Merge Sort

3 2 6 | 1 5 4

✓ 1. divide into two parts : work

2. sort them individually : recursive case ??

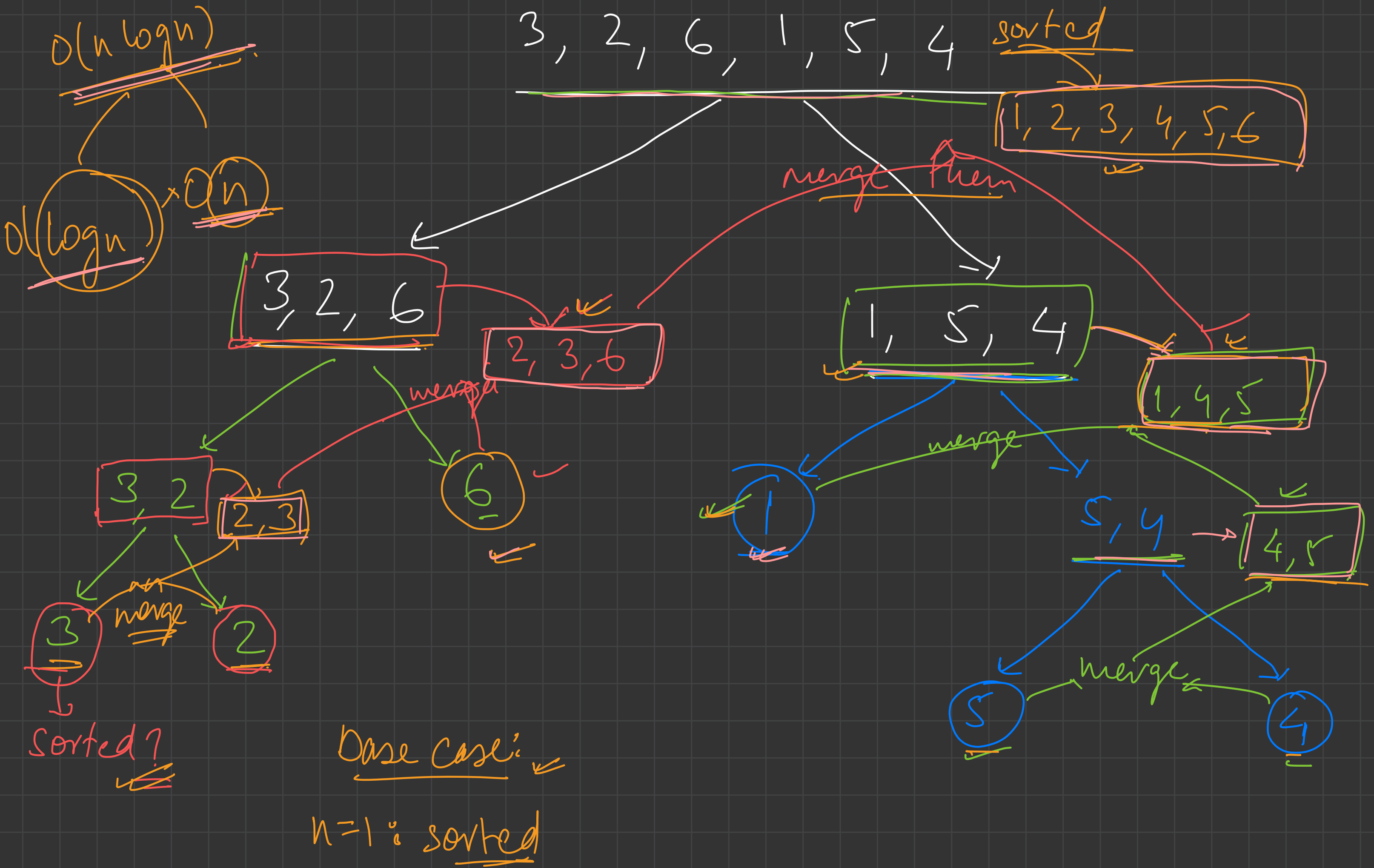
✓ 3. merge them : use recursive solution to build answer

Merge Two Sorted Arrays

3 2 6 1 5 4

sort: 2 3 6 sort: 1 4 5

merge: 1 2 3 4 5 6 ~~4~~



O(n logn)
O(logn)
× O(n)
3, 2, 6, 1, 5, 4
sorted
1, 2, 3, 4, 5, 6
merge them
3, 2, 6
2, 3, 6
merge
1, 5, 4
1, 4, 5
3 2
2, 3
6
1
merge
5, 4
4, 5
3
merge
2
5
merge
4
sorted?
Base case:
n=1 : sorted

O(n)

a → ~~1~~, ~~2~~, ~~5~~, ~~5~~, ~~8~~, 8, 9, 9

b → ~~1~~, ~~2~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~

merged sorted array

C : 1, 1, 2, 2, 2, 3, 4, 5, 5, 5, 8, 8, 9, 9

✓ In-place mergesort: no extra space ✗

→ merge step: $O(n^2)$

overall

space: —

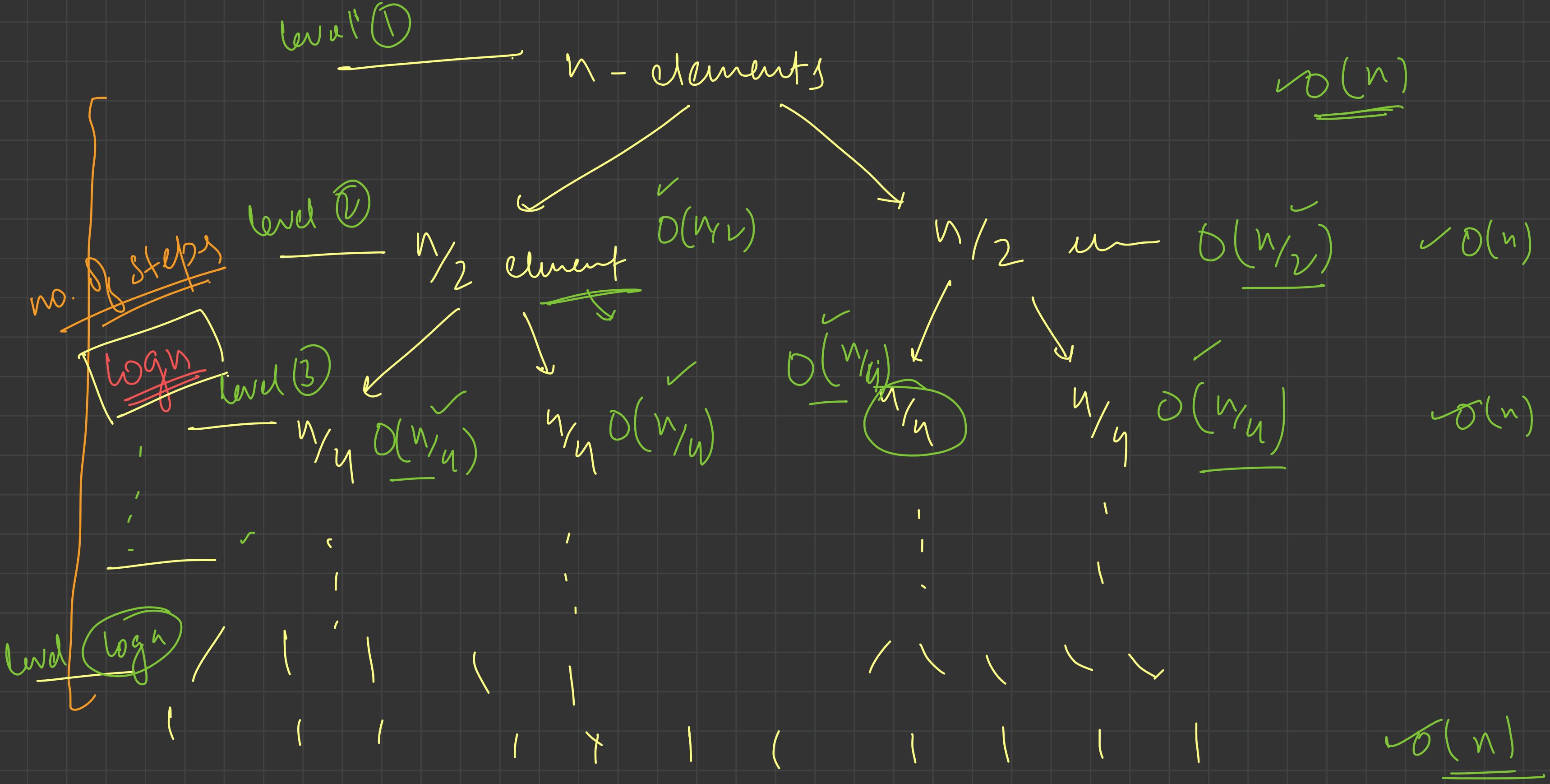
time: $O(n^2 \cdot \log n)$

Out-of place mergesort ✓ : uses extra space ✓ $O(n)$

→ merge step is $O(n)$

space : $O(n)$

time: $O(n \log n)$


level ①
n - elements
O(n)
level ②
n/2 element
O(n/2)
n/2
O(n/2)
O(n)
no. of steps
log n
level ③
n/4
O(n/4)
n/4
O(n/4)
O(n/4)
n/4
n/4
O(n/4)
O(n)
level log n
O(n)

$$n = 2^x \rightarrow \frac{2^x}{2} = \frac{2^{x-1}}{2} = \frac{2^{x-2}}{2} = 2^{x-3} \ldots \; 2^{x-x} = 2^0 = 1$$

x, min. value such that

$\boxed{n} \le 2^x$ $\rightarrow$ no. of steps

$2^x \Rightarrow$ $x+1$ steps

$\log_2 2^x \Rightarrow \boxed{x}$

$\log n \rightarrow$ no. of steps to reduce to 1

Q. n elements?

no. of steps / height of tree ??

$n = 2^x \rightarrow x$ steps

$\boxed{\log n}$

$n \neq 2^x$

$n \leq \boxed{2^y}$ → min such y

y steps $\log 2^y$

no. of steps ≤ no. of steps

y

$\boxed{\text{at most } y\text{-steps}}$

O( ) worst-case

at-max

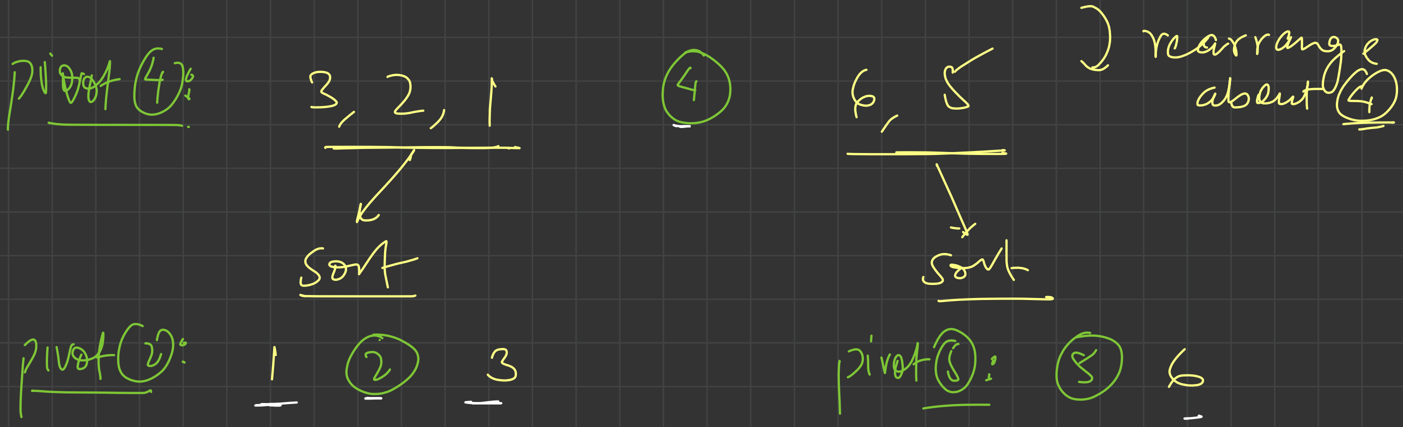
at-most

Time Complexity: $O(\log n) \times O(n)$

$= O(n \log n)$

QuickSort

3, 2, 6, 1, 5, 4



① pick a pivot element and partition about it.

– get the partition_index (p)

② Sort the subarray (start, p-1) and (p+1, end) recursively.

Choice of pivot

① should we pick an element which is not in the array?? → it will work ?!

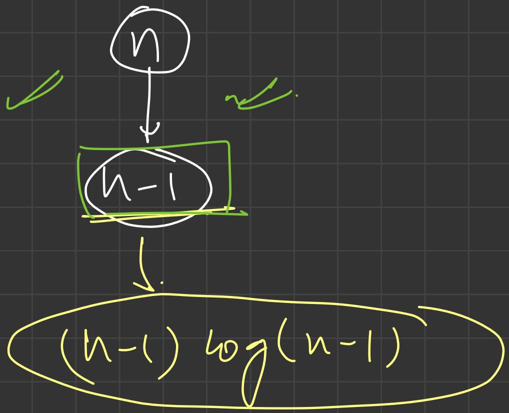
3 6 8 4 7 1   5

3 4 1   ~~5~~   6 8 7

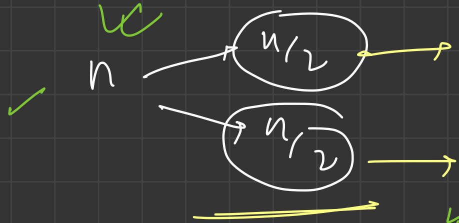
→ we choose pivot from the array in order to reduce the no. of elements in next step.

3, 6, 8, 4, 7, 1

pivot ⑧: 3, 6, 4, 7, 1 ⑧



pivot ④ 3, 1 ④ , 6, 8, 7



$$\frac{n}{2}\log\left(\frac{n}{2}\right) = \frac{n}{2}\left(\log(n) - 1\right)$$

$$\frac{n}{2}\log\left(\frac{n}{2}\right)$$

$$\log\left(\frac{n}{2}\right) = \boxed{\log n - 1}$$

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

$$n(\log n - 1)$$

$$n \underline{\log n} - \boxed{n}$$

$$(n-1)\log(n-1)$$

$$n \underline{\log(n-1)} - \boxed{\log(n-1)}$$

① preferable pivot is such a value which partitions the elements in equal (almost) halves.

→ median value

✓ How to find the median value efficiently??

→ ① Pick any value at random

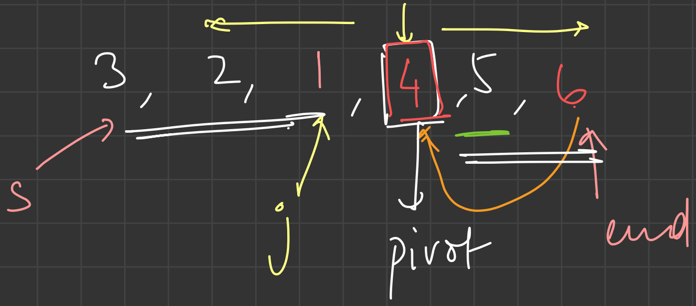
② Pick the mid element

③ Pick the last element

④ Pick the first element

worst-case: $O(n^2)$

best-case: $O(n \log n)$



```
int pivot = arr[end]; = 4

int j = start - 1;
for (int i = start; i < end; ++i){
   if(arr[i] < pivot){
      swap(arr[i], arr[++j]);
   }
}

swap(arr[j+1], arr[end]);
return j+1;
```

pivot

arr[i] = 3 < 4

swap(3, 3); X

arr[i] = 2 < 4 ✓

Swap(2, 2) ; ✗

arr[i] = 6 < 4 X

arr[i] = 1 < 4

swap(1, 6) ; ✓

arr[i] = 8 < 4 X

Permutation ??

↓ ↓

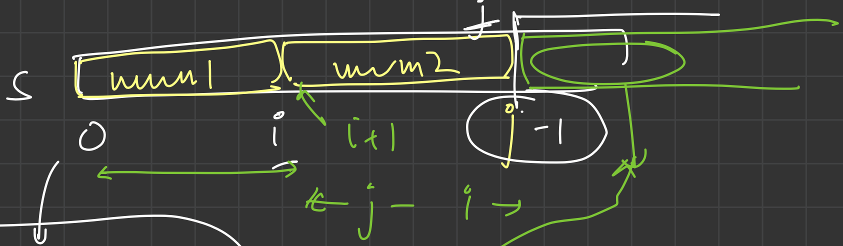
1 9 9 100 199

first / second num

Should not have more

than half digits

Ⓢ



```
for(i -> 1, n/2){

    for(j -> i+1, n){
        num1 -> s.substr(0, i)
        num2 -> s.substr(i+1, j-i);
    }

}

String rem = s.substr(j);
```

$$\overset{100}{n3} = \overset{1}{num1} + \overset{99}{num2};$$

n3 → String "100"

s → 199 | 100199

it should have a prefix of n3

otherwise no solution

num1 = 99

num2 = 100

199100199

n3 = 199 → "199"