Seraching&Sorting

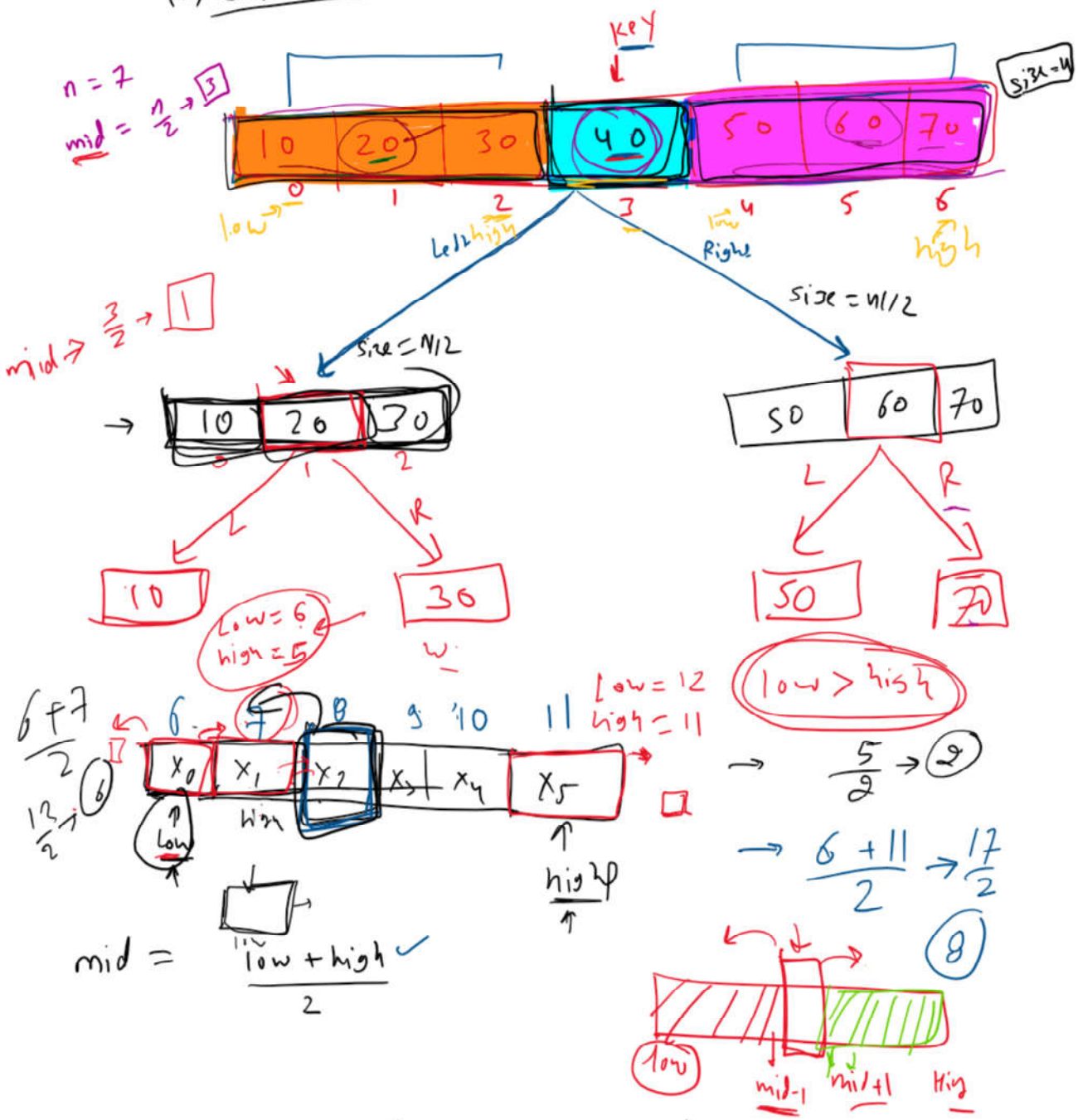
Wednesday, September 1, 2021 3:33 PM

Divide & Conquer

↳ Break problem into similar sub problems then solve them.

Binary Seach

(i) Sorted Array




```
BinarySearch (arr, low, high, key)
    if low > high:
        return -1
    mid = (low + high) // 2
    if arr[mid] > key:
        return BinarySearch(arr, low, mid-1, key)
```

To

return BinarySearch(arr, Low, mid-1, key)

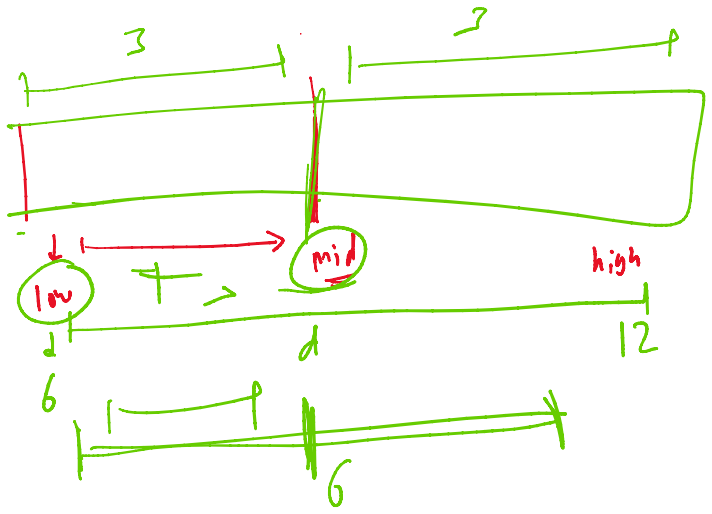
elif arr[mid] < key:

return BinarySearch(arr, mid+1, high, key)

else:

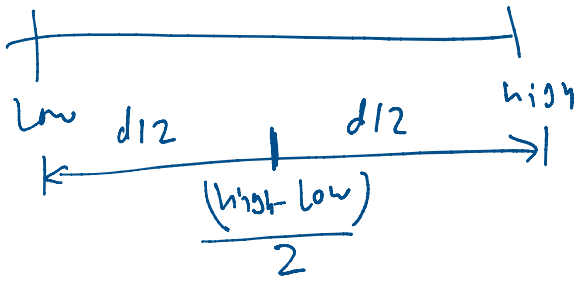
return mid

mid = low + (high - low) // 2



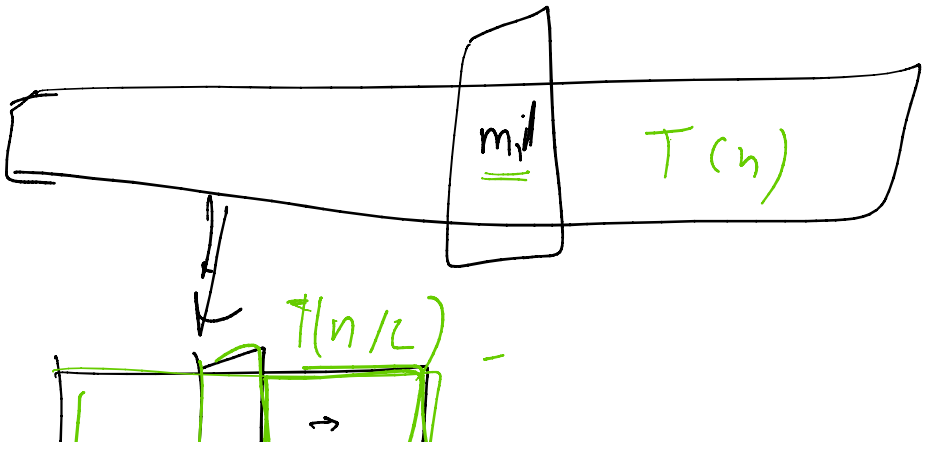
= 6 + (6) // 2

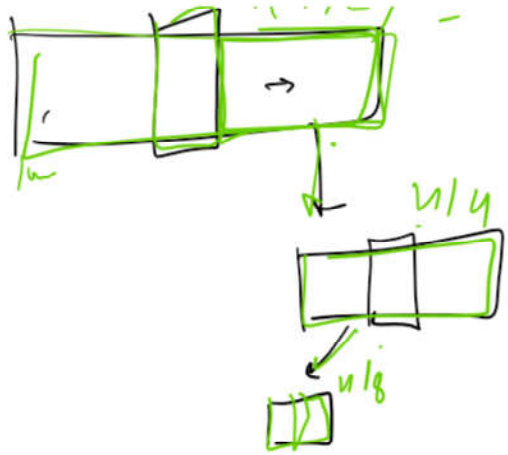
= 6 + 3 → 9



mid = high - d/2

mid = low + d/2

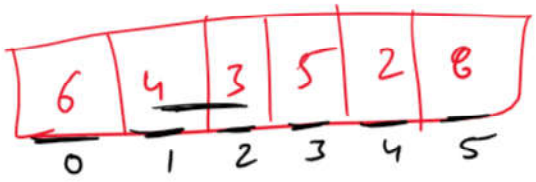




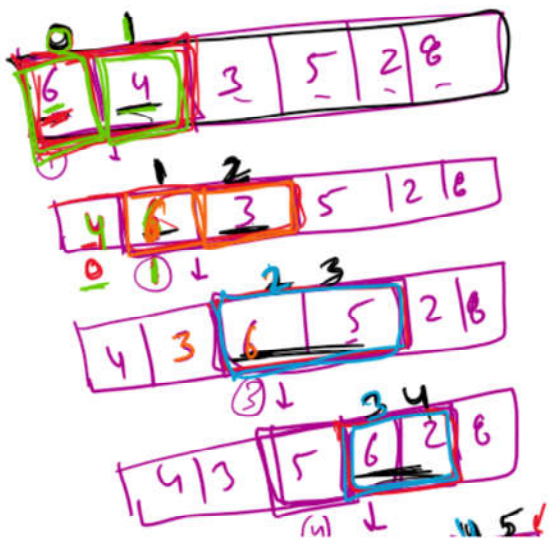
⇒ Sorting Algorithms

1. Bubble sort
2. Selection sort
3. Insertion sort
4. Quick sort
5. merge sort

* Bubble Sort ⇒ (Consecutive Elements are compared)



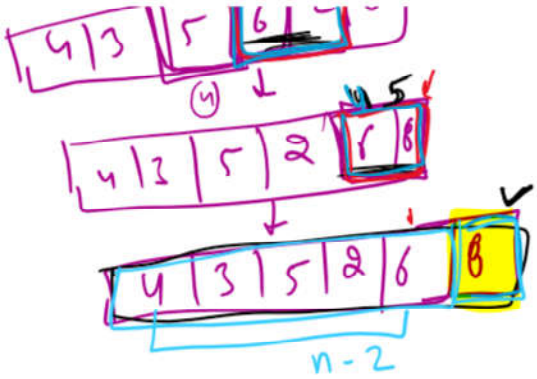
pass - 1



n = 6
n-2 ⇒ 4

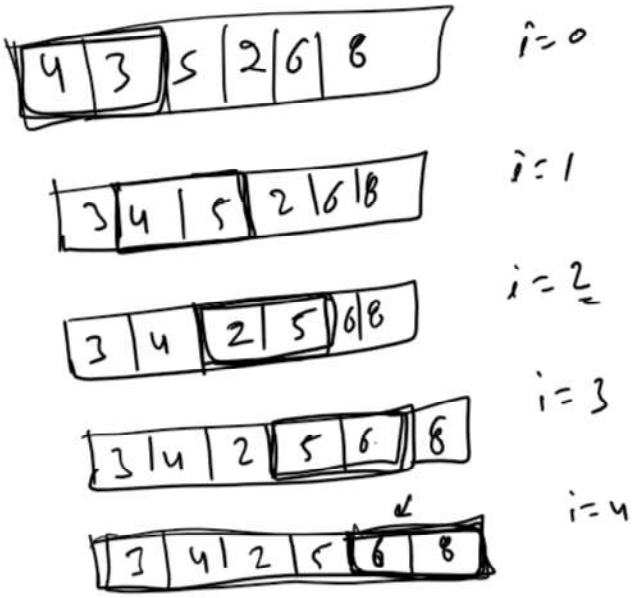
Bubble_pass(arr, n)

i = 0
while i <= n-2
Do
if arr[i] > arr[i+1]:
swap
t = arr[i] #6
arr[i] = arr[i+1]
arr[i+1] = t
i = i+1



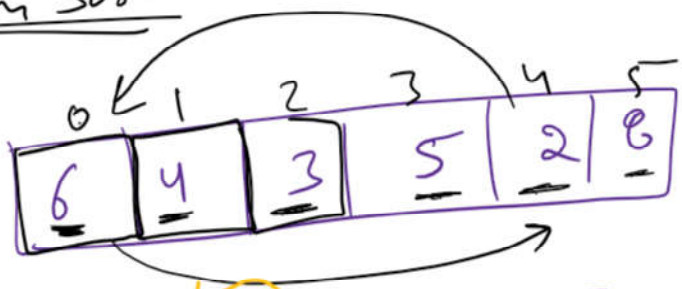
i = i + 1
return arr

2 - pass

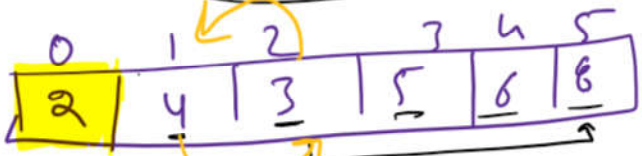


⇒ Selection Sort

arr →

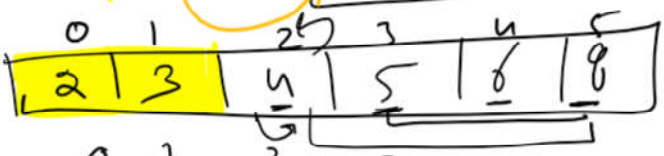


1 - pass →



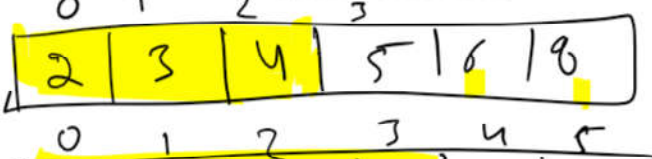
min = 2
loc = 4
i = 0

2 - pass →

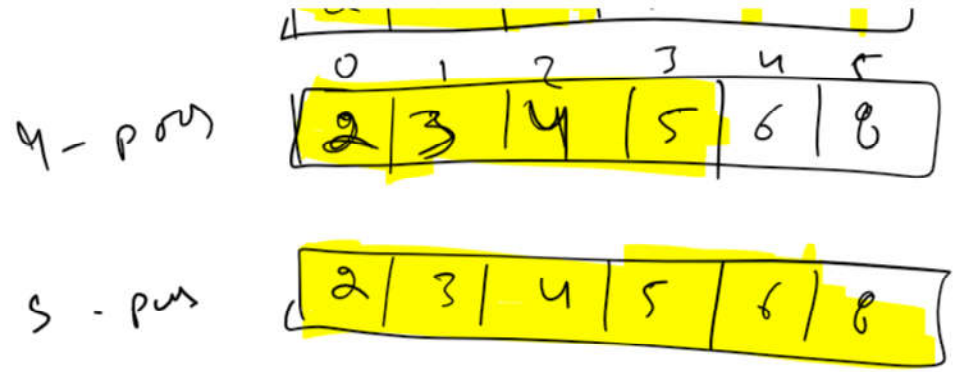


i = 1
min = 3
loc = 2

3 - pass →



i = 2
min = 4

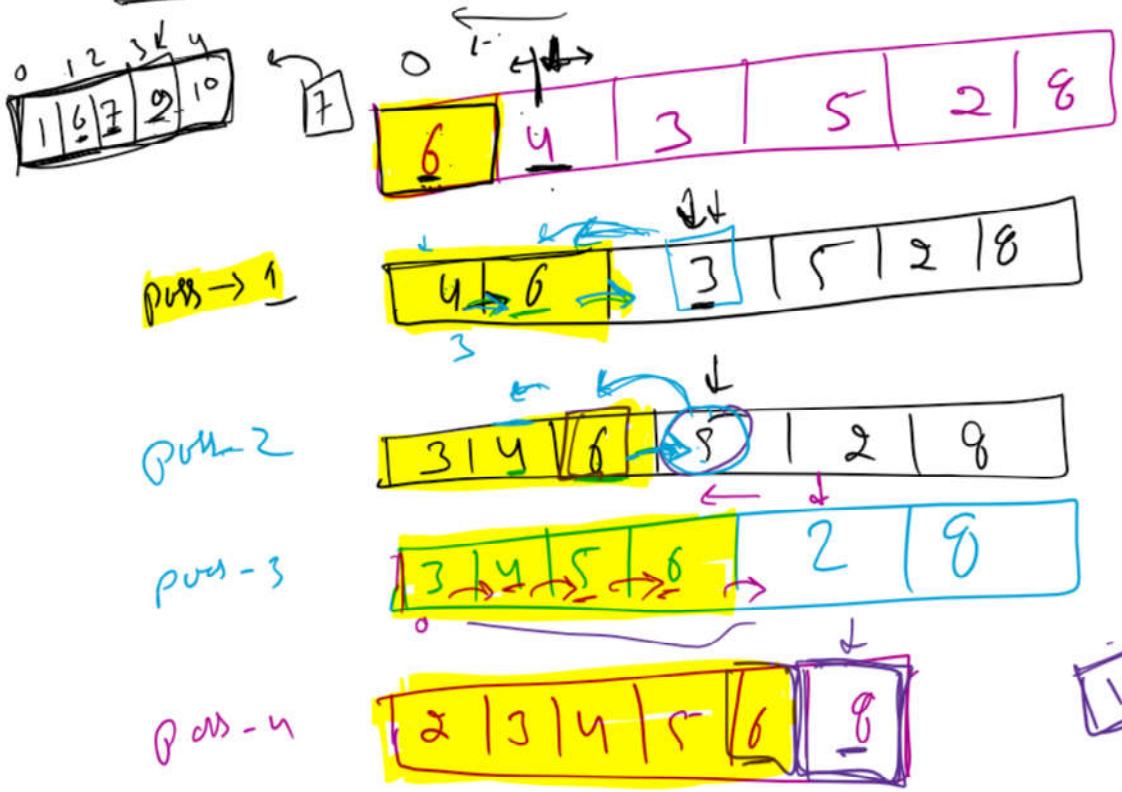


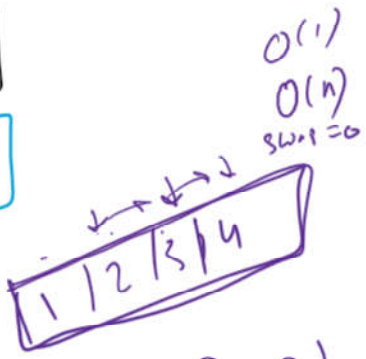
i=2
min=4
loc=2

i=3
min=5
loc=3

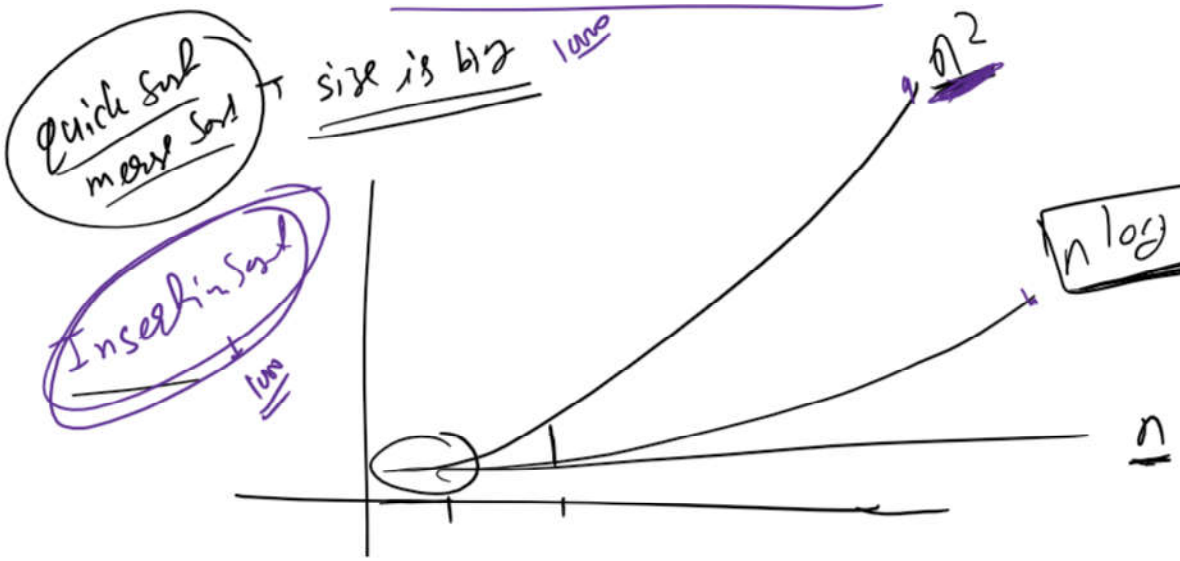
i=4
min=6
loc

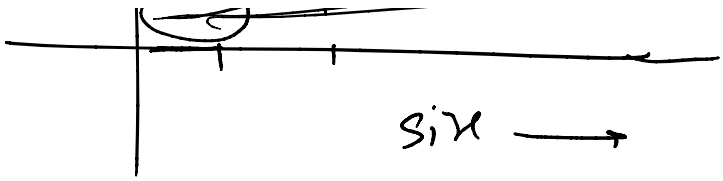
⇒ Insertion Sort





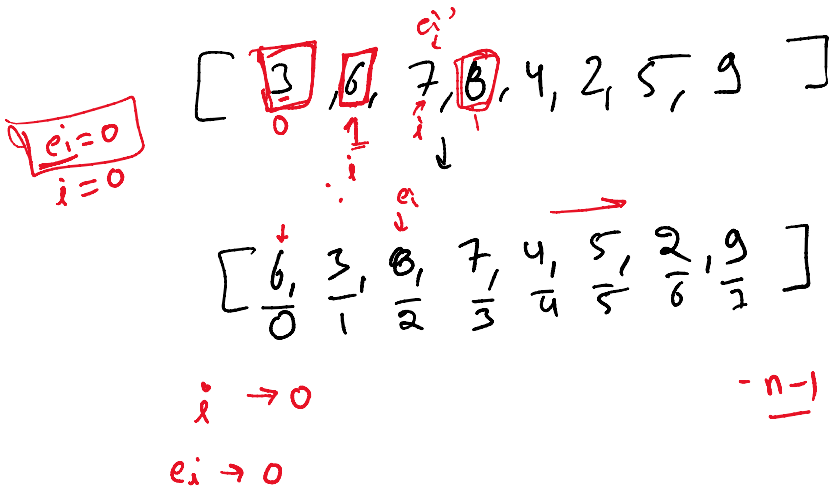
O(n²)



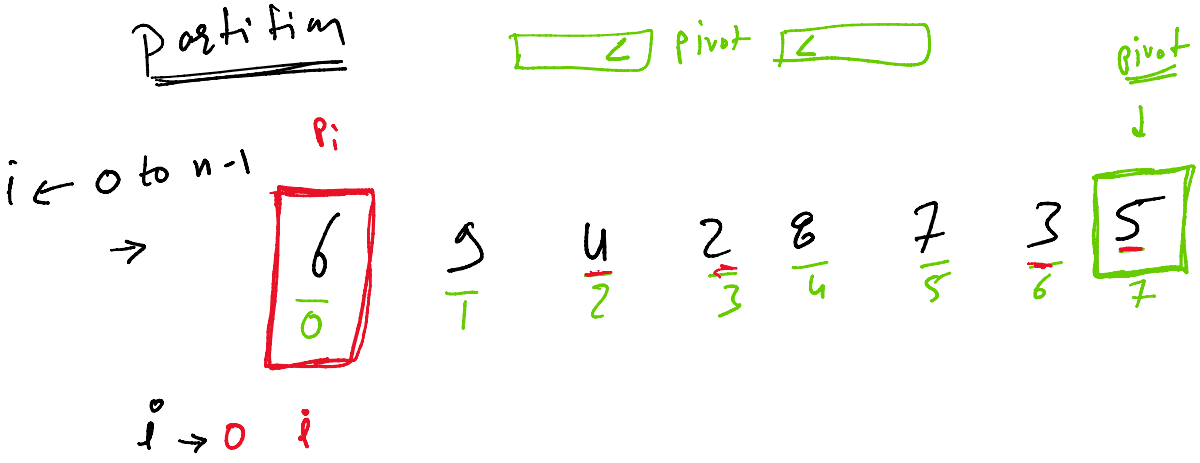


Quick Sort

lomuto partition scheme



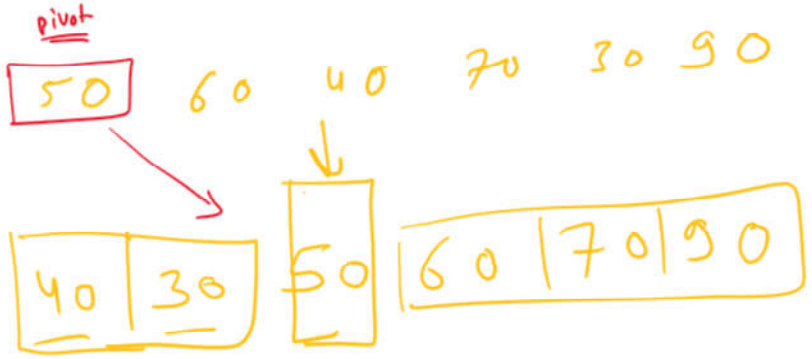
partition



$P_i \rightarrow 0$

Quick Sort → Divide & Conquer

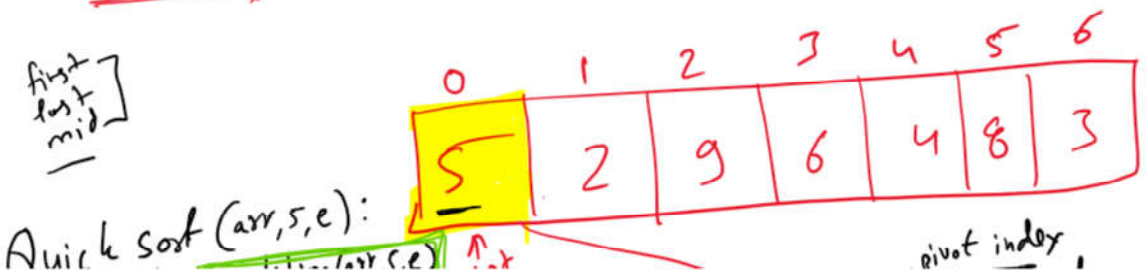
① Partition

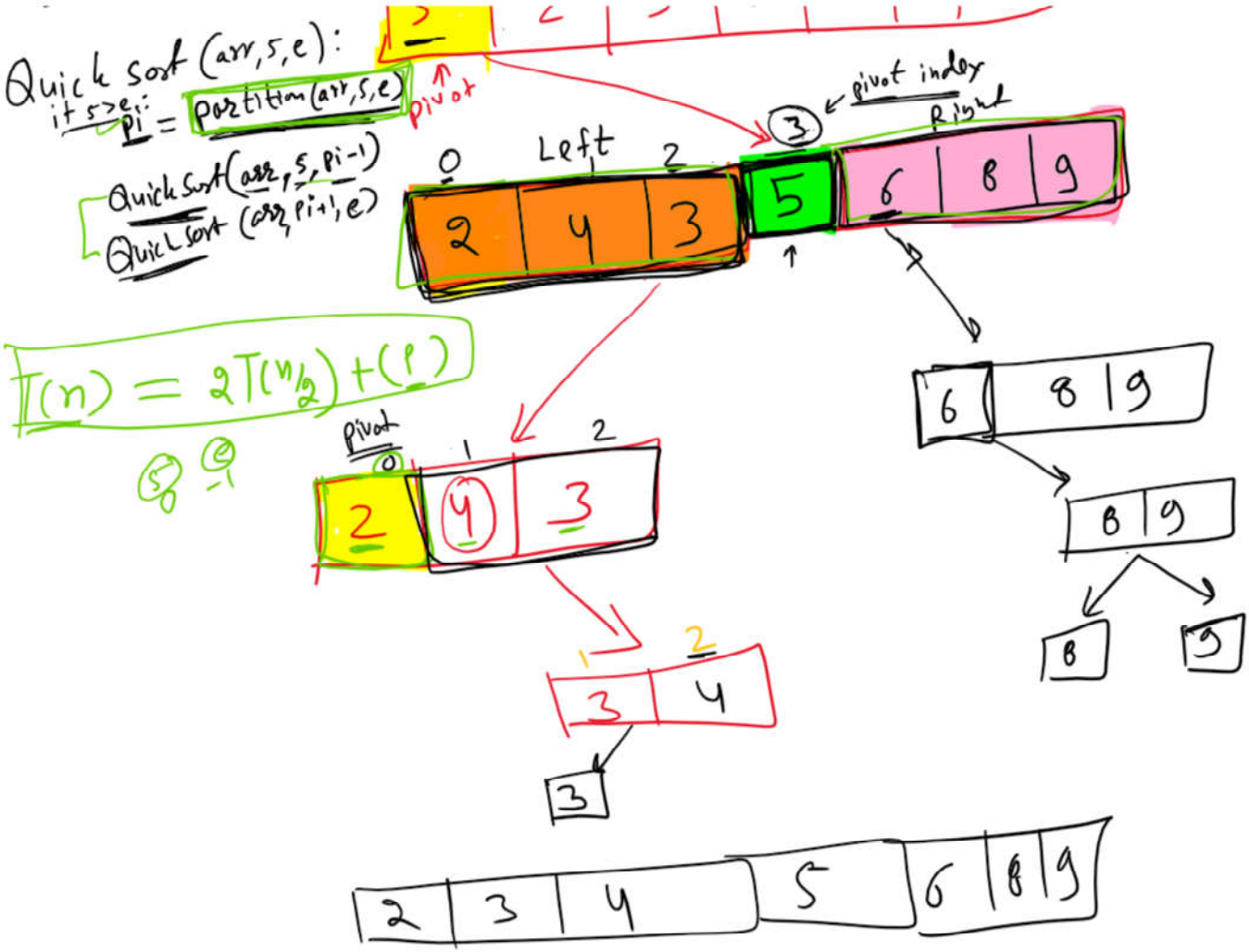


→ Divide & Conquer

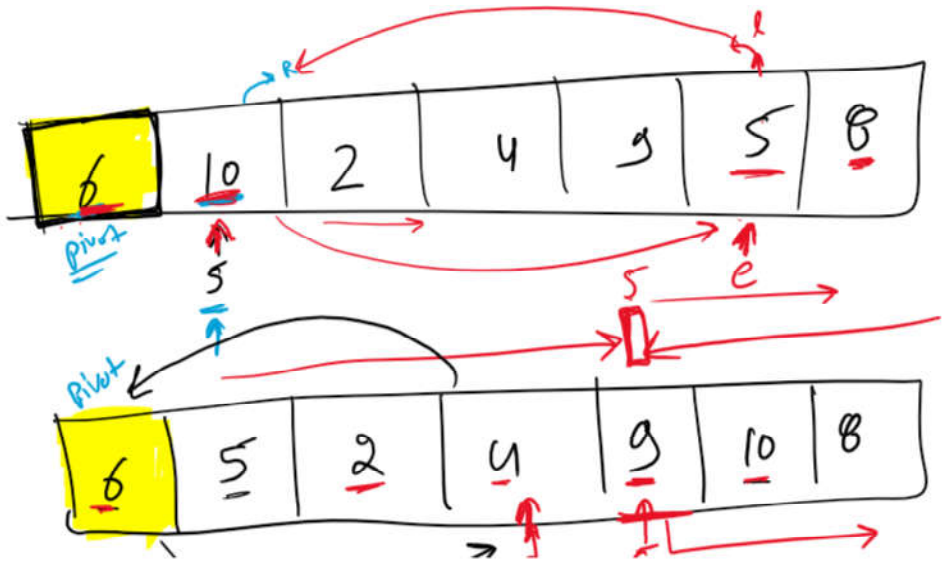
Quick Sort

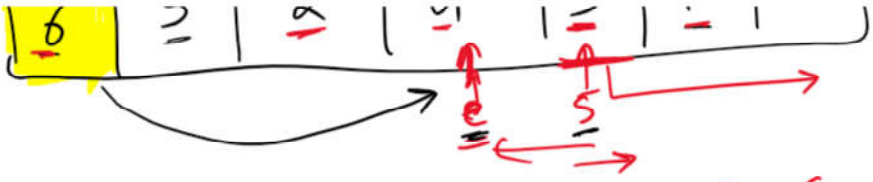
T ⇒ put pivot element in its position

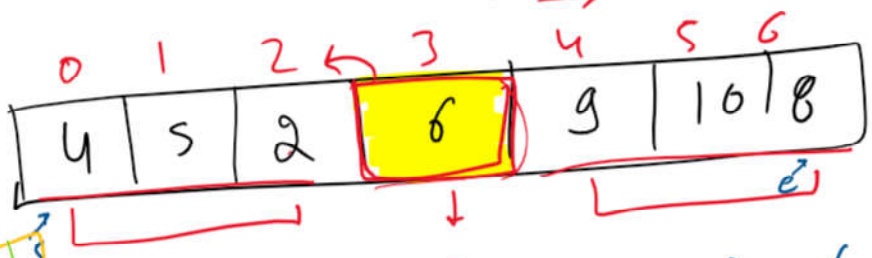


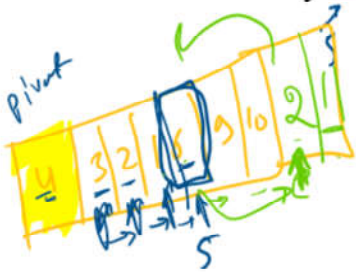


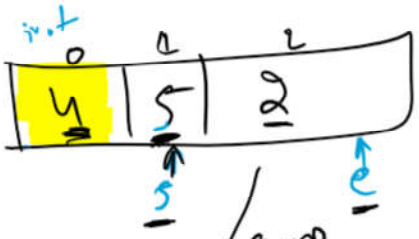
→ partition Hoare



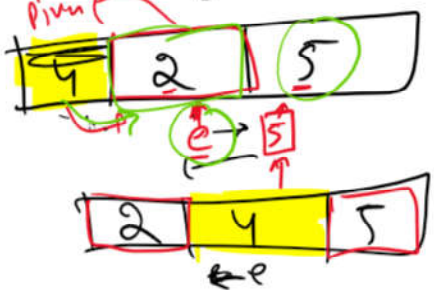








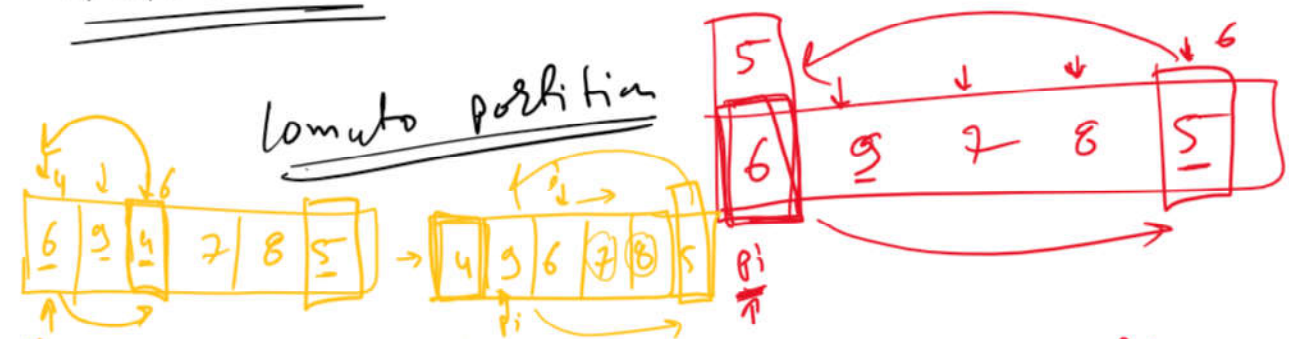
swap



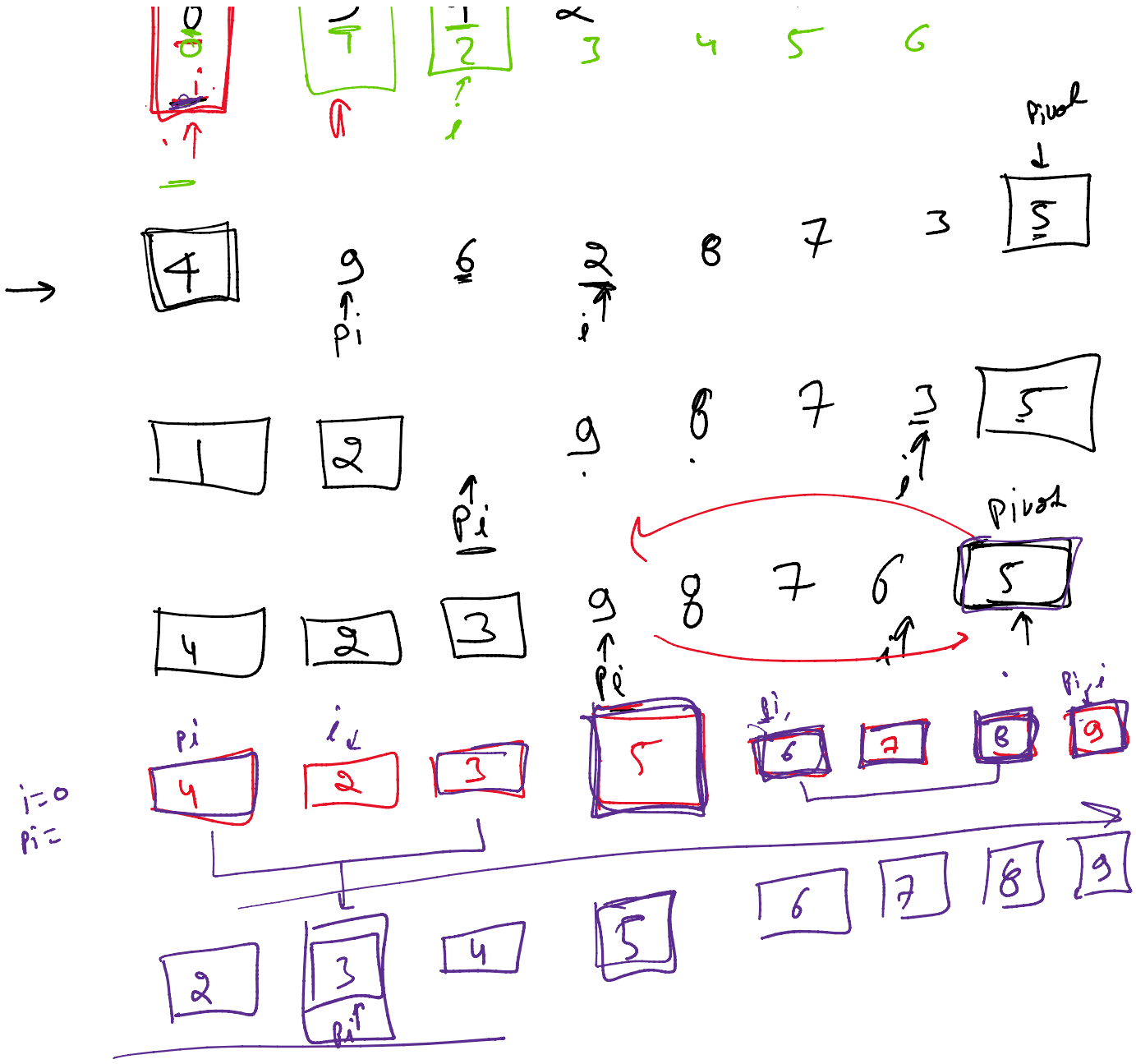

```
partition(arr, s, e)
    pi    = s
    pivot = arr[pi]
    s = s+1
    while s >= e:
        while arr[s] <= arr[pi]:
            s = s+1

    while arr[e] > arr[pi]:
        e = e-1

    swap(arr[s], arr[e])

    swap(arr[pi], arr[e])
```

→ Quick Sort

Lomuto partition



pi → update pi if you find an element which is less than pivot elem




```
Quick Sort(arr, start, end)
    if start < end:
        → pi = Partition(arr, start, end)
        → Quick Sort(arr, start, pi-1)
        → Quick Sort(arr, pi+1, end)
```

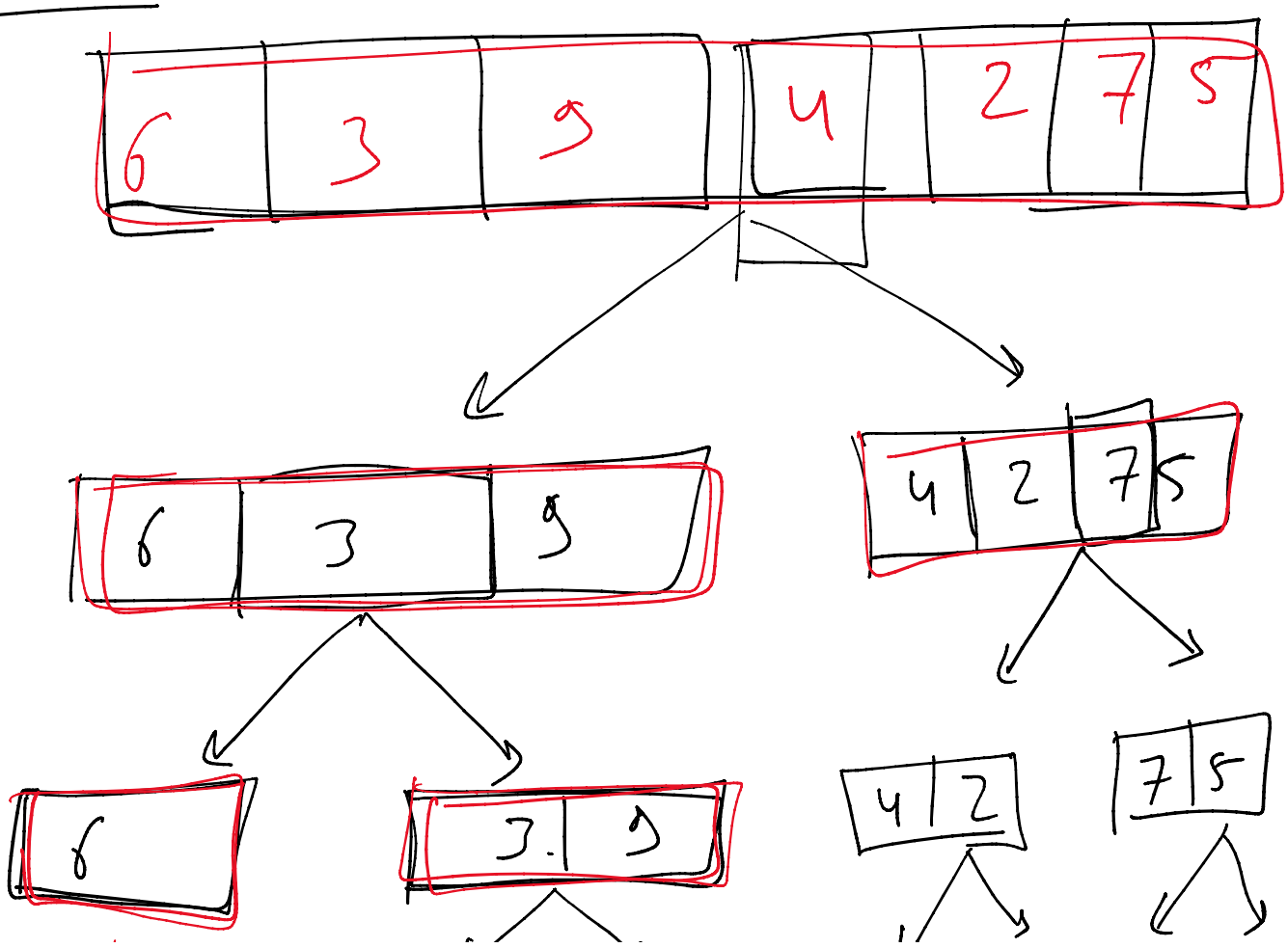
$$T(n) = 2T(n/2) + O(n) \quad \text{---(1)}$$

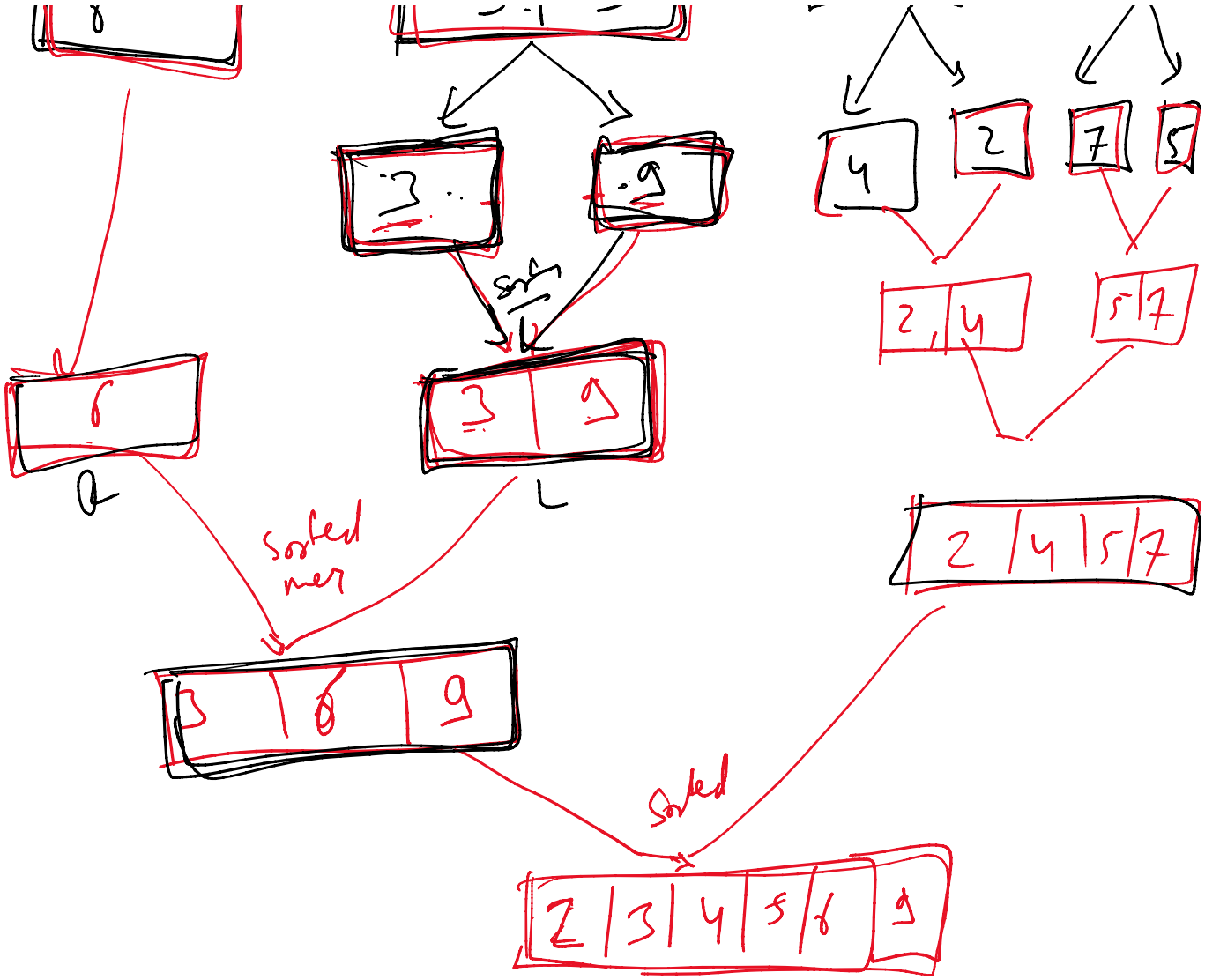
$$T(n/2) = 2T(n/4) + O(n/2) \quad \text{---(ii)}$$

$$T(n/4) = 2T(n/8) + O(n/4) \quad \text{---(iii)}$$

$$T(n/2^k) = 2T\left(\frac{n}{2^{k+1}}\right) + O\left(\frac{n}{2^k}\right)$$

Merge Sort





arr1 = [6, 9, 12, 15]

arr2 = [3, 7, 14, 16, 19, 20]

[3, 6, 7, 9, 12, 14, 15, 16, 19, 20]

Sorted Merge (arr1, arr2)

```
i = 0
j = 0
arr = [ ]
... i < len(arr1) and j < len(arr2):
```

```
arr = []
while i < len(arr1) and j < len(arr2):
    if arr1[i] < arr2[j]:
        arr.append(arr1[i])
        i += 1
    else:
        arr.append(arr2[j])
        j += 1

while i < len(arr1):
    arr.append(arr1[i])
    i += 1

while j < len(arr2):
    arr.append(arr2[j])
    j += 1
```