Interview Questions

1) Kth smallest element   k = 2 (2nd smallest element)

↕

↳ Quicksort → Sorted array → Θ(nlogn)

Merge Sort ↗

↓

Kth Largest element   return arr(k-1)

✓ Selection Procedure → Θ(n)

↓

m → Pivot element index

20, 40, 5, 7, 9, 12, 46

(QuickSort)

Partition → return position of pivot element

Selection Procedure (arr, 2, 0, len(arr)-1)

k = 2, i = 0, J = len(arr)-1

↳ m = 4

$\underbrace{\overset{0}{5},\ \overset{1}{7},\ \overset{2}{9},\ \overset{3}{12}},\ \overset{4}{20},\ \underbrace{\overset{5}{40},\ \overset{6}{46}}$

k=2

↳ output = 7

m > k

4

Left side

Selection Procedure (arr, k, i, m-1)

m-i ⇐

else: Right side

J-m ⇐   Selection Procedure(arr, k, m+1, J)

Recurrence Relation

Partition algo

$$T(n) = \begin{cases} T(n-1) + \overset{\uparrow}{n} & \rightarrow \text{Left side} \\ \text{OR} \\ T(j-n) + n & \hookrightarrow \text{Right side} \end{cases}$$

Best/average

$$T(n) = T(n/2) + n$$

$$= \Theta(n)$$

Worst case

$$T(n) = T(n-1) + n$$

$$= \Theta(n^2)$$

3) kth Largest element

Partition ↓ Pivot ↓

{5, 7, 9, 12, 20, 40, 46}

5	7	9	12	20	40	46
0	1	2	3	4	5	6

$k = 2$

↳ Output = 40

$n = 4$

$n > k$:

↳ Right search space

Sort colors

↳ [0, 1, 2) → Sorting algo

↳ Quicksort

↓

$\theta(n \log n)$

Two Pointers approach

$\theta(n)$

curr = 0

p0 → zeros (0's)   p0 = 0

p2 → two (2's)   p2 = len(nums) -1.

[2, 0, 2, 1, 1, 0]

(0, 0, 1, 1, 2, 2)

↑ 0   ↑ n -1

nums[curr] == 0

↳ swap(nums(curr), nums(p0))

p_0 += 1

curr += 1

→ nums(curr) == 1:

curr+ = 1

nums(curr) == 2:

swap(nums(curr, nums(p2)))

p2 -= 1

4) Majority Element

Hash Table

Approach 1

Hashing → Dictionary

⇓

(key-value) pair

$(\overset{\checkmark}{2}, \overset{\checkmark}{2}, \overset{\checkmark}{1}, \overset{\checkmark}{1}, \overset{\checkmark}{1}, \overset{\checkmark}{2}, \overset{\checkmark}{2})$

freq

key	value
2	4 *
1	3

2

Dictionary → { }

List → [ ]

5

[1, 2, 1, 3, 5, 6, 4]
0 1 2 3 4 5 6

nums → array name → optimized approach

output → 6

n = 7

[4, 3]
0 1
→ 4

nums(0) > nums(1)
↳ return 0

[_ _ _ _ _ 4, 6] → 6
0 1 2 3 4 5 6

nums[-1] > nums(-2)
↳ return len(nums) - 1

$(0+6)/2 = 3$

start = 0
end = n - 1 (6)

(1, 2, 1, 3, 5, 6, 4)
0 1 2 3 4 5 6
↑

```
while start < end:
    mid = start + (end - start)//2
    if nums(mid) < num(mid+1)
        start = mid+1
    else:
        end = mid
return end
```

6 < 4

3 < 5 5 < 6

end = 5
mid = 5

start = ~~4~~ 5

mid = (5+4)/2
= 4