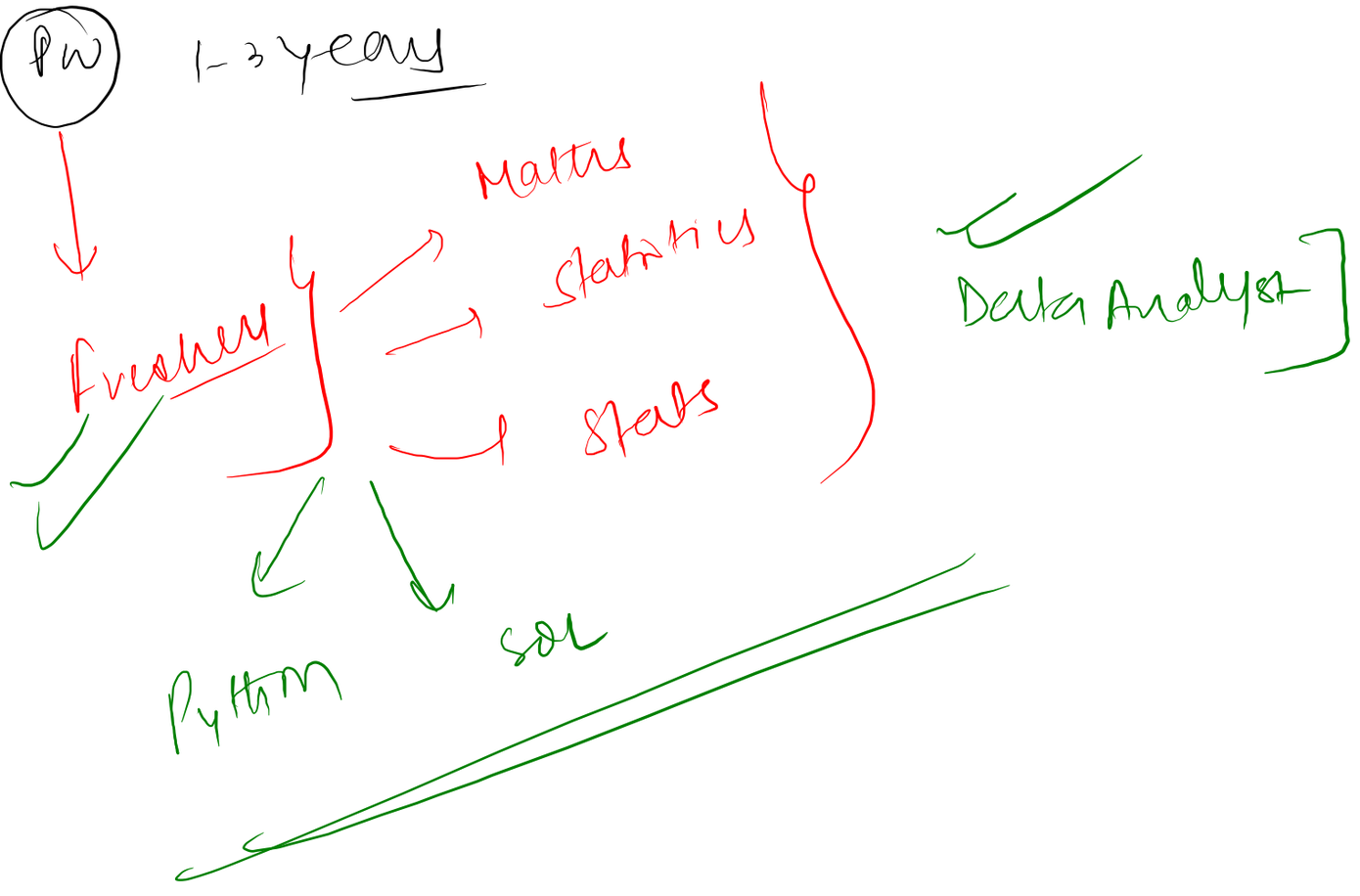

Data Scientist
Programming
Hyp data
Statistical math

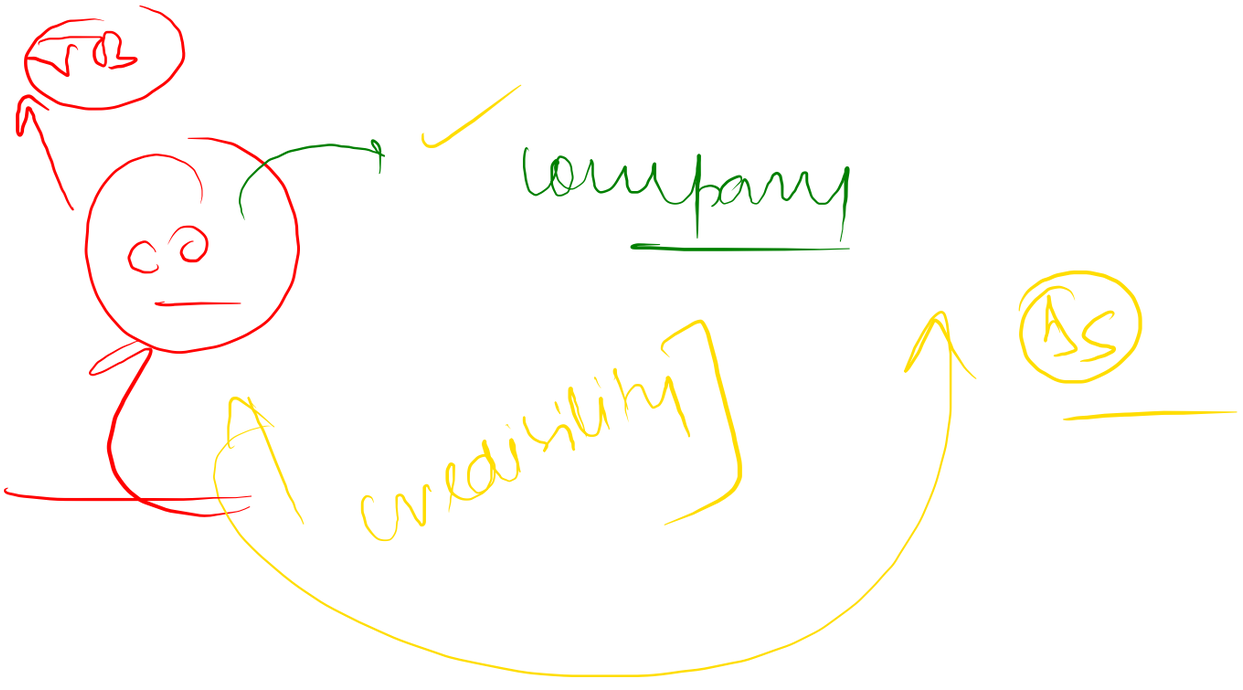

Python
store
Data
compute
DSA
Maths/Statistics
Python libraries
ML
SL
USL
DL/NLP/ANN/CV
LLMs/GenAI
Langchain

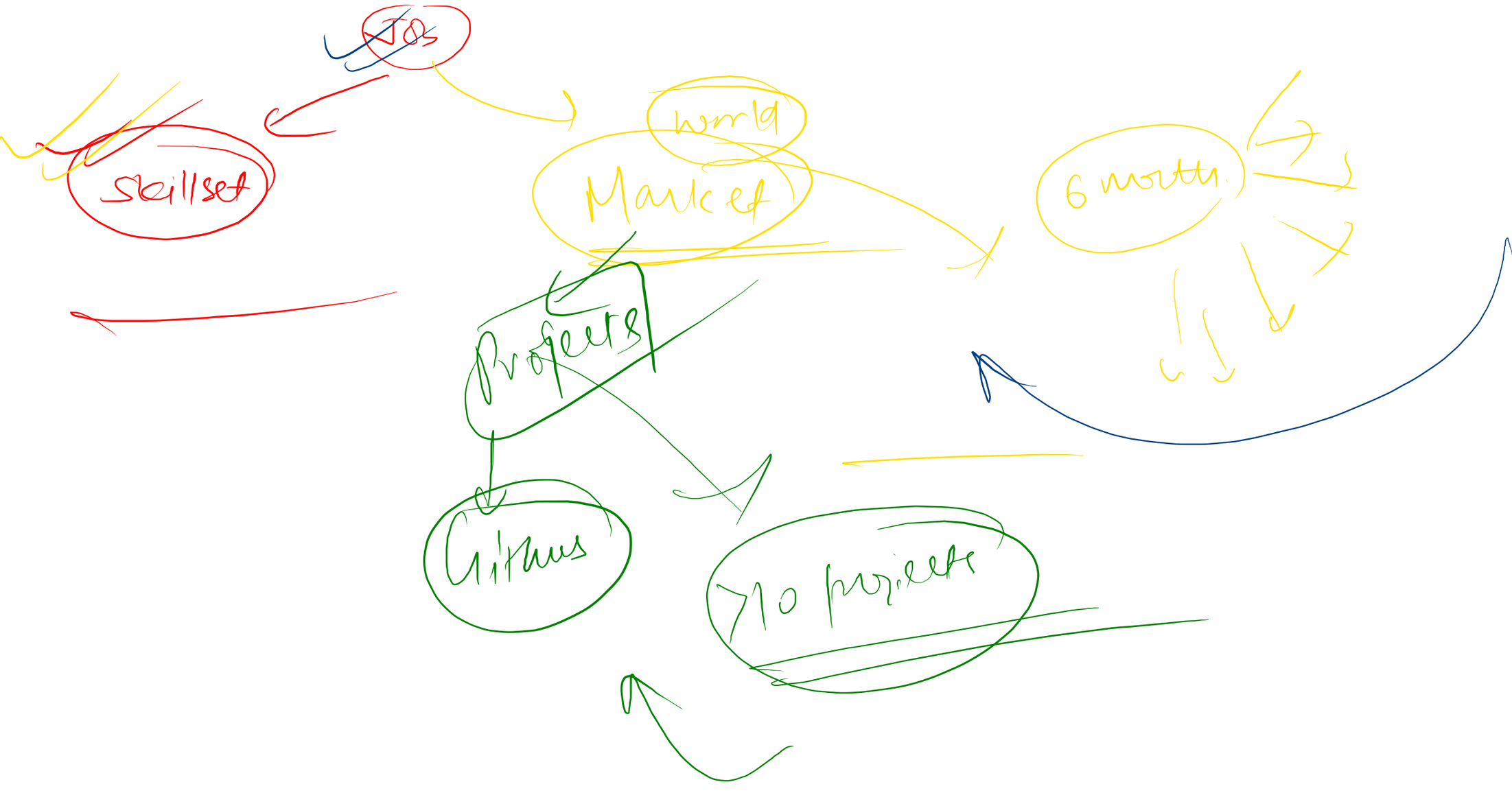

Company
clear interview

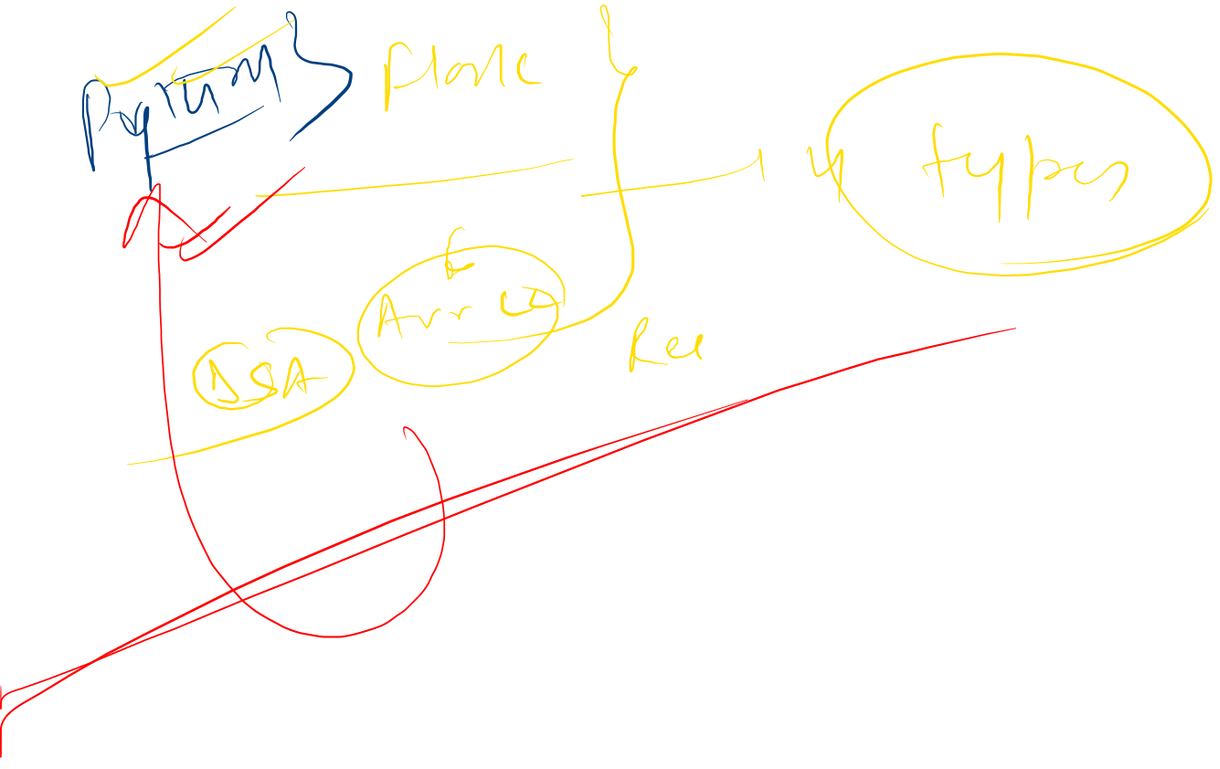

Interview
(DS)
3-5 Rounds
1-2
Coding
1
Case study

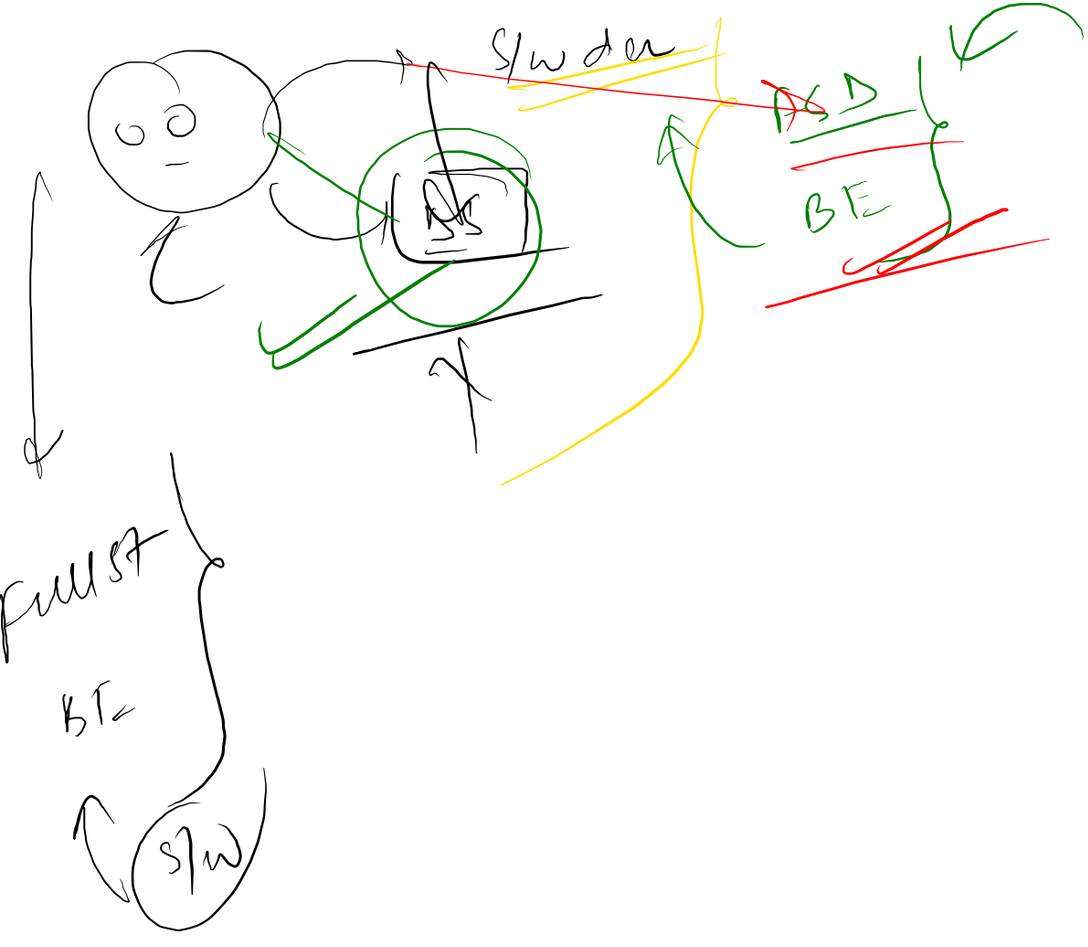

Batch
Academic
satisfied
Professional
readiness


PW
1-3 years
Maths
Statistics
Stats
Fresher
Data Analyst
Python
SQL


company
credibility
As


JDs
Skillset
world
Market
6 months
Projects
Github
>10 projects


Python
Plone
types
Avro
JSA
Rec


S/w

Cam → 9 months

Pre placement Training

PPT

MD

Spro

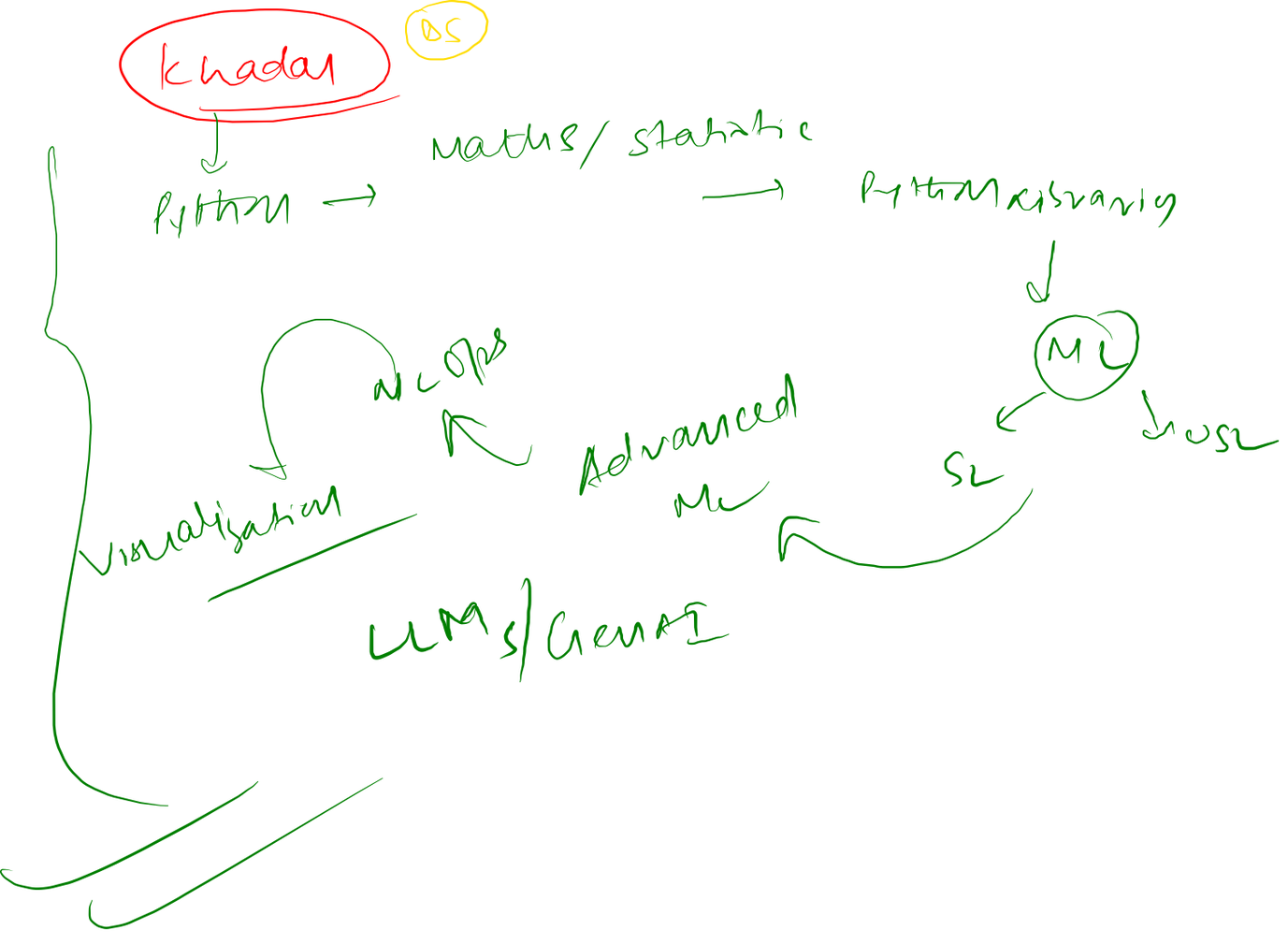
LI

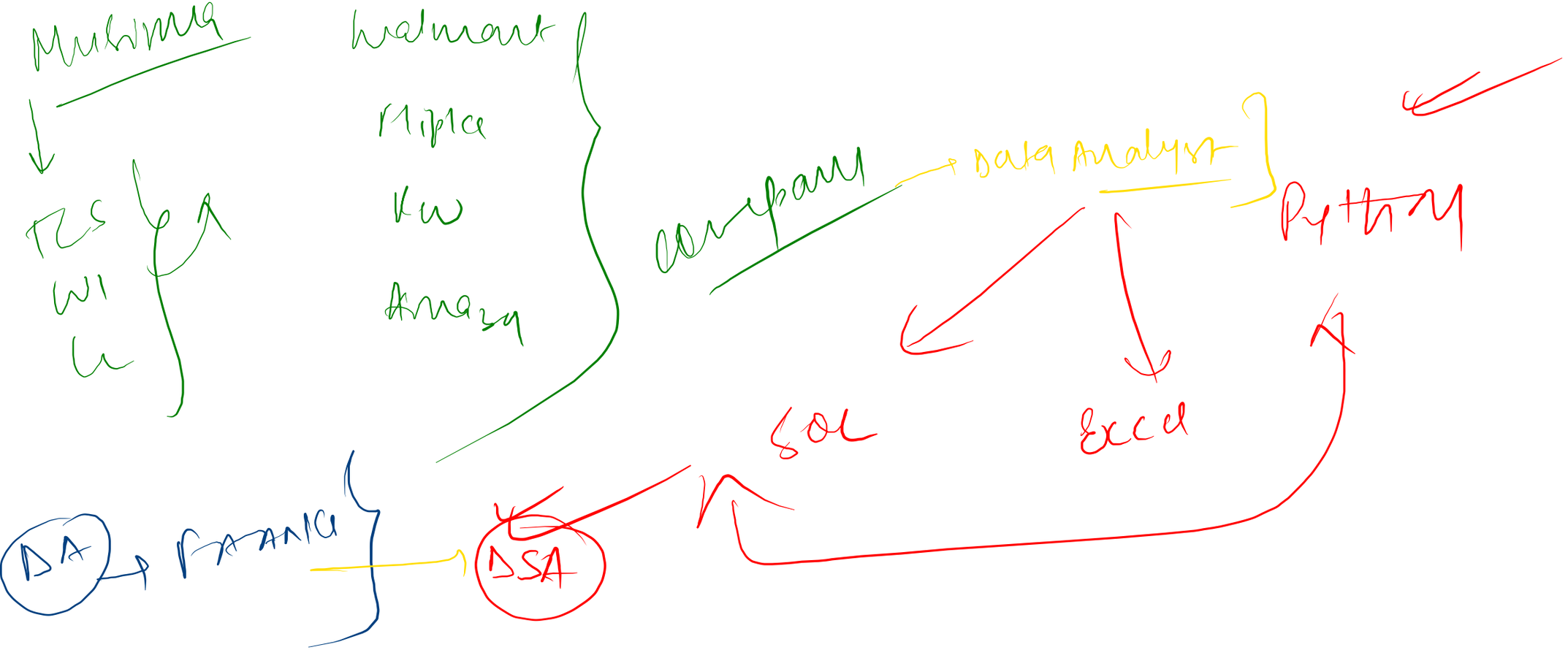
Re

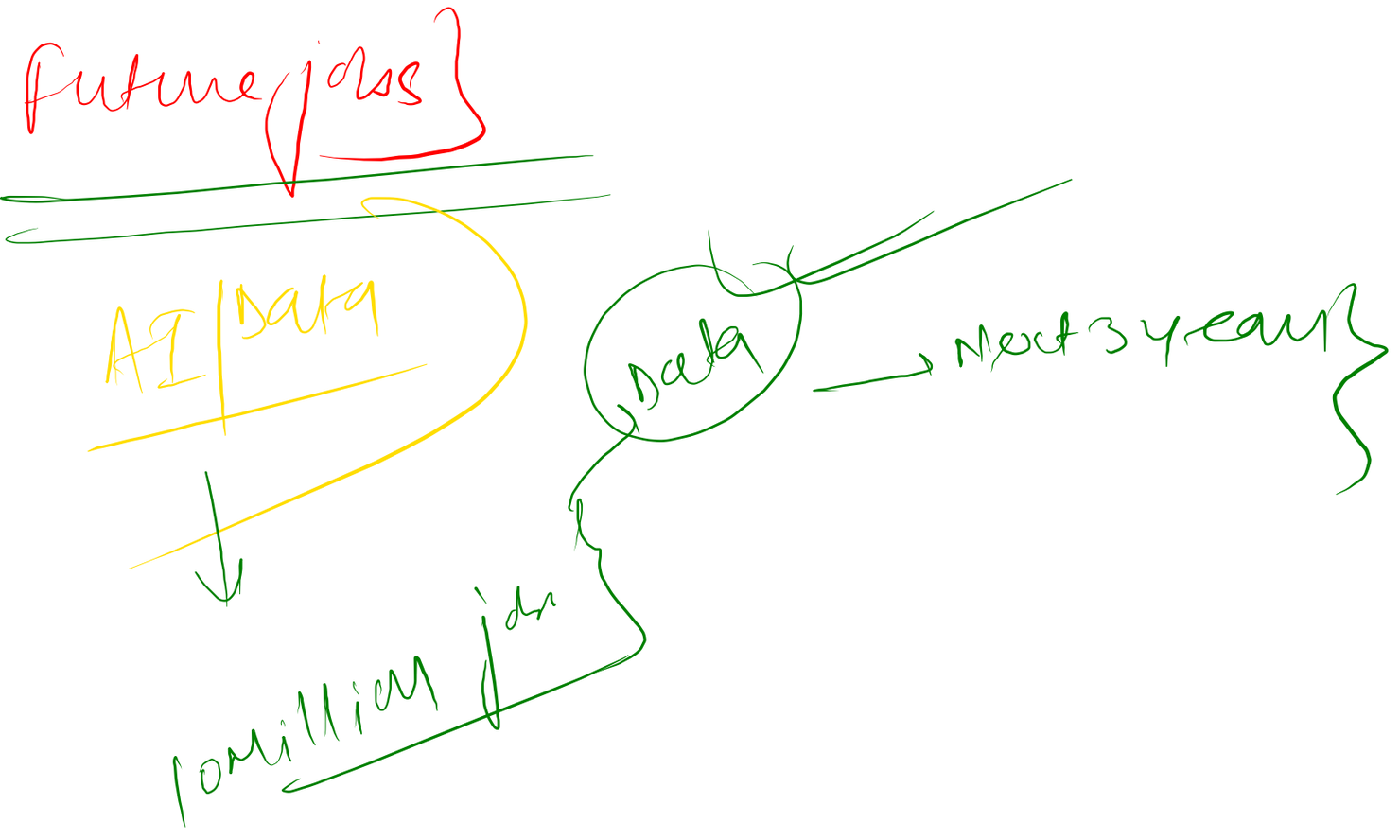
Readiness for Quick verification

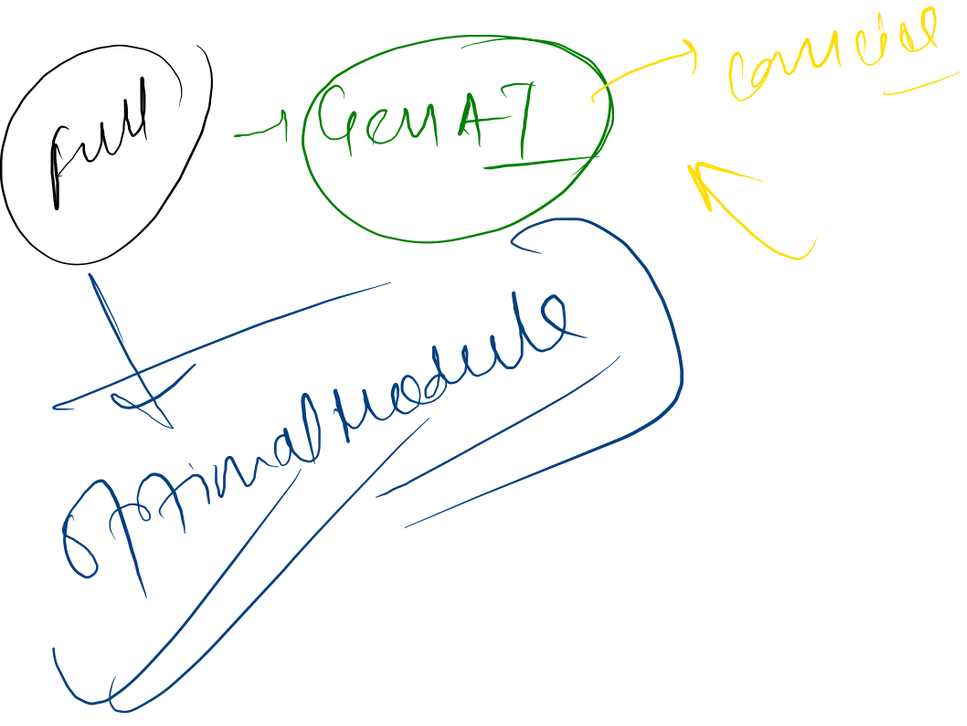
Tele Interview

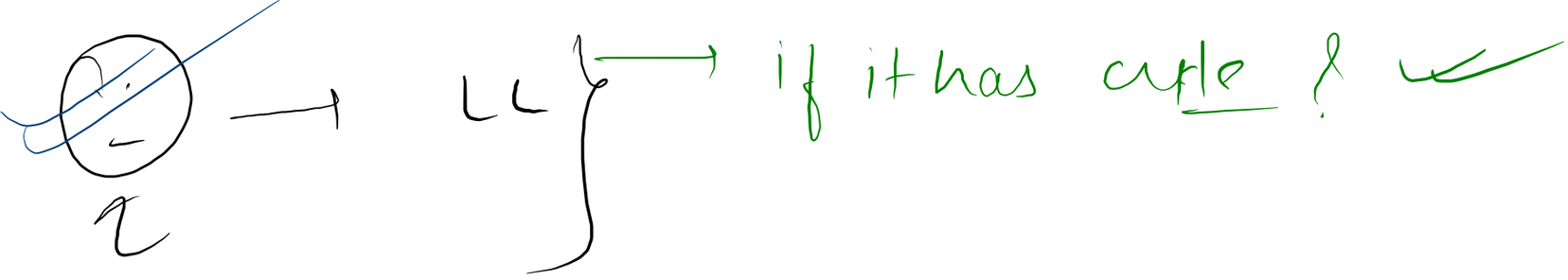
Test

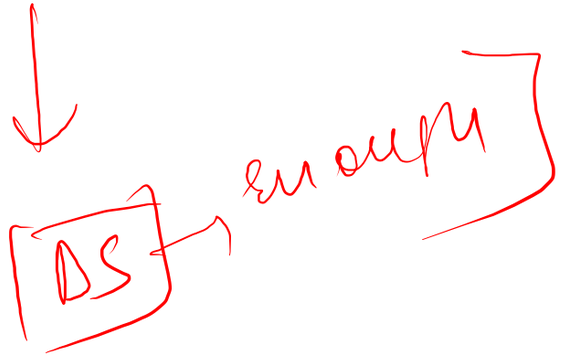

Khadar
DS
Python
Maths/ Statistic
Python Library
ML
MLOps
Advanced ML
SL
USL
Visualisation
LLMs/GenAI

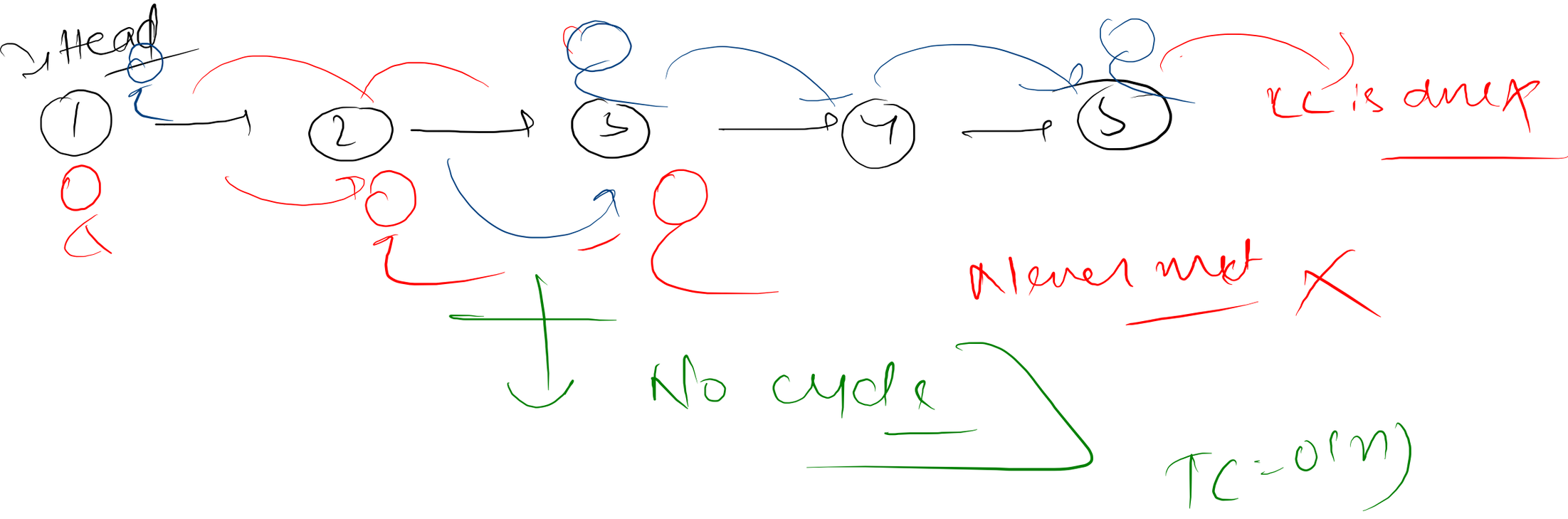

Walmart
Myntra
Flipkart
Amazon
TCS
Wipro
company
Data Analyst
Python
SQL
Excel
DA
Banking
DSA


future jobs
AI
Data
Data
Next 3 years
10 million jobs


pull
GenAI
corrected
Optimal module


LL
if it has cycle


DS
empty


head
1
2
3
4
5
LL is cyclic
Never met
No cycle
TC - O(n)

```
def findMidPoint(head):
    slow = head
    fast = head

    while(fast != None and fast.getNext()!= None):
        slow = slow.getNext()
        fast = fast.getNext().getNext()

    return slow.getData()
```