Splitwise

How to compute List<Transaction>

Scenario

Group A - B - C - D

Expense 1

amount = 1000

paid By A: 500 B: 500

owed By A: 250 B: 250 C: 250 D: 250

equally
manually

Expense 2

amount = 500

paid By C: 500

A: 150

Owed By A: 150 B: 50

Expense 3

amount = 2000

paid By = A: 500 B: 500
C: 500 D: 500

Owed By : A: 800 B: 800
C: 200 D: 200

→ Since we need to calculate how much extra has everyone paid

① add how much they had paid

② Subtract how much they should be paid

+500 −250 + 0 − 150 +500 −800

A ⇒ −200

they should have paid

$+500 + \cancel{500} - 250 - 200 \cancel{-500}$

→ +50

B

C

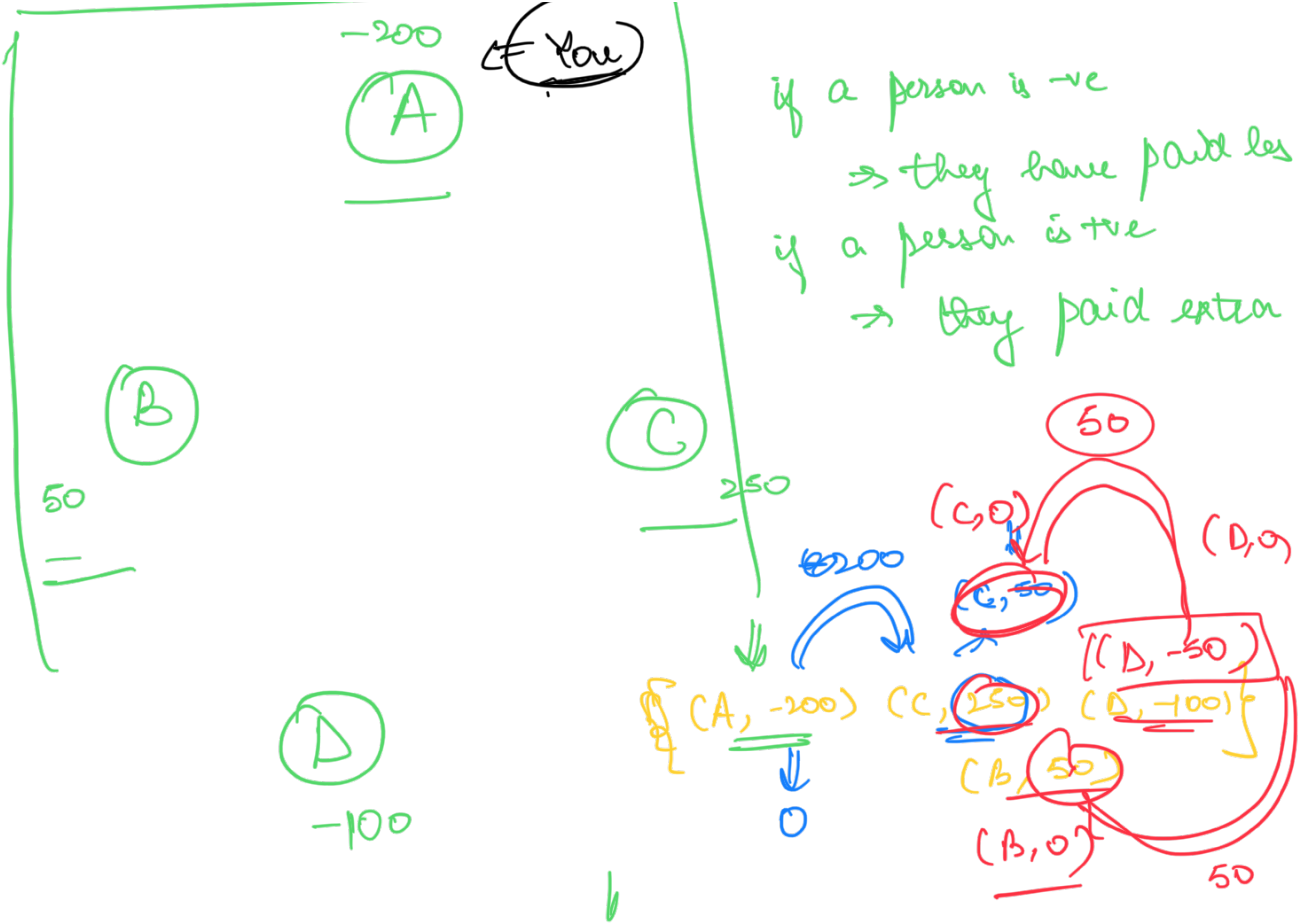
$500 + \cancel{500} - 250 - \cancel{500}$

+250

D

$500 - 250 - 150 - 200$

→ −100



if a person is -ve

⇒ they have paid les

if a person is +ve

→ they paid extra

Solution 1 : Give to next person

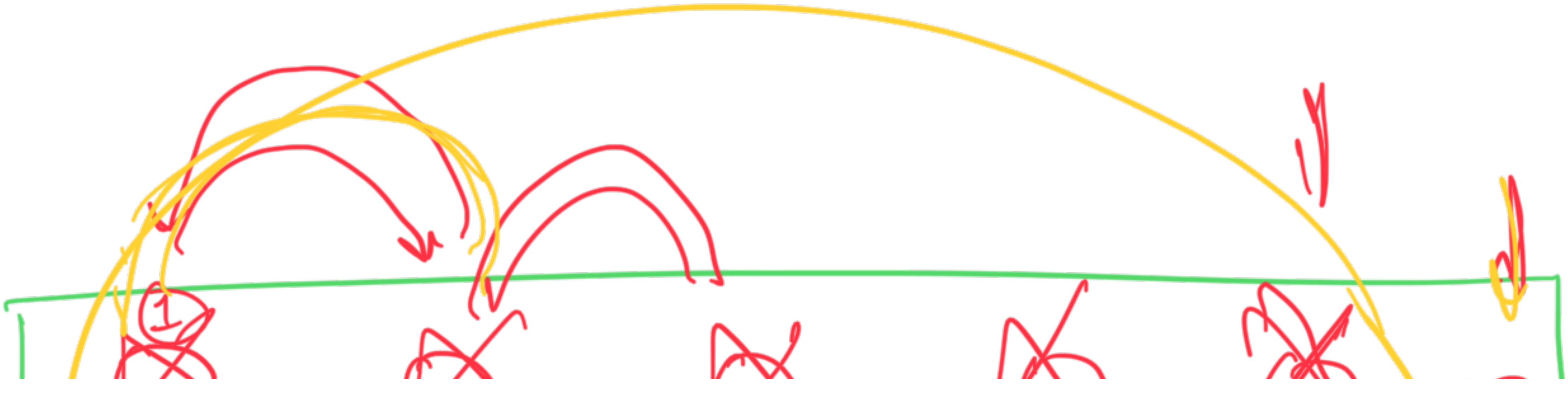
$O(N)$

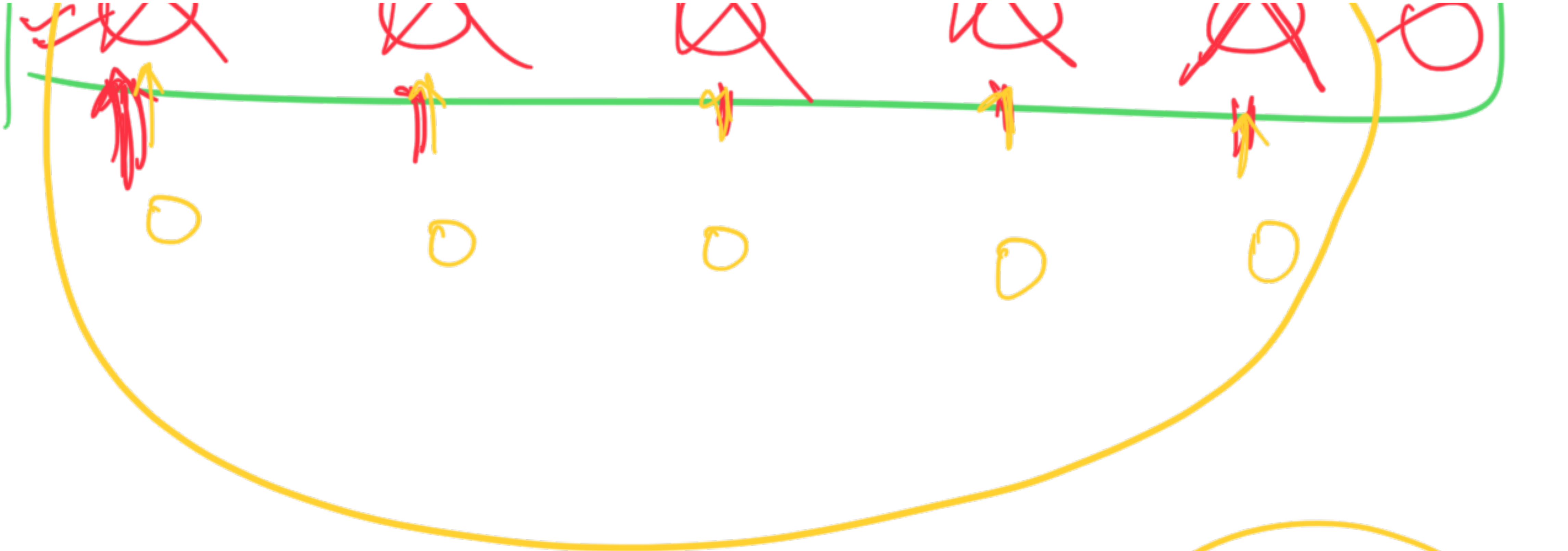
I care about myself

① either I will pay to the next one

② or I will get from the next one

and the next one will take care ahead



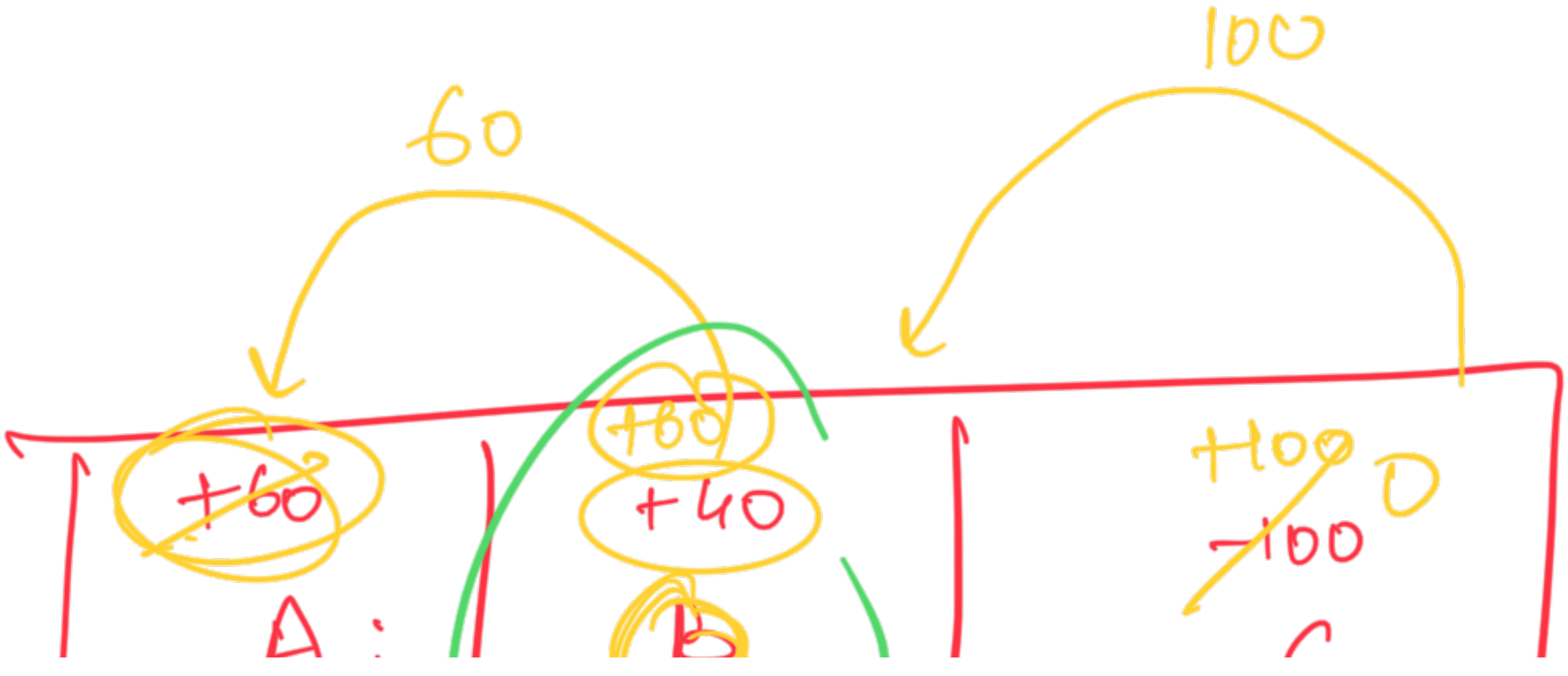


N-1

```
for (i = 0; i ≤ N-1 ; ++i) {
    if amount_extra > 0
        getMoney(i+1, i, amount_extra)
```

```
[ increaamount extra of i+1

else if amount_extra < 0
    → give Money (i+1, i, amount_extra)
        reduce extra amount from i+1
}
```

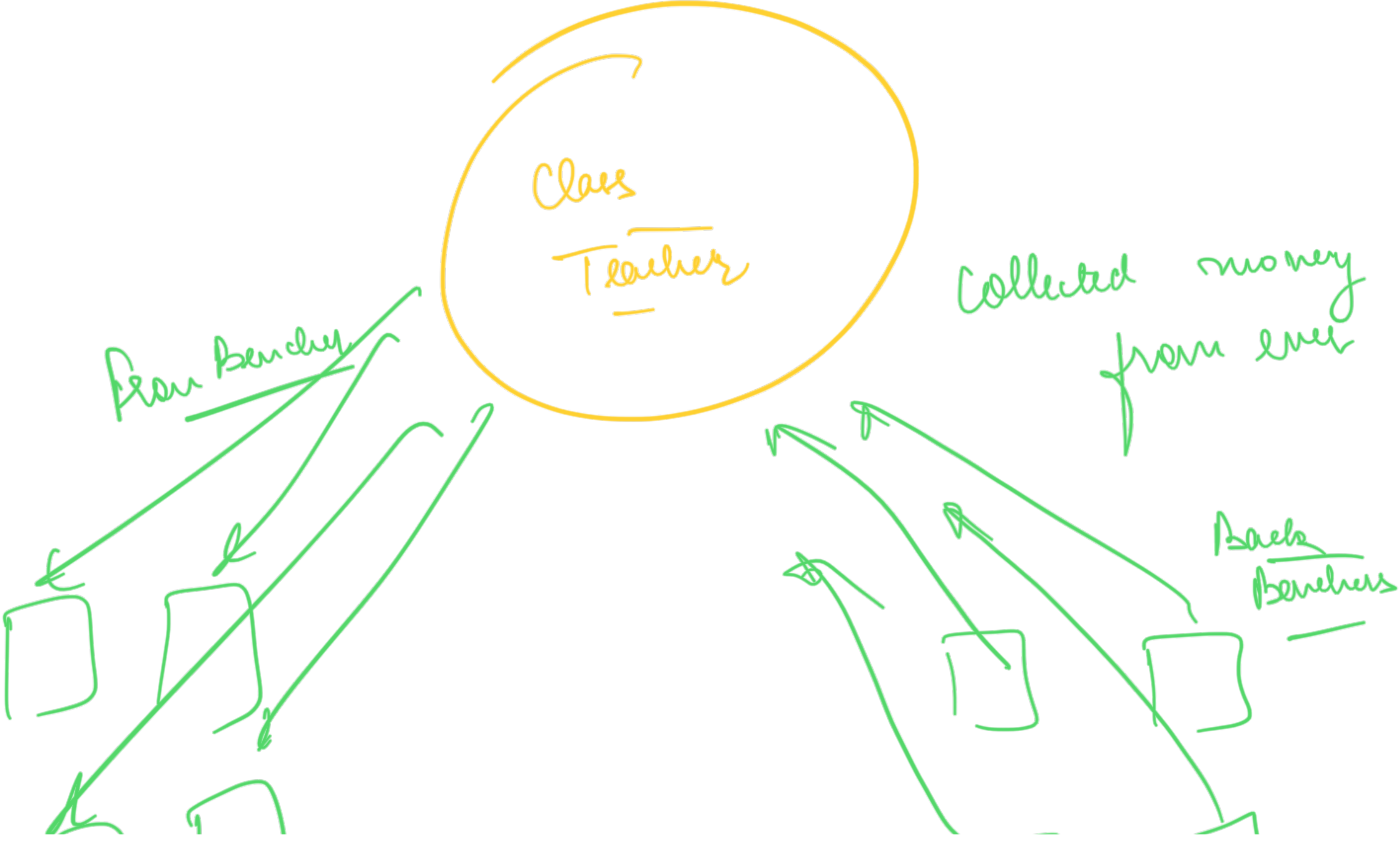


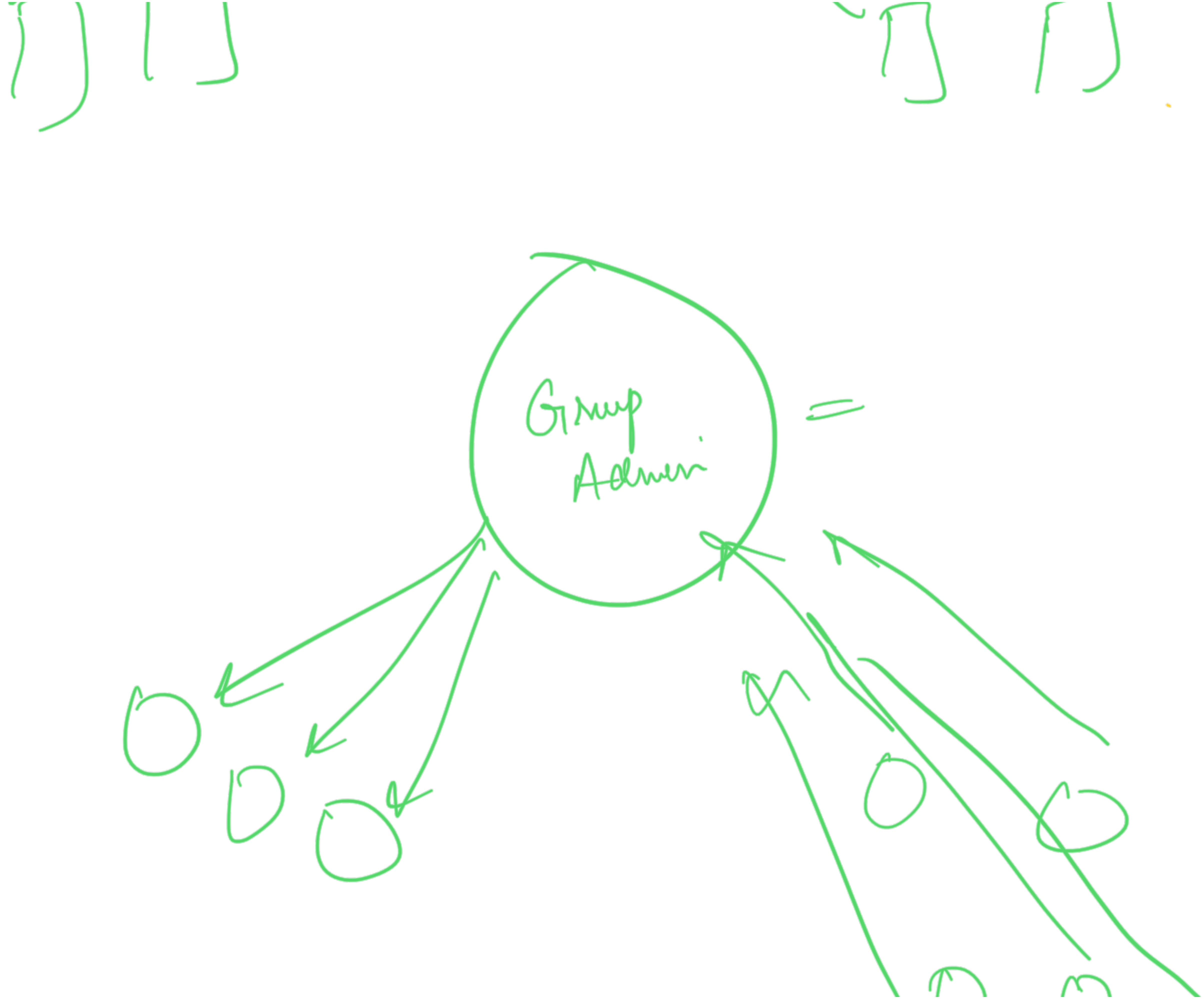
Transactions: (B, A, 60)

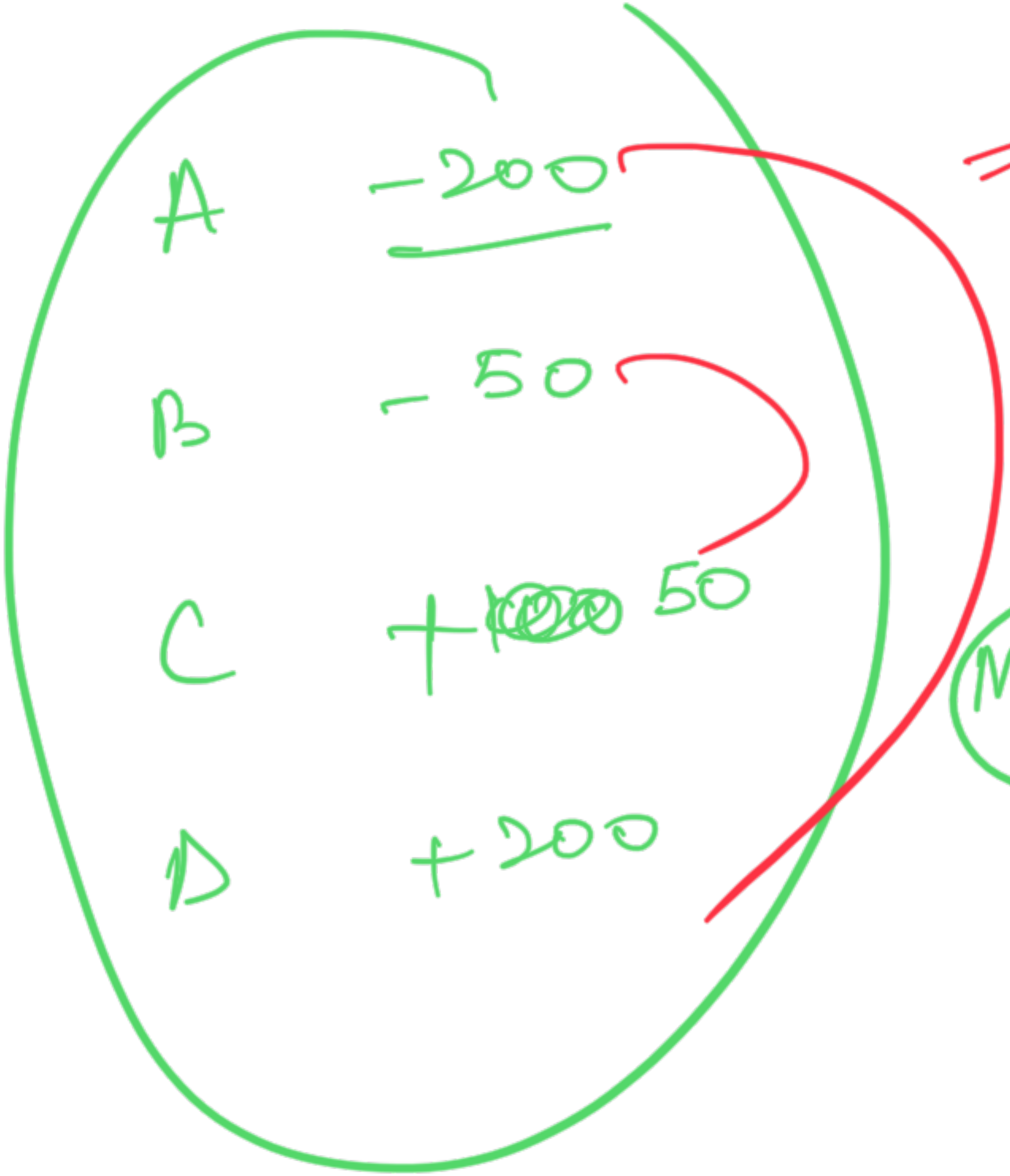
(C, B, 100)

Break till 10:25

Class Teacher Approach



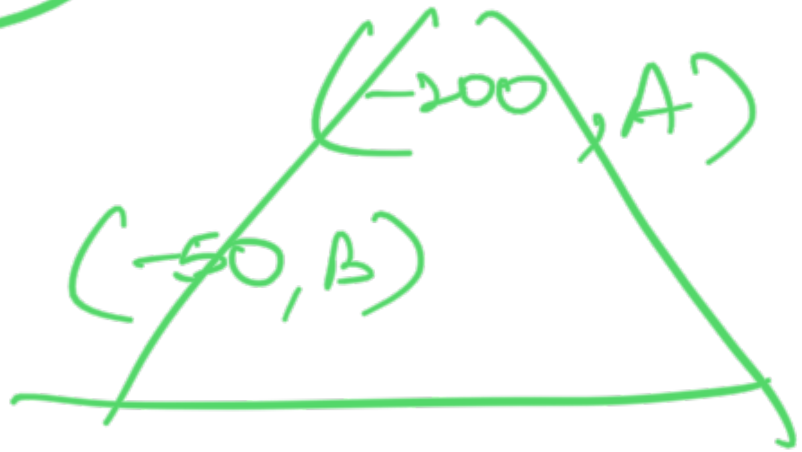

Group
Admin



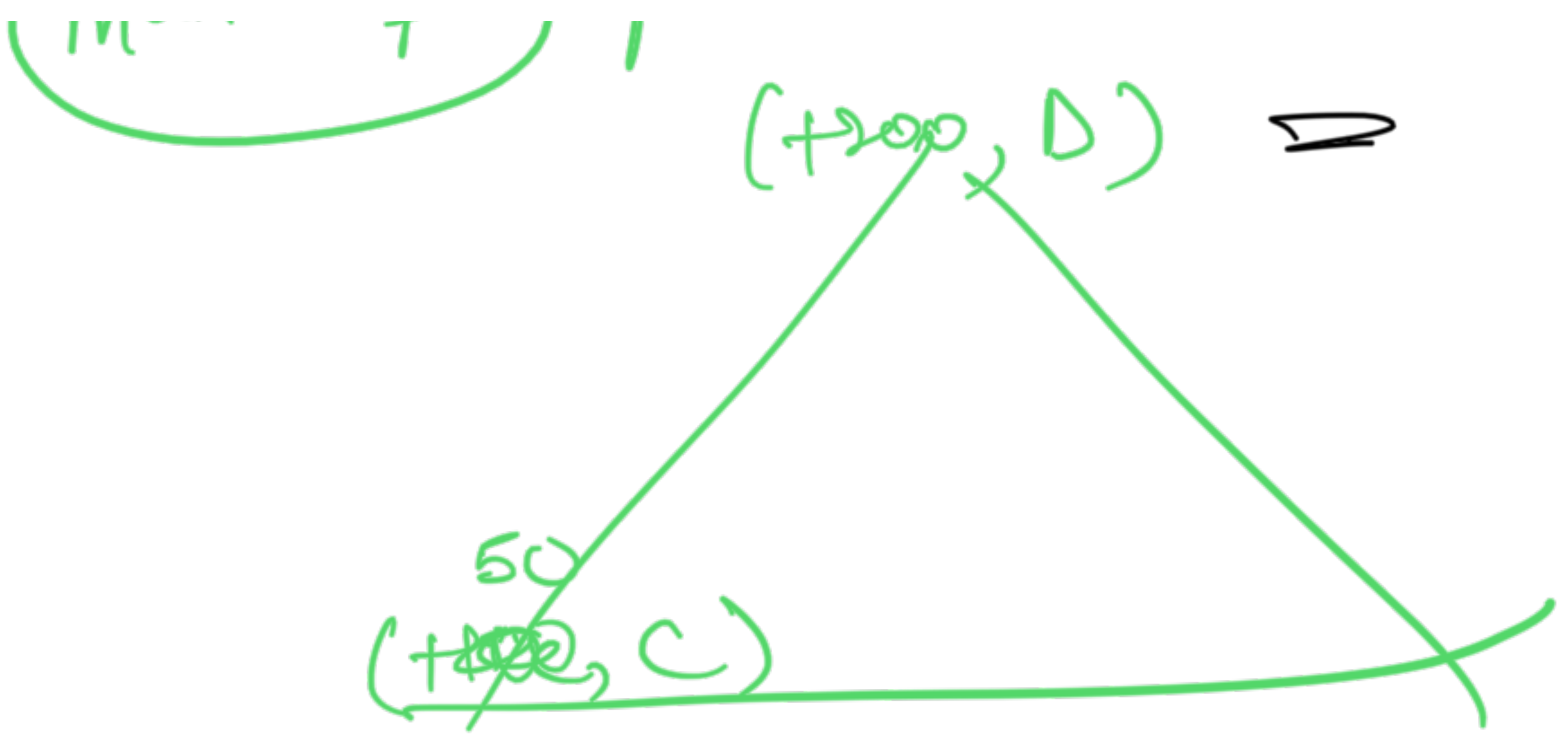
⇒ ②

Create 2 heap

Min Heap put all -ve ones



Max Heap put all +ve ones



At <u>every step</u>

a) get the minimum # from min heap

(−200, A)

b) get the max^m # from max heap

(+200, D)

c) ~~Set~~ Zero the person who has the lesser amount

iver Getter

Scenario 1 ~~(-150, A)~~ (200, D)

A gives 150 to D (D, 50)

Scenario 2 (-200, A) ~~(150, D)~~

D takes 150 from A ⇐ (A, -50)

Scenario 3 ~~(-200, A)~~ ~~(200, D)~~

A gives 200 to D

d) If there is a person who is still not zero, put them in the heap via

= $\boxed{n \to 1000}$

150 ms

180 ms

Cons

1. Complex to implement
2. O(N logn)
3. Space comp of O(N)

Pro

① Real world practically

(1) Real world / ______

give you an
algo problem
tell you that don't care
about TC

Q Algo to guaranteed minimize the # of transactio

No X

in polynomial time

NP- Hard ✗

Brute Force