Let's learn

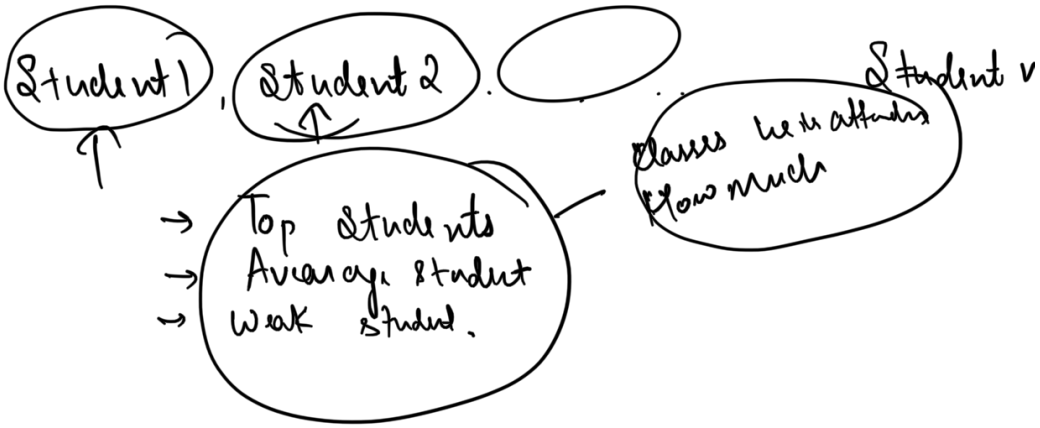
Sorting

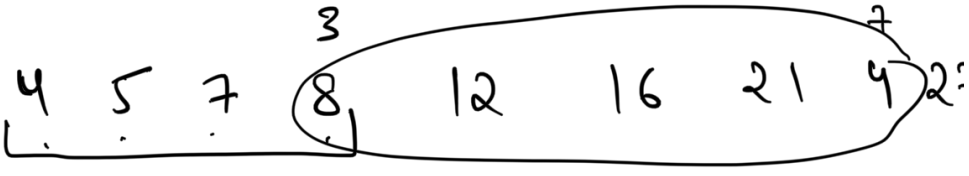
We will start the class at 8:34

→ (Aim) → Beginner Oriented class ✓

→ Fundamental sorting technique
→ Stability in sorting and inplace sorting
→ $O(n^2)$

→ Sorting ? what is sorting !
→ easier access





(0, (4))

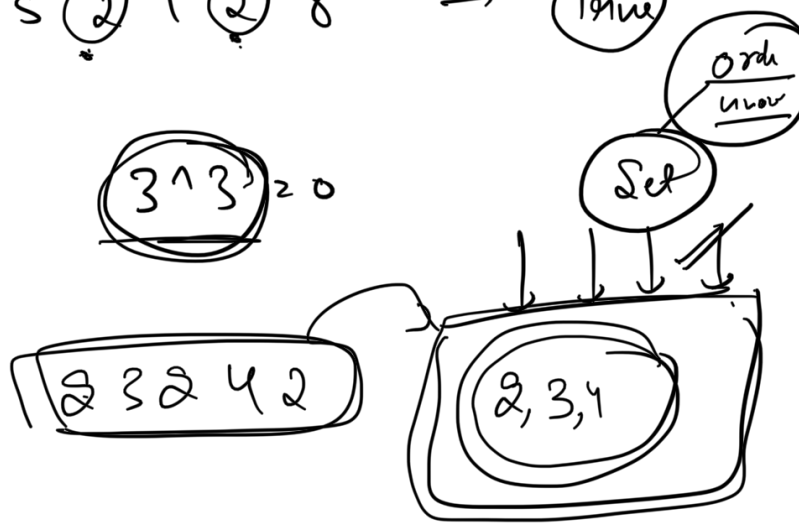
→ Question → we have been given a list we just need to figure out, In this array/list are there two numbers which are equal.

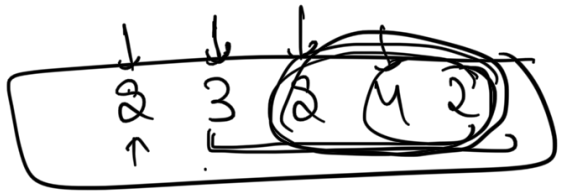
→ 5 2 1 6 7 → Palse

→ 5 (2) 1 (2) 8 → True

→ 3^3 = 0

Unordered_set
Ordered_set





(2,3) (2,2) (2,4) (2,2)

$\frac{n \times (n-1)}{2}$

(n-1) + (n-2) + n-3 + . . . 1

$= \frac{n^2 - n}{2} = O(n^2)$

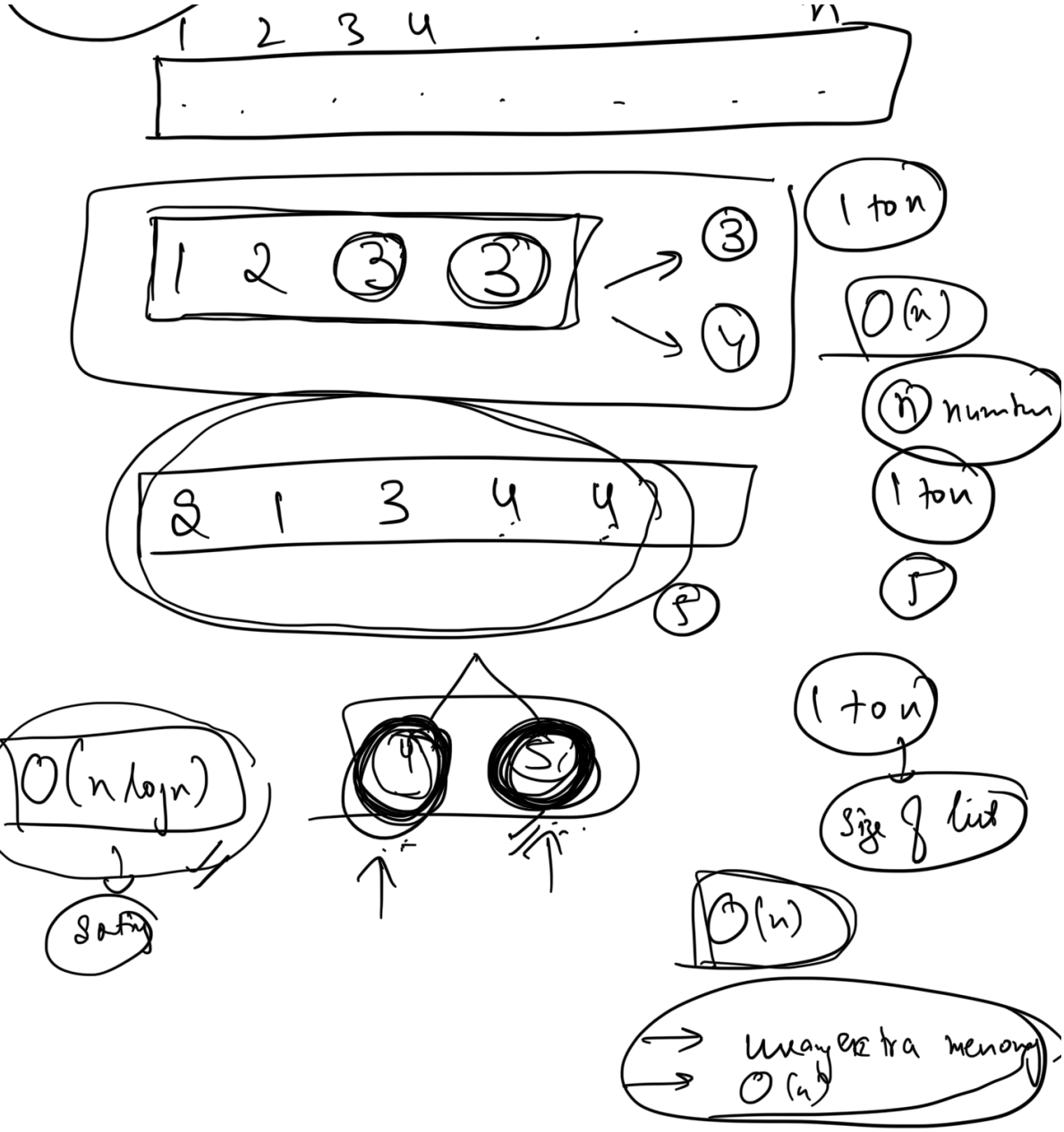
O(n log n)

Sorting is a good solution

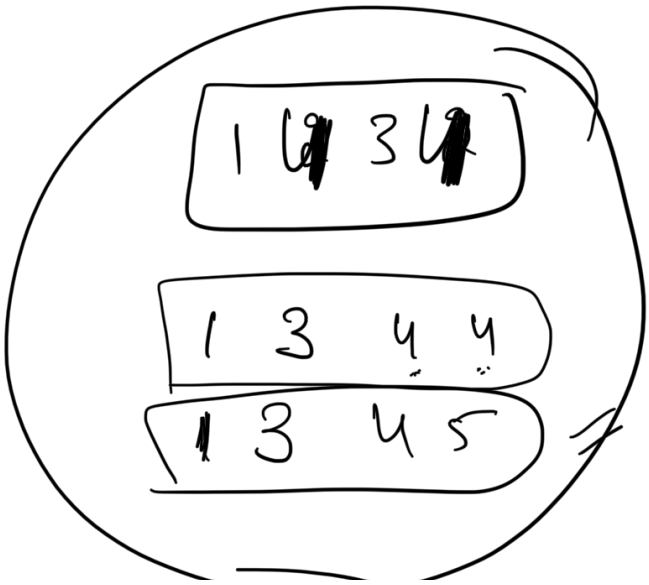
1 + 2 + 3 + . . . n-1

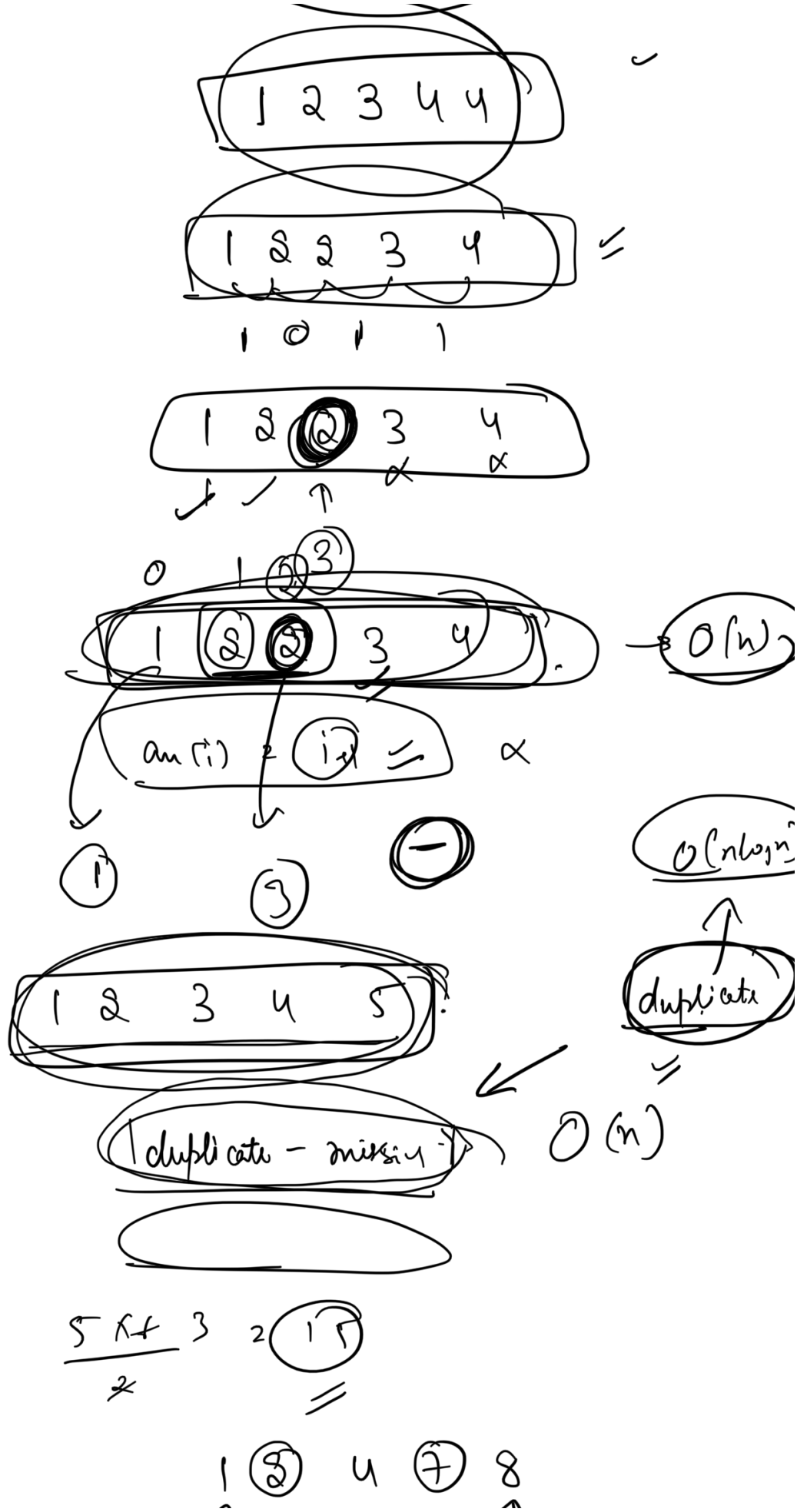
$\frac{(n-1) \times n}{2} = O(n^2) =$

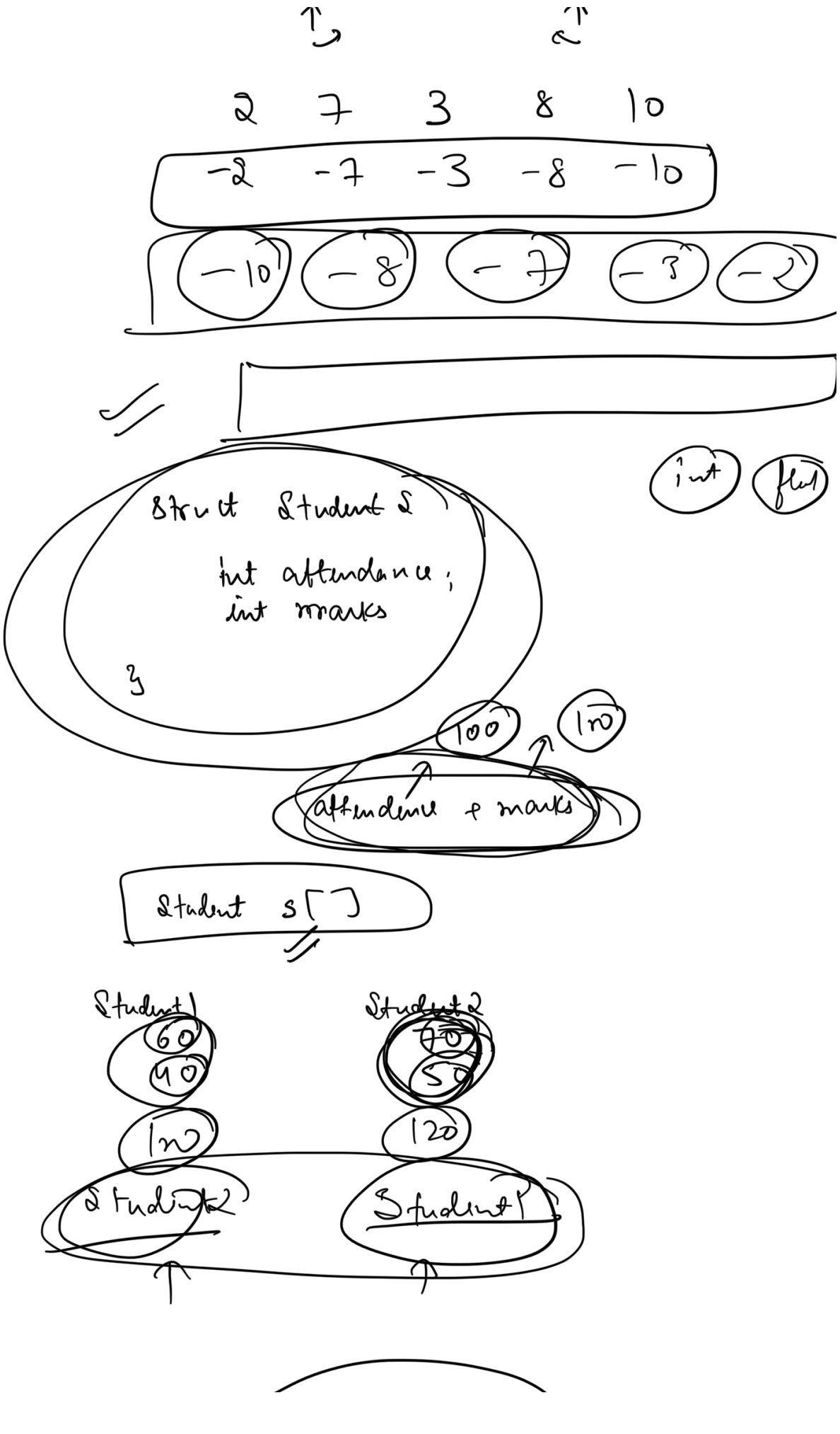
$O(n^2)$

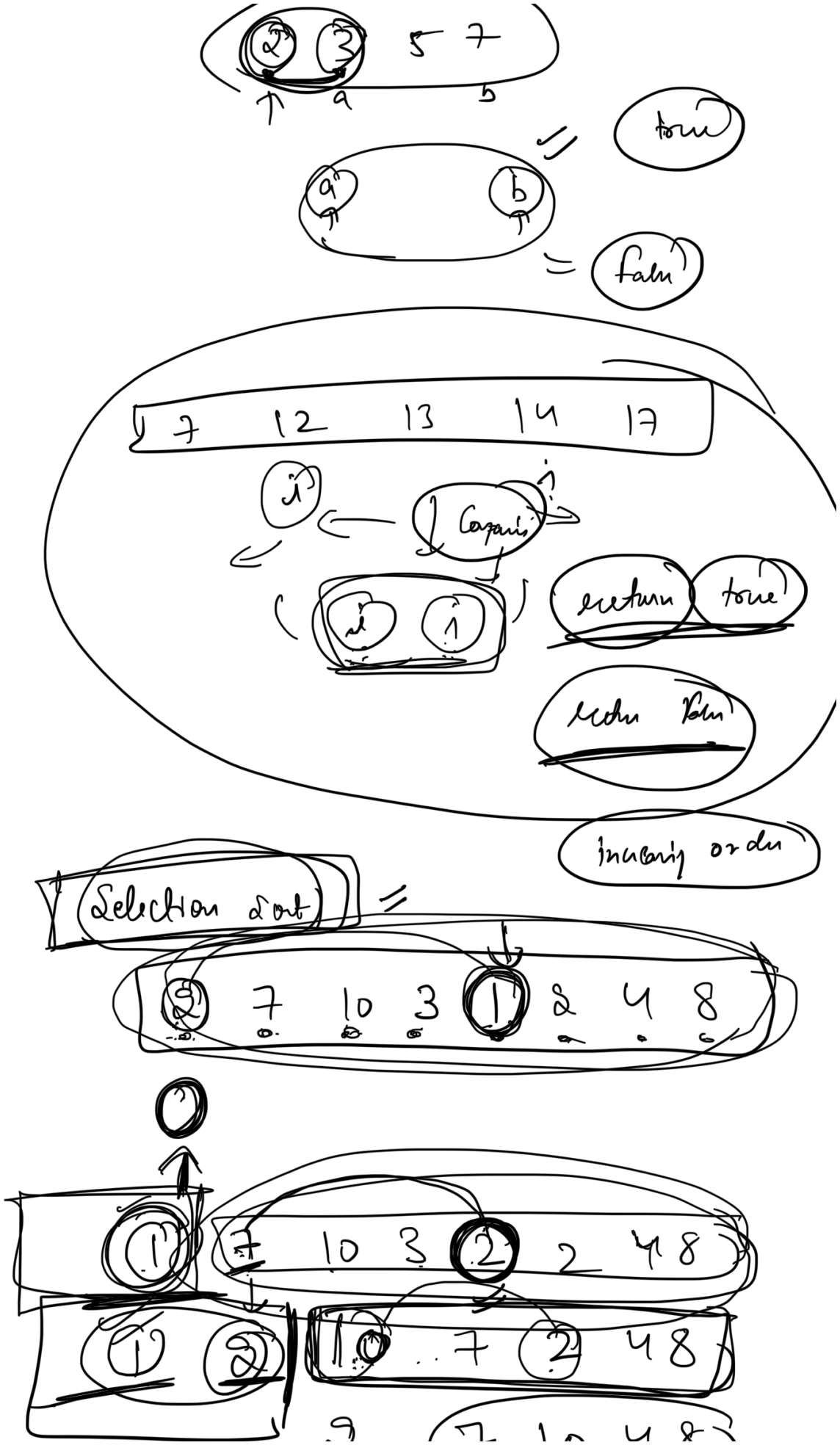

1 2 3 4 . . n
1 2 3 3
3
4
1 to n
O(n)
n number
1 to n
2 1 3 4 4
O(n log n)
sorting
1 to n
size of list
O(n)
use extra memory
O(n)

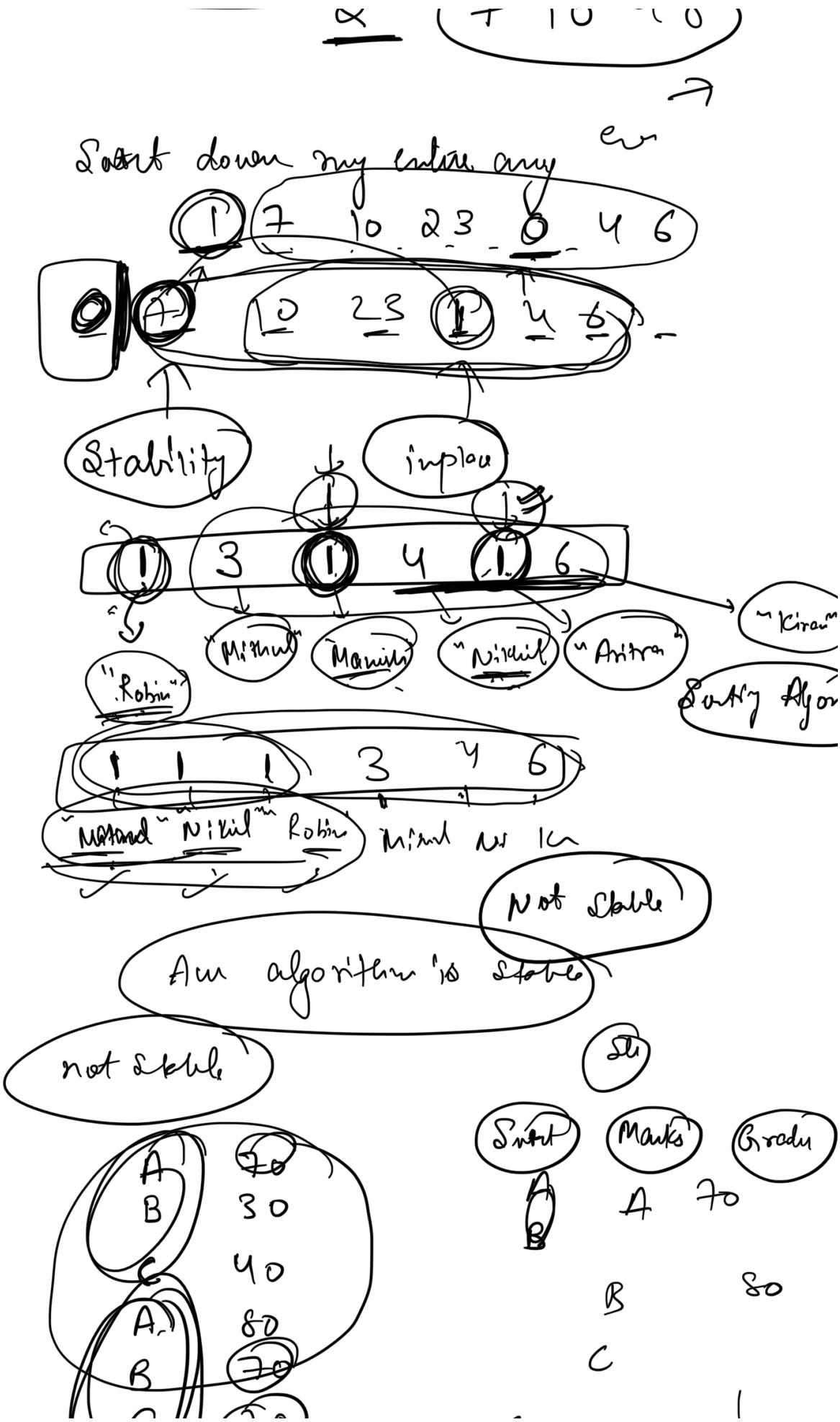
1 2 3

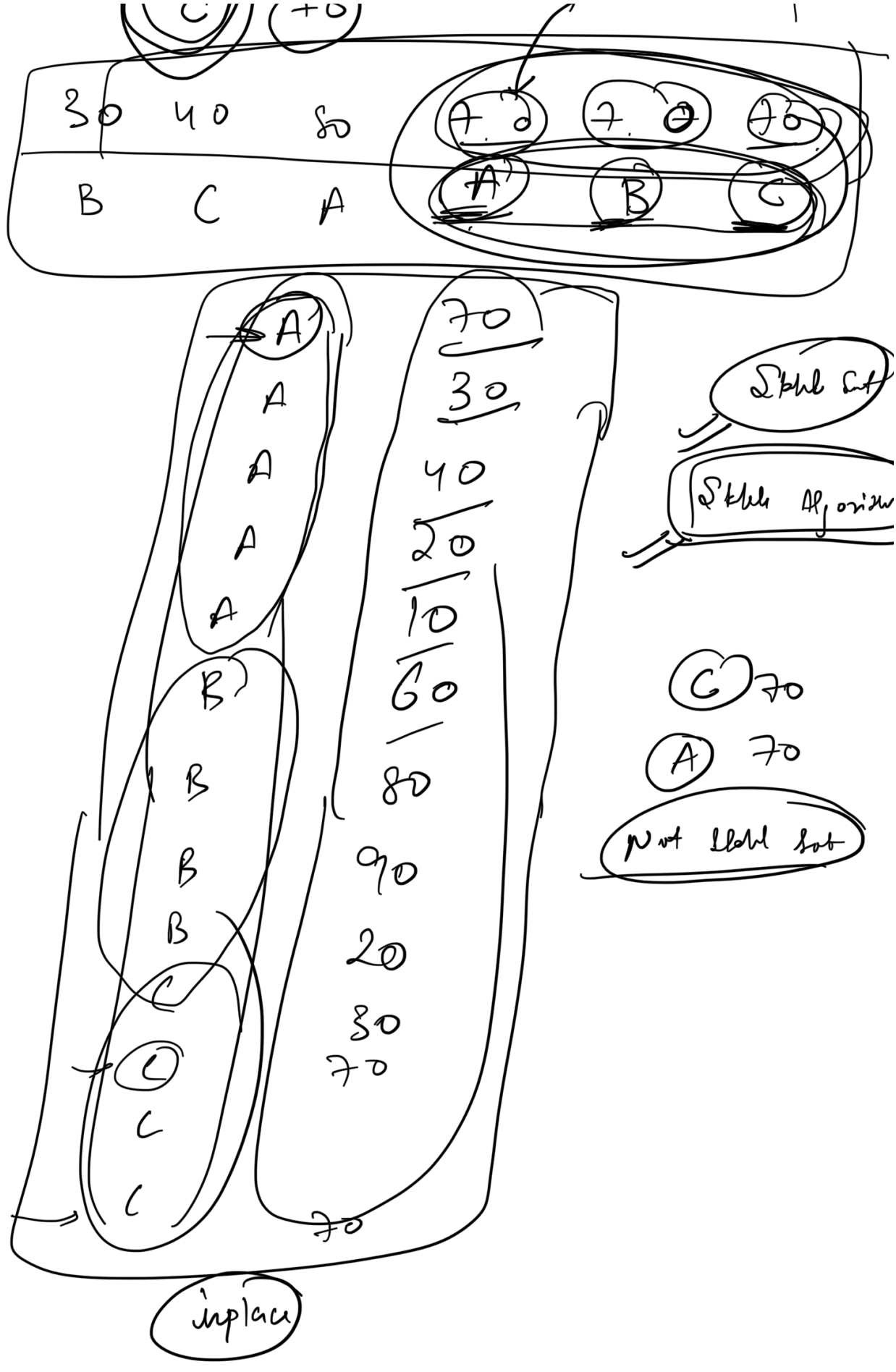

1 3 4
1 3 4 4
1 3 4 5


1 2 3 4 4
1 2 2 3 4
duplicate
duplicate - missing
O(n)
O(nlogn)

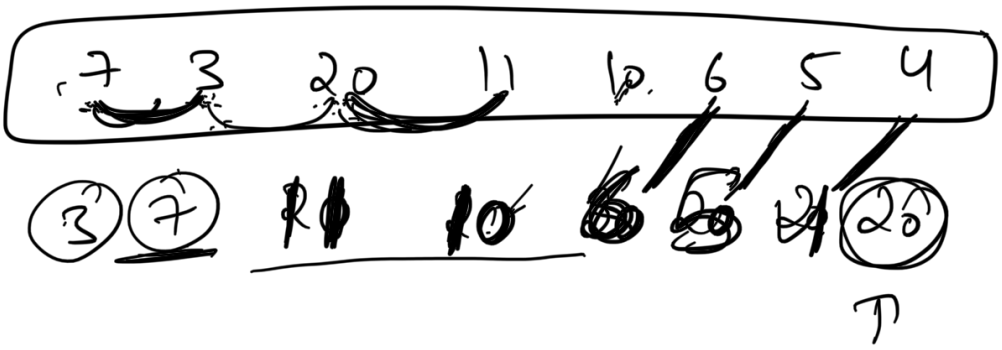

2 7 3 8 10
-2 -7 -3 -8 -10
-10 -8 -7 -3 -2
int
float
struct Student s
int attendance;
int marks
}
100
110
attendance + marks
Student s[]
Student1
60
40
110
Student2
70
50
120
Student2
Student1


2
3
5
7
a
b
true
a
b
false
7
12
13
14
17
i
Compare
a
1
return true
return false
increasing order
Selection sort
9
7
10
3
1
2
4
8
1
7
10
3
2
2
4
8
1
2
10
7
2
4
8

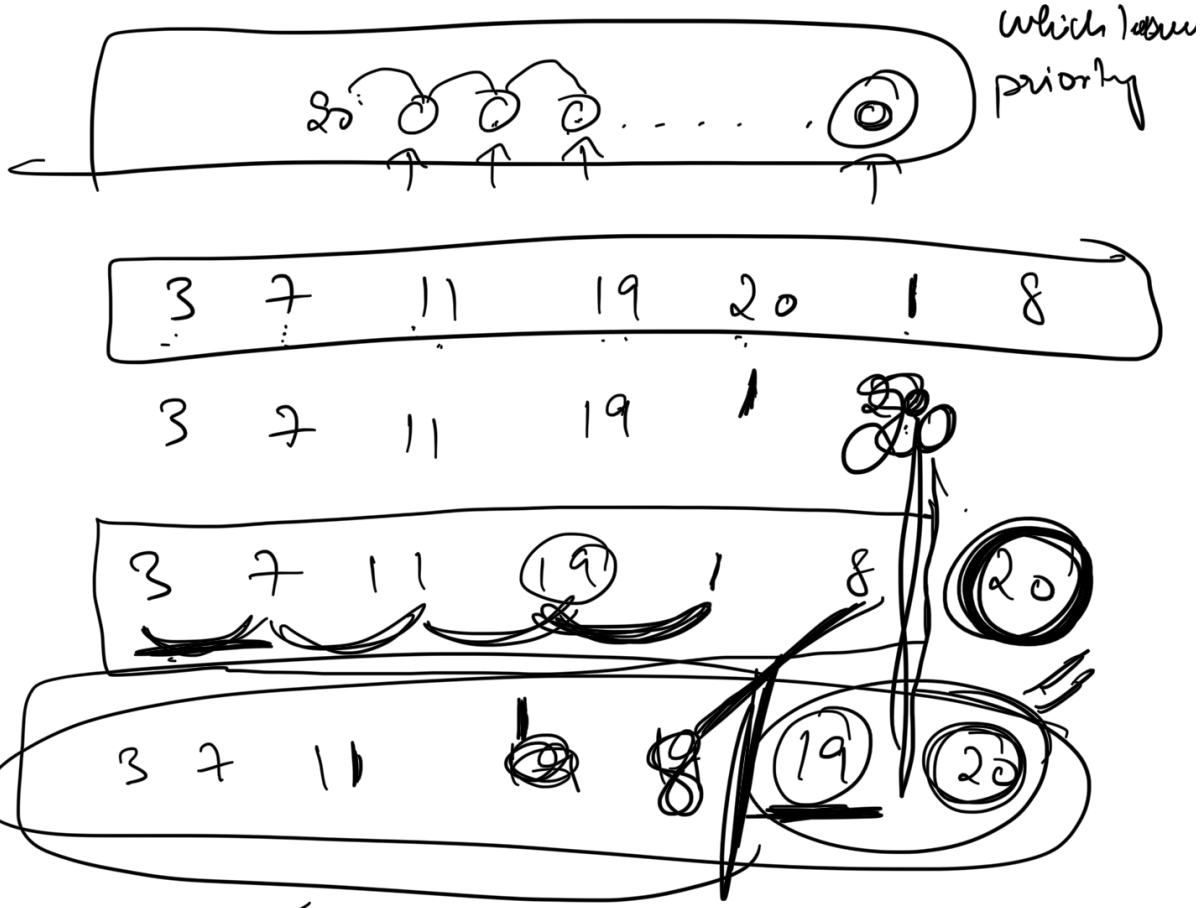

Sort down my entire array
Stability
inplace
"Robin"
Not Stable
An algorithm is stable
not stable
Sr
Sort
Marks
Grade
A 70
B 80
C


30 40 80
B C A
70 70 70
A B C
A
A
A
A
A
70
30
40
20
10
60
80
90
20
30
70
70
B
B
B
B
C
C
C
C
inplace
Stable Sort
Stable Algorithm
C 70
A 70
Not Stable Sort

Bubble Sort

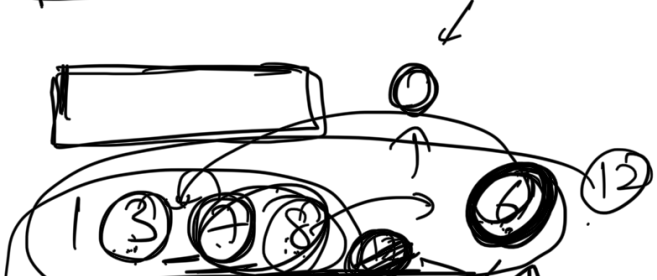


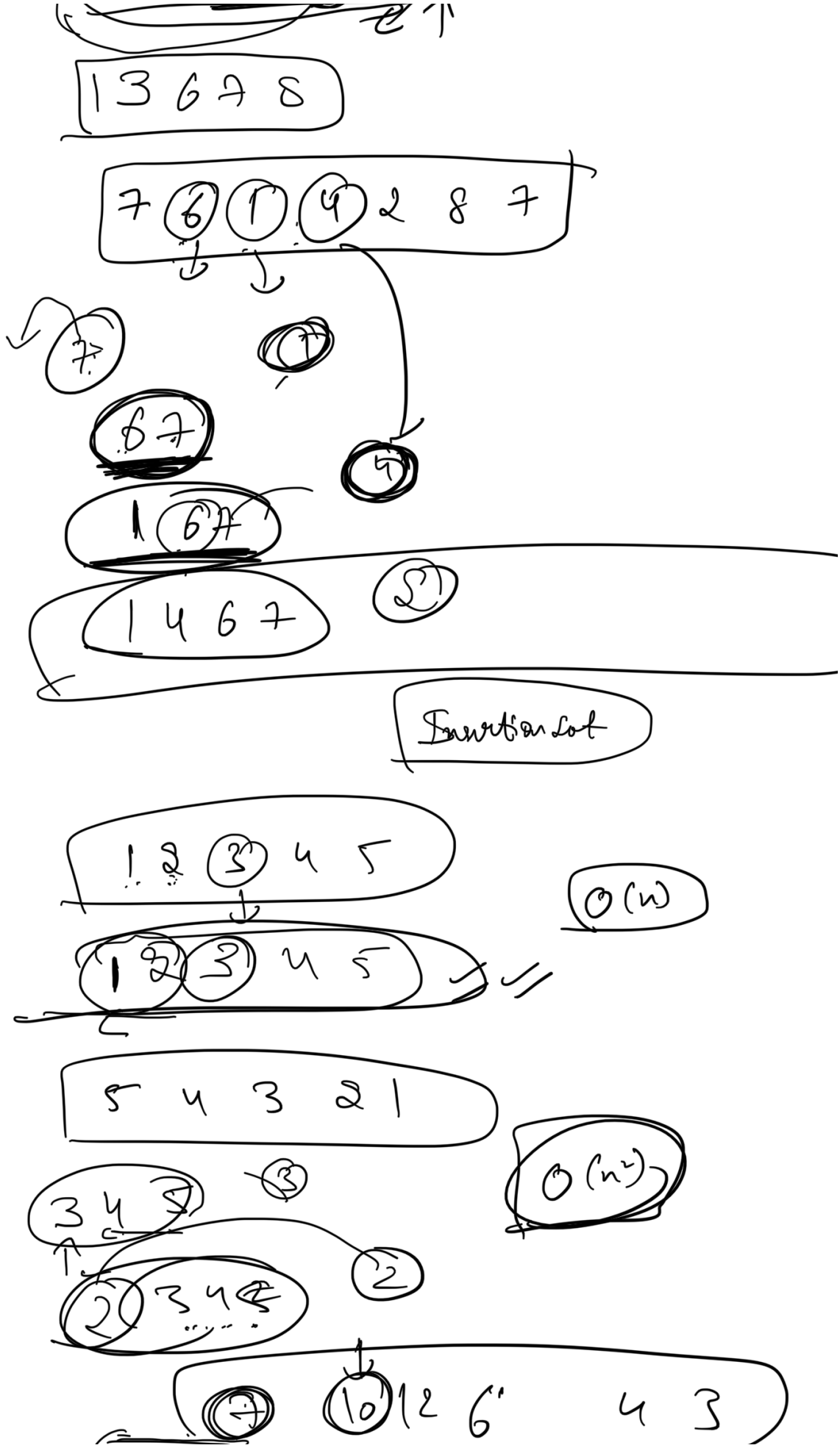
which have more priority

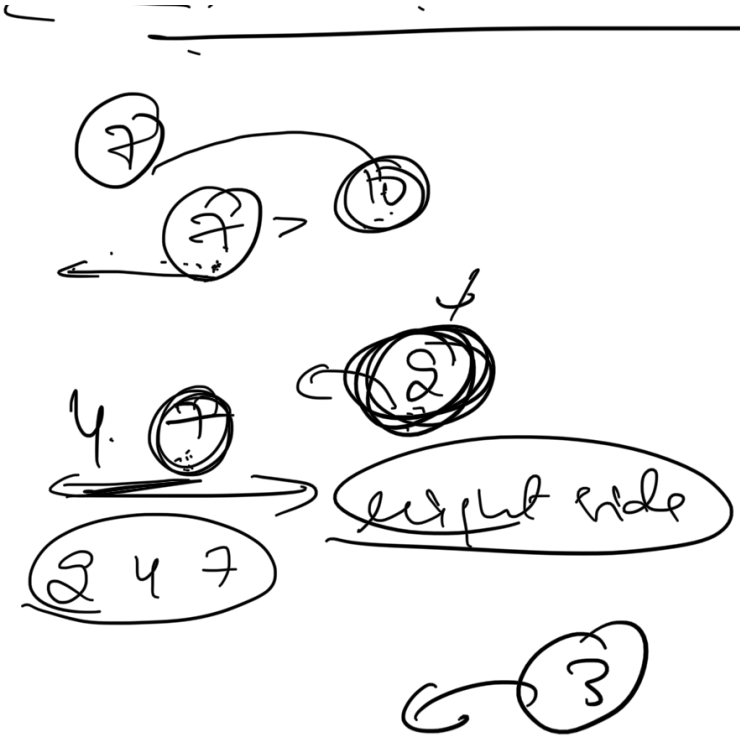


Sorted our entire list

Insertion Sort




1 3 6 7 8
7 6 1 4 2 8 7
Insertion Sort
1 2 3 4 5
O(n)
5 4 3 2 1
O(n²)



2 4 7