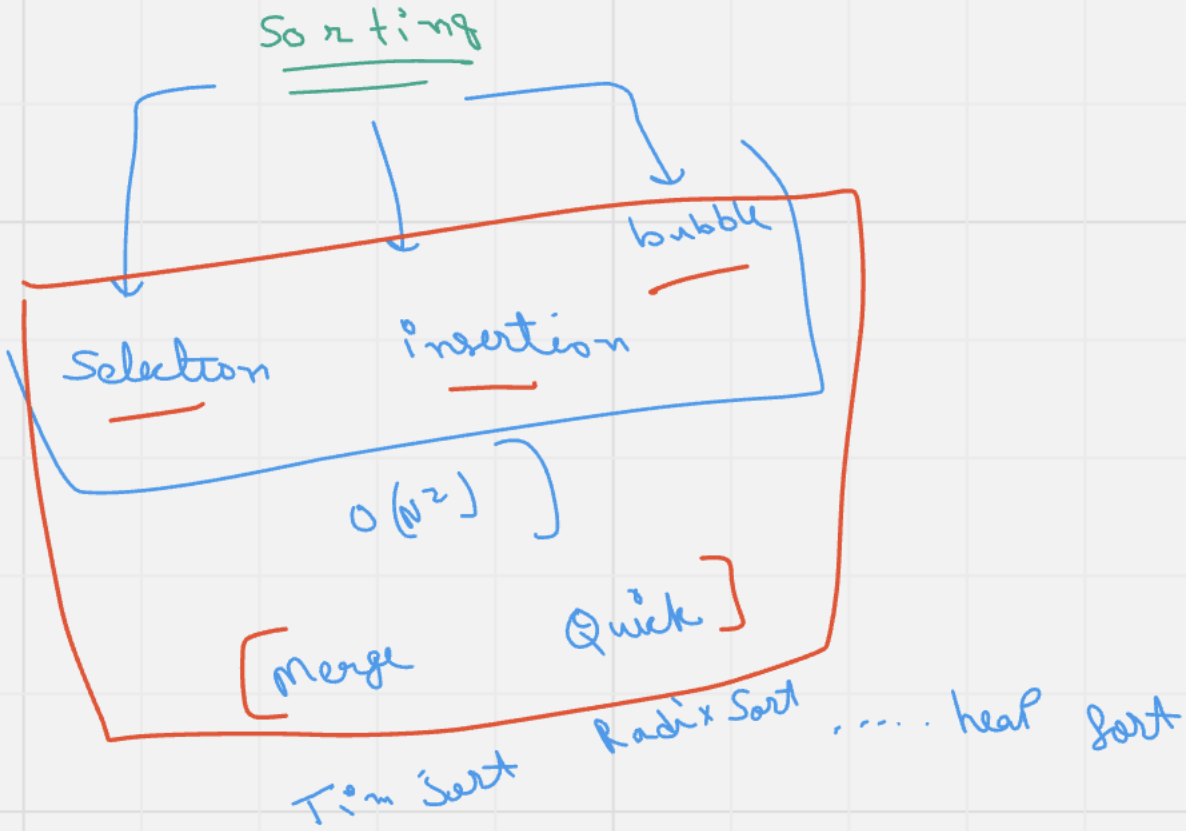
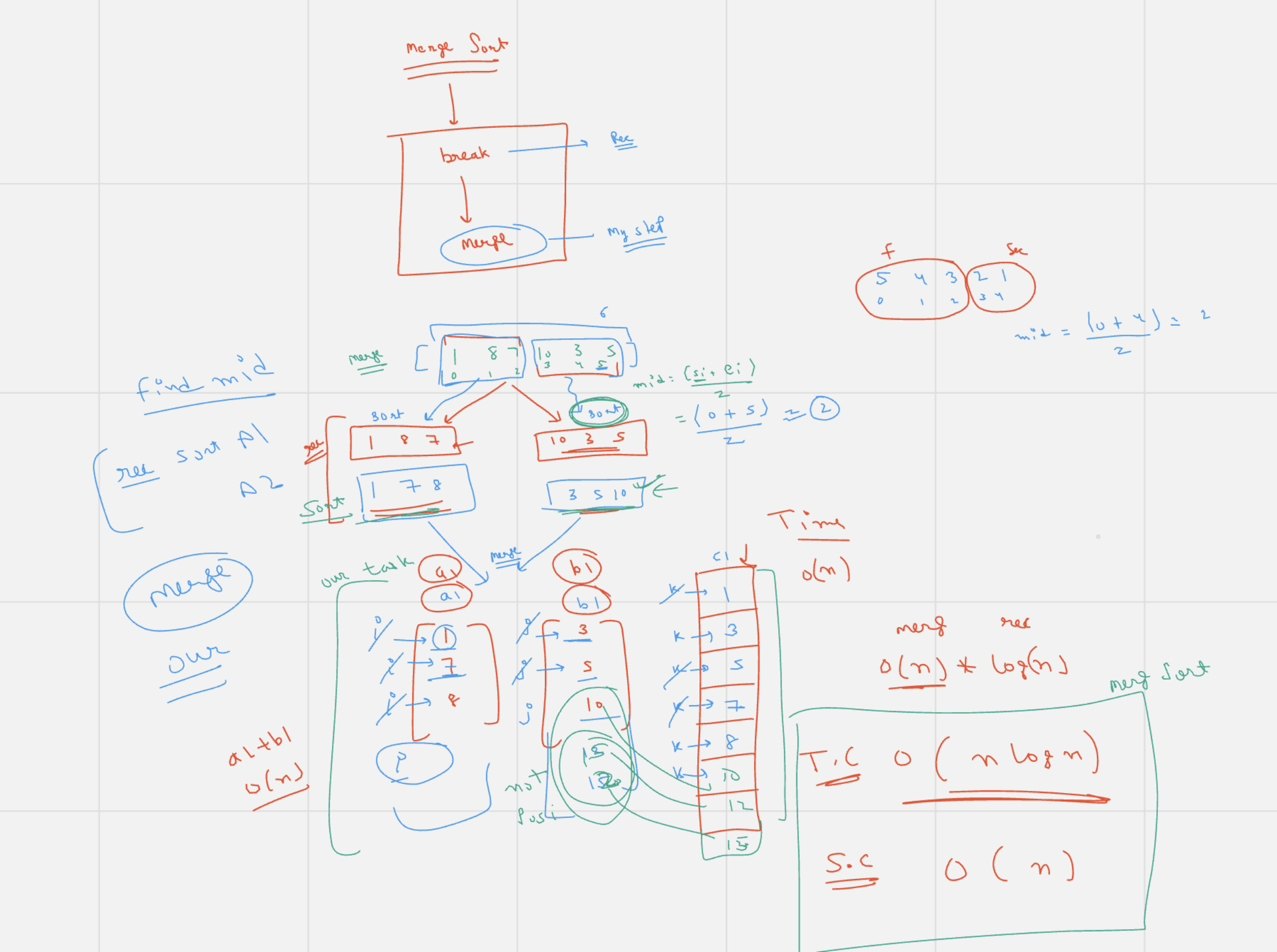
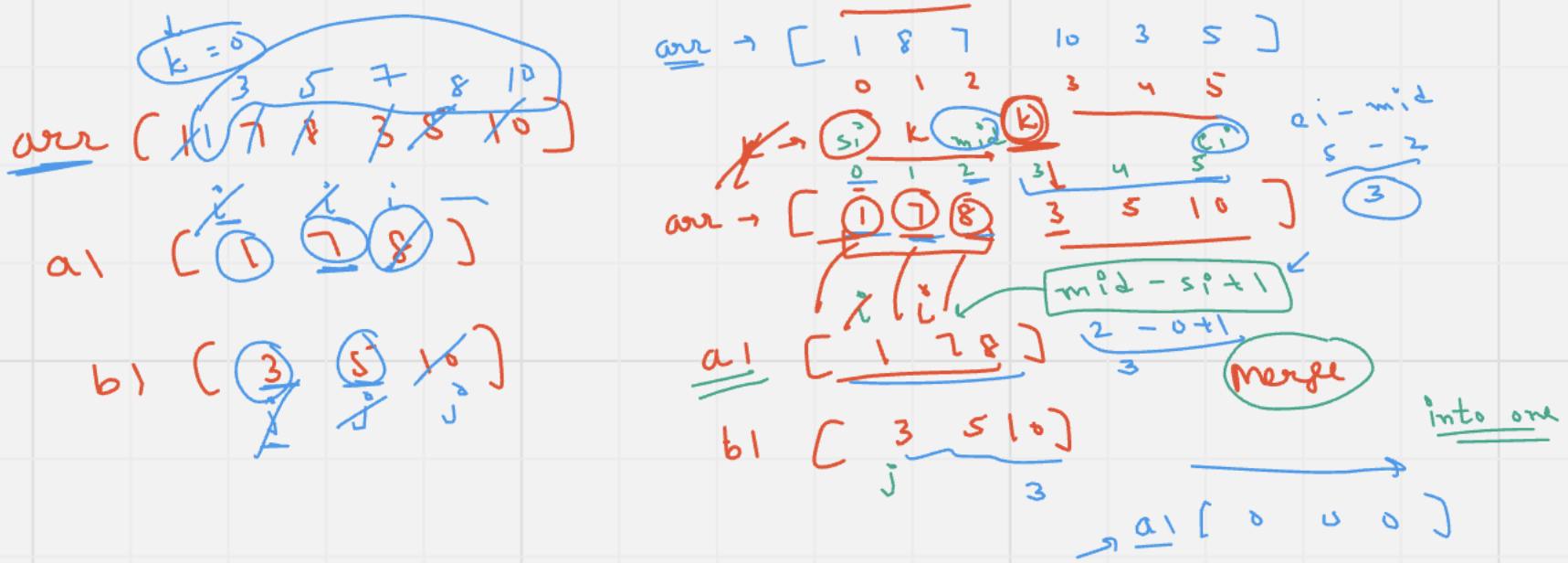
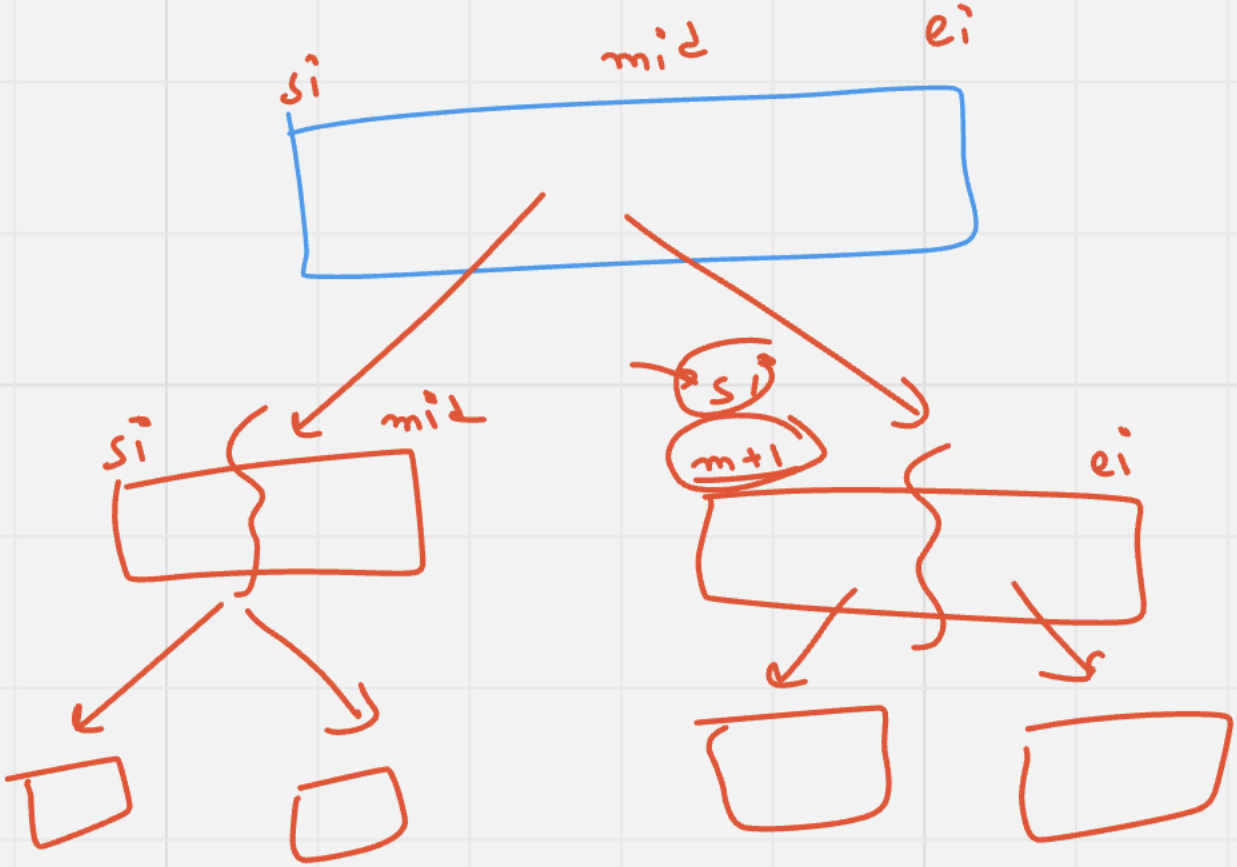
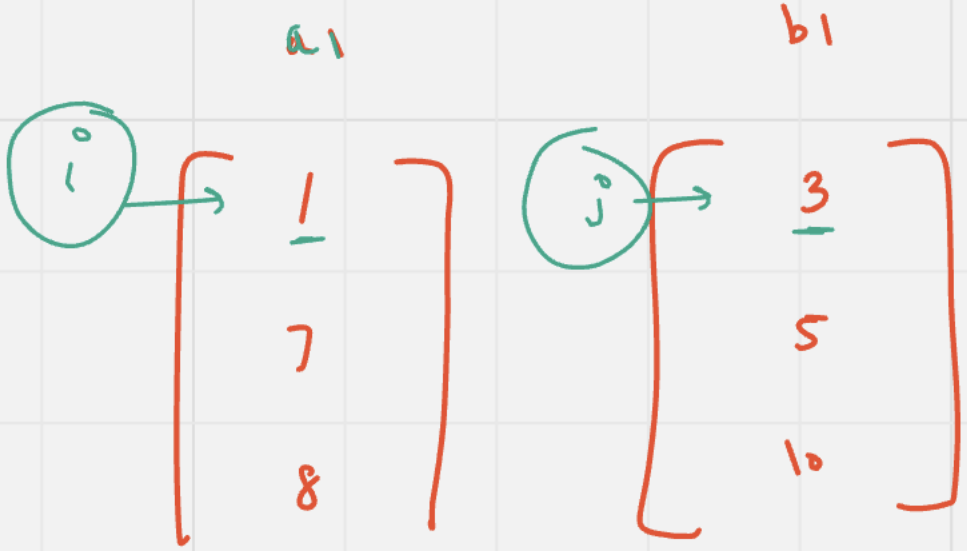
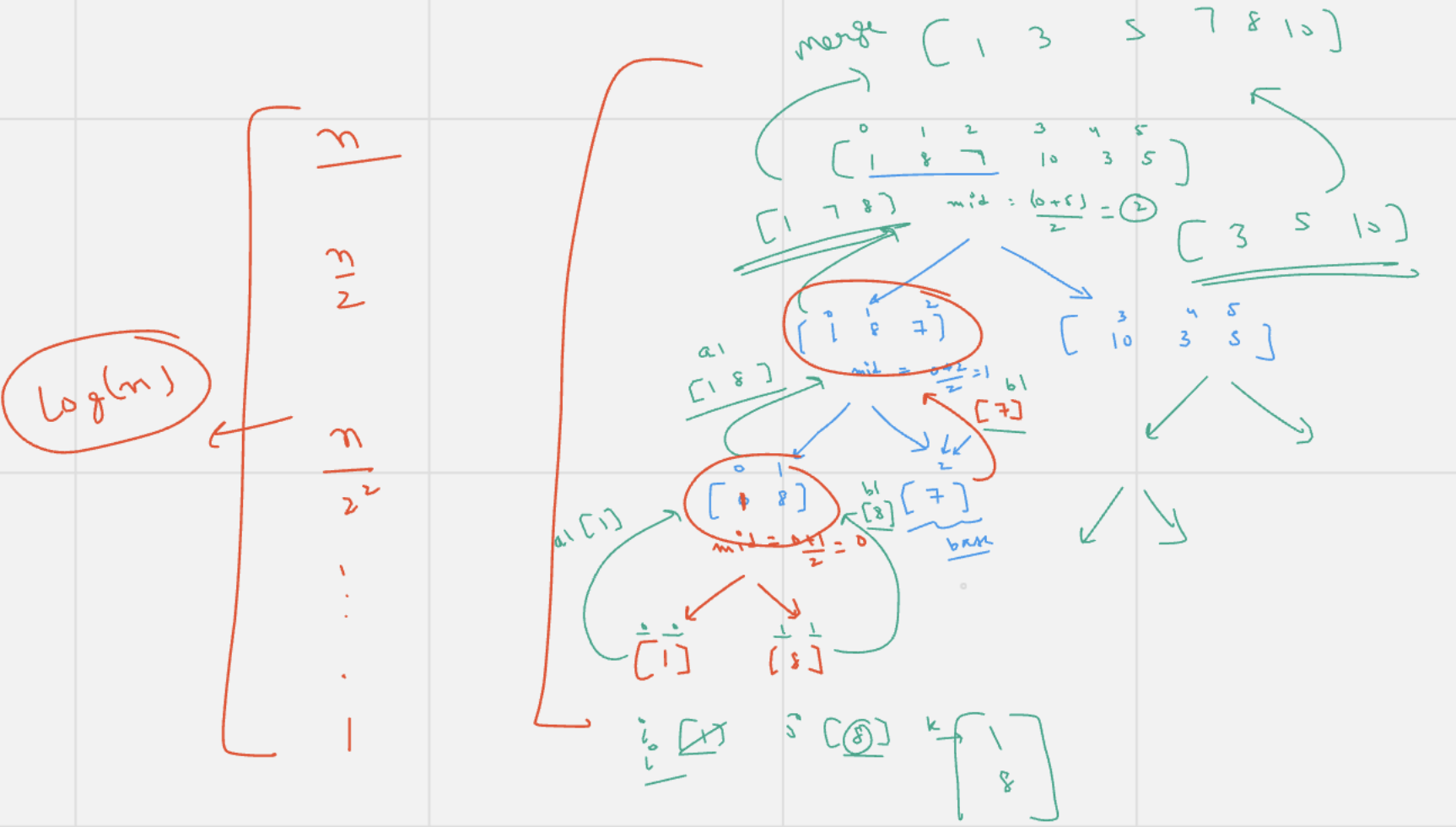

Sorting
bubble
Selection
insertion
O(N²)
Merge
Quick
Tim Sort
Radix Sort
.... heap Sort


Merge Sort
break
Rec
merge
my self
f
Sec
5 4 3
2 1
0 1 2
3 4
mid = (0 + 4) / 2 = 2
find mid
rec sort A1
A2
merge
our
merge
sort
sort
rec
1 8 7
10 3 5
0 1 2
3 4 5
mid = (si + ei) / 2
= (0 + 5) / 2 = 2
1 7 8
3 5 10
Sort
our task
merge
a1
b1
Time
O(n)
c1
merge rec
O(n) * log(n)
merge Sort
T.C O(n log n)
S.C O(n)
a1+b1
O(n)
not
posi


k = 0
3 5 7 8 10
arr [1 7 8 3 8 10]
a1 [1 7 8]
b1 [3 5 10]
arr → [1 8 7 10 3 5]
0 1 2 3 4 5
si
k
mid
k
ei
ei - mid
5 - 2
3
arr → [1 7 8 3 5 10]
mid - si + 1
2 - 0 + 1
3
a1 [1 7 8]
merge
b1 [3 5 10]
j
3
into one
a1 [0 0 0]


si
mid
ei
si
mid
s
m+1
ei


a1
b1
i
0
1
1
7
8
j
0
3
5
10

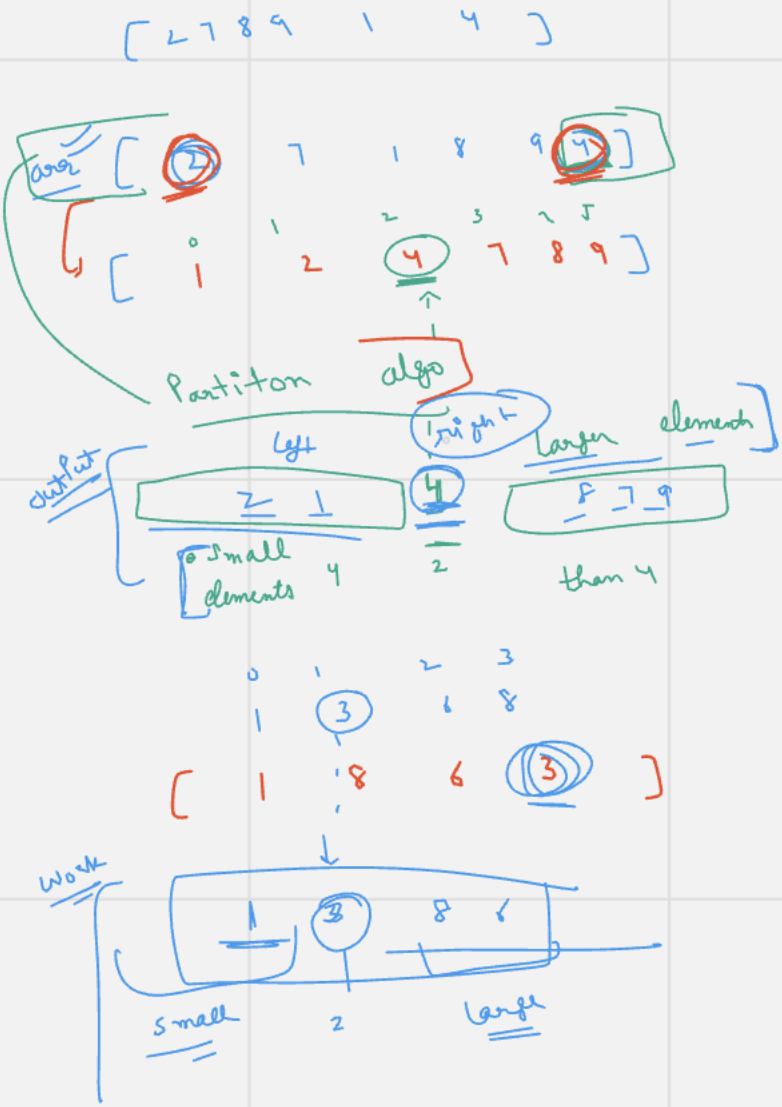

log(n)
n
n/2
n/2²
1
merge
[1 3 5 7 8 10]
[1 8 7 10 3 5]
mid : (0+5)/2 = 2
[1 7 8]
[3 5 10]
[1 8 7]
mid = 0+2/2 = 1
[10 3 5]
a1
[1 8]
b1
[7]
[1 8]
mid = 0+1/2 = 0
a1 [1]
b1 [8]
[7]
bmm
[1]
[8]
i
j
k
[1 8]

write Quick sort code

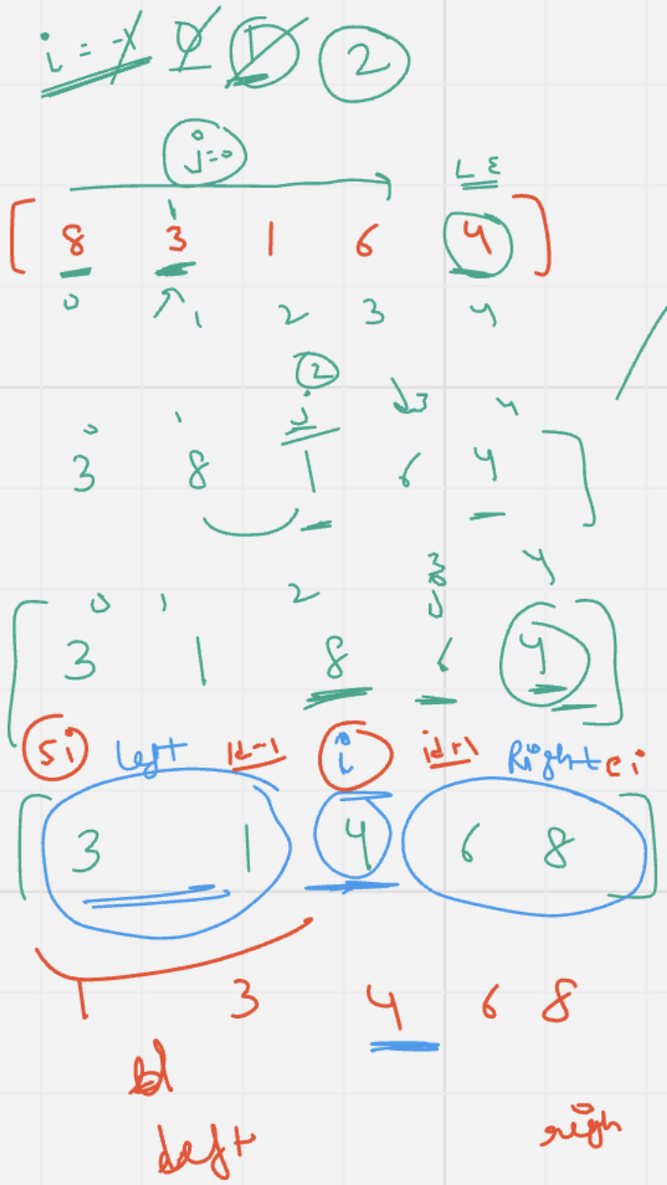
Quick Sort

partition algo

[rest code is super easy]



if j smaller than lastEle:
i++
swap i, j
.....
i++
swap i and lastEle



will not Sort your array