# Quick Sort → Partition

↳ Divided the array → smarter way

↳ Partition algo

1) Inplace
2) Not stable sorting algo
3) Practical real life use-cases
   ↳ Actually sorting is required

No need of combine part

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| ~~50~~ | 40 | ~~70~~ | ~~10~~ | ~~30~~ | 90 | ~~45~~ | 67 | 79 |
| 45 | | 10 | 70 | 50 | | | | |

P J → J → J → J → J → J → J → J

pivot = 50

i i i i i

{ i → smaller than pivot
  J → greater than pivot

partition ↳ (4)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 45 | 40 | 10 | 30 | 50 | 90 | 70 | 67 | 79 |

Smaller < 50 ↳ pivot element

greater > 50 ↳ pivot element

QS(arr, 0, 3)

Qs(arr, 5, 7)

O(n)

```
partition(arr, p, q):
    pivot = arr(p)
    i = p
    for(j = i+1 to q):
        if arr(j) ≤ pivot:
            i += 1
            swap(arr(i), arr(j))
```

swap(arr(i), arr(p))

↳ pivot

return i

T(n)

QS(arr, p, q):

mid = p + (q-p)//2

m ✗ -p ✗

if i < j:

m

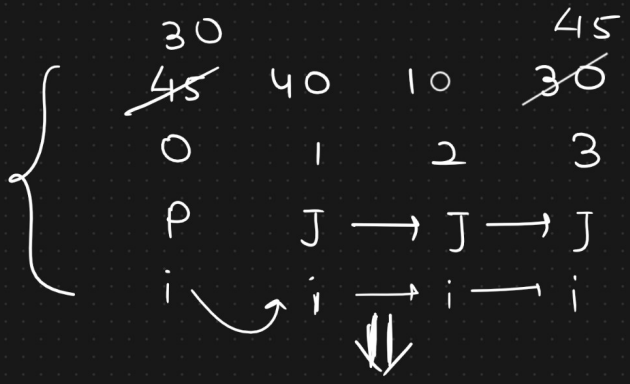
m = partition(arr, p, q)

T(m-p) — QS(arr, p, m-1)

T(q-m) — QS(arr, m+1, q)

↳ q - (m+1) + 1

q - m




i = 3

(30 40 10) 45

0 1 2 3

i

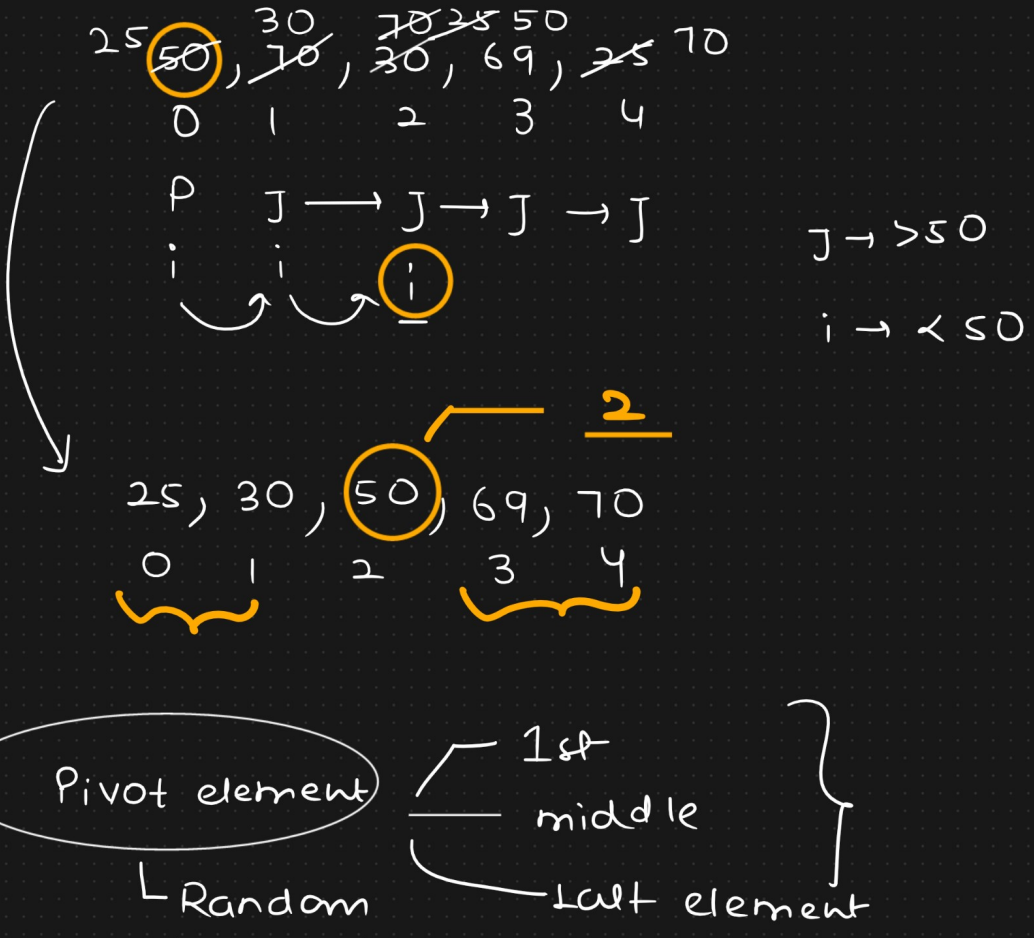
QS(arr, 0, 2)

Small Problem

↳ p == q

↳ return arr(p)

pivot = 50




## Mergesort

mid = p + (q - p) // 2

T(n/2) — Mergesort (arr, p, mid)

T(n/2) — Mergesort (arr, mid + 1, q)

## Recurrence Relation

$$T(n) = \begin{cases} c & ; n = 1 \\ T(n-p) + T(q-n) + n & ; n > 1 \end{cases}$$

## Best case / Average case

$$T(n) = T\left(\frac{n}{2}\right) + T\left(n/2\right) + n$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= \Theta(n \log n)$$

## Worst case Scenario

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

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

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 50 | 70 | 60 | 69 | 72 | 87 | 68 |

i → J → J → J → J → J → J

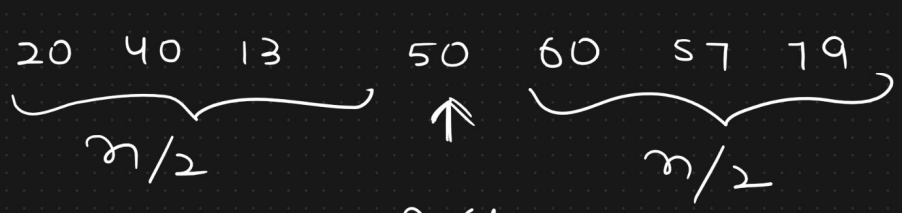
p

No element

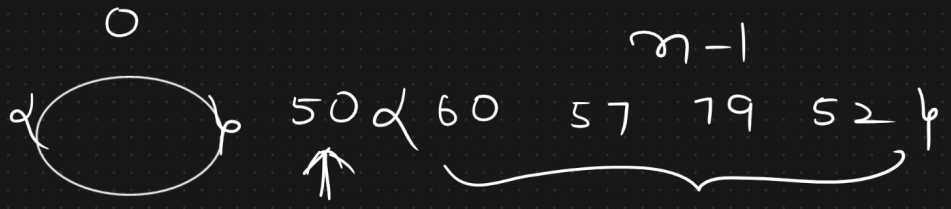
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| ( ) | 50 | ( 70 | 60 | 69 | 72 | 87 | 69 ) |

Empty

QS(arr, 1, 6)

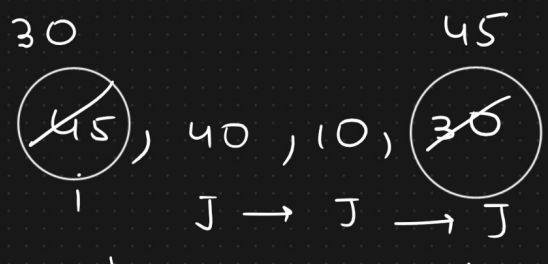


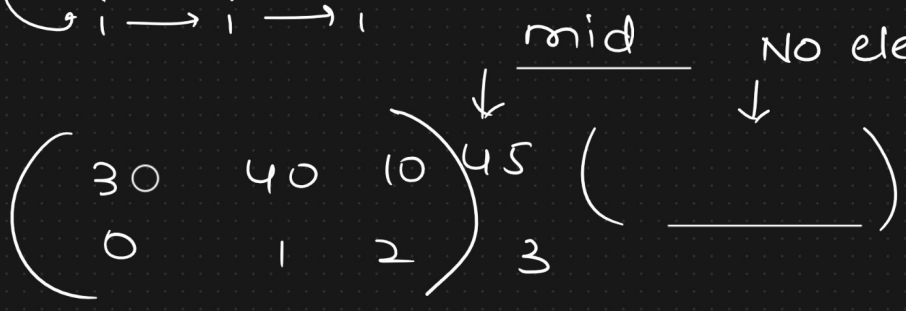

Best case



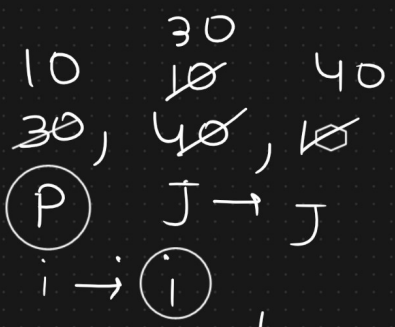

$T(n) = T(n-1) + n$

---




↳ Qs(arr, 0, 2)




↳ 10, 30, 40

Worst case Scenario → almost or Completely Sorted array

O(n) ← Insertion Sort ←

10, 20, 30, 40, 50

i → J → J → J → J

P

Qs(1,4)

( ) 10, (20, 30, 40, 50)

↳ NO element

0 1 2 3 4

↓

20, 30, 40, 50

i J → J → J

P

Qs(2,4)

( ) 20, (30, 40, 50)

↳ NO element

1 2 3 4

Never ever go for Quicksort

↓

O(n²)

## Randomized QuickSort

↳ Pivot element randomly

random
↳ random index → pivot element

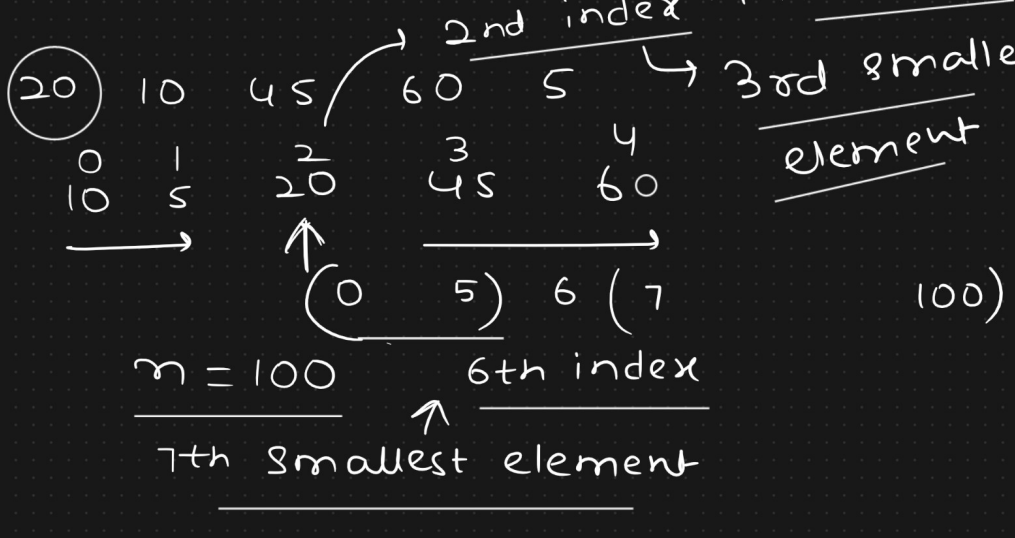
⇓

6 ⟷ 0

↓

same code

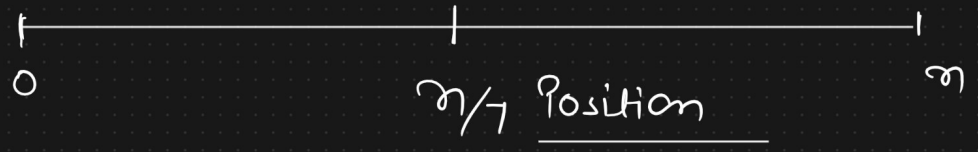
Probability → Worst case Scenario
↳ Quite less ∝ normal QuickSort



Interview

Partition → $O(n)$

1) $n/7^{th}$ smallest element → Position → Pivot → choosing the
   └ $O(n^2)$



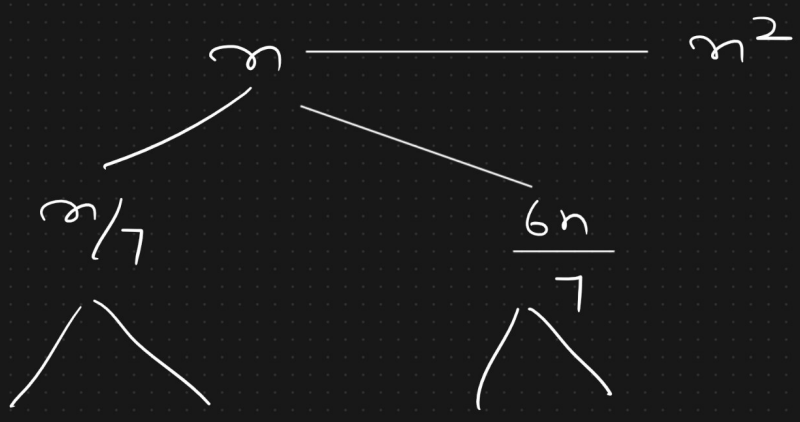

Worst case time complexity of quickSort

$$T(n) = T(n/7) + T(n - n/7) + n^2 + n$$

- $4 \log n$
- $4 n \log n$
- $4 n^2$
- $4 n^3$

$$T(n) = T(n/7) + T\left(\frac{6n}{7}\right) + n^2 \approx O(n^2)$$

↳ Recursive Tree Approach



2)


25th largest element

$T(n) = T(24) + T(n-25) +$ ↳ Pivot element

$n^2 + n$ ↳ $O(n^2)$

$T(n) = T(n-25) + n^2$

Partition → $O(n)$

$T(n) = O(n^3)$

Worst case Quicksort

$T(n) =$

| 20 | 47 | 52 | 12 | 26 | 69 | 74 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

k = 2

↳ 2nd smallest element

↳ pivot + 1

↳ position

SelectionProcedure

{ k = 2
  m = 4

Output = 20

T(n)

```
SelectionProcedure(arr, p, q):          position
                                        ↑
    m = Partition(arr, p, q)
    if k == m:
        return arr(m-1)
    elif k < m:
        SelectionProcedure(arr,
            p, m-2)
    else:
        SelectionProcedure(arr,
            m, q)
```

C

T(m-p) — Left side

T(q-m) — Right side

pos = index + 1

## Recurrence Relation

Left side

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

↳ Partition

or

$$T(n) = T(q-m) + n$$

↳ Right side

### Best case

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

$a = 1$

$b = 2$

$\log_b^a = 0$

$k = 1, p = 0$

$\log_a b < k \longrightarrow$ Case 3

$\Theta(n)$

### Worst case

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

$= O(n^2)$