Quick Sort

Imagine Sorting Students in a line

we have 2 ways

1. → Sort our self by figuring out.
2. → make a reference point & ask them to sort themselves

↓

assume we only pick shortest of all or tallest of all, rest will arrange.

Now writing few lists

All variants of Quick Sort ☐ highlights pivot

$\boxed{10}$	80	90	60	30	20				
6	3	5	4	2	1	$\boxed{9}$			
7	6	4	[illegible]	$\boxed{10}$	16	12	13	14.	

which elements are sorted ?

An element is sorted if all elements on left one ~~sorted~~ smaller & all elements on right one larger.

→ Quick sort is a divide & conquer algorithm

0	1	2	3	4	5	6	7	8	9
10	16	8	12	15	6	3	9	5	+∞

Steps

① Take first element as pivot 10

e.g. $\boxed{10}$ → Assign l as low

② Partition to put 10, pivot on right place. For that, take i & j

l

10	16	8	12	15	6	3	9	5	∞
	i								j

first increment i & check if element [i] > 10 [pivot]

Then

decrement j until element [j] < 10 [pivot]
swap i & j & then continue

10	5	8	12	15	6	3	9	16	∞
	i							j	

Next i at 12 & j at 9

10	5	8	9	15	6	3	12	16	∞
			i				j		

10 5 8 9 15 6 3 12 16 ∞

i j

swap.

l

10 5 8 9 3 6 15 12 16 ∞

j i

Stop here

replace l with j
(pivot

so j is position of pivot.

$\underbrace{6 \quad 5 \quad 8 \quad 9 \quad 3}_{\text{unsorted}} \quad \underbrace{10}_{\text{sorted}} \quad \underbrace{15 \quad 12 \quad 16 \quad \infty}_{\text{unsorted}}$

pivot/
partiong
position

↓

recursion, do

same

1. → pivot.
2. partition

6 5 8 9 3

l i i j

6 5 3 9 8

j i j

5 3 6 9 8

us sorted us.

5 3 ∞ 9 8 ∞

j i i j i

3 5. 8, 9.

only swap

15 12 16 ∞

i j i j

12 15 16.

Partition Algo

h → length of array

```
partition (l, h)
{   pivot = A[l]
    i = l & j = h
    while (i < j)
{
    do
      { i++
      } while (A[i] <= pivot);
    do
      {
        j --;
      } while ( A[j] > point);
    if (i < j)
    swap (A[i], A[j])
}
    swap A[l], A[j];
    return j;
}
```

Quick Sort Algo

```
QuickSort (l, h)

{ if (l < h)
  { j = portion (l, h)

    QuickSort (l, j);
    QuickSort (j+1, h);
  }

}
```

Quick Sort Analysis

(2)

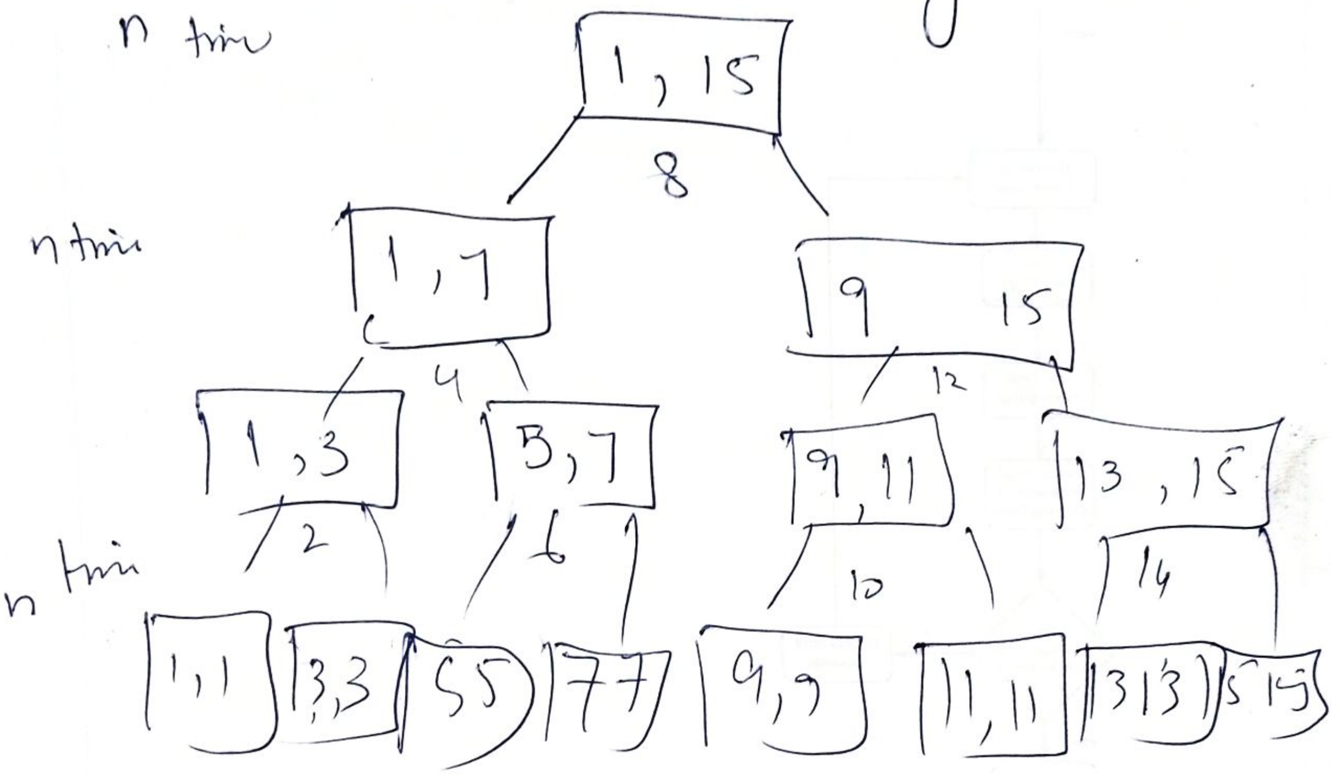
→ It is recursive

So list is divided

Assume = $\boxed{\text{What is best \& worst case}}$?

we have 15 elements 1–15

& partition is always mid.



Almost (n)

As it is dividing $\frac{n}{2} \Rightarrow \frac{\frac{n}{2}}{2} = \frac{n}{4} = \frac{n}{2^k}$

we can apply $= n(\log n)$

$\boxed{\text{Best Case}}$ if partition is middle

~~Median~~

Normal case

8

~~Worst Case~~

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

1 2 3 4 5 6 7

median is the middle element of sorted list.

It may happen but not always possible.

So ~~wort~~ worst case

2 4 8 10 16 18 17

i j

so after partition.

2 4 8 10 16 18 17

j i

j

Same element is sorted

So if we make a Tree

~~0 1 2 3 4 5~~

~~2 4 6 8 10 16 18 17~~

	1	2	3	4	5	6	7
1	2	4	8	10	16	18	17

n $\boxed{1,7}$

n - 1 $\boxed{2,7}$

n - 2 $\boxed{3,7}$

$\boxed{4,7}$

$\boxed{5,7}$

2 $\boxed{6,7}$

1 $\boxed{7,7}$

$$= n + (n-1) - (n-2) \ldots 2 + 1$$

$$= \frac{n(n+1)}{2} = O(n^2)$$

QSO

Best $n \log n$
Averge $n \log n$.
Worst n^2

In merge Sort only $n \log(n)$

QSO

how to Solve this issue

~~middle~~

1 → Select middle element as pivot

2 → Select Random element as pivot.

Still some cases can be n^2

Space Complexity $\log n$ to n,

due to Stack

Perform
Example

Make a Treee Tree for.

35 50 15 25 80 20 90 45 ∞

25 20 15 35 80 50 90 45

25 20 15 80 50 45 90.

⇒ why Quick Sort is better

① → less space complexity
no additional resource required

② Randomize version is as efficient as merge sort