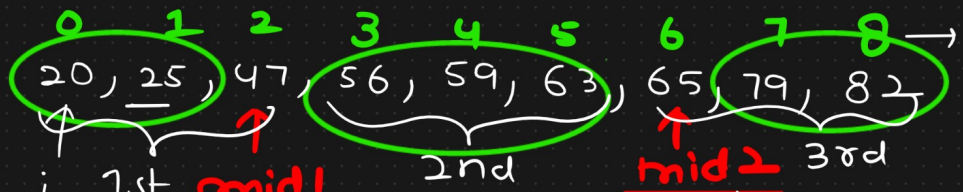
Ternary Search



i = 0

J = 8

mid1 = i + (J - i)//3 = 0 + 8//3 = 0 + 2 = 2

key = 79

mid2 = J - (J - i)//3

= 8 - 8//3

= 8 - 2 = 6

i, mid1 - 1 → 1st Part

mid2 + 1, J → 3rd Part

2nd Part

mid1 + 1 —— mid2 - 1

```
ternarySearch(arr, i, J, key):
while i<=J: if arr[mid1] == key:
                return mid1

            if arr(mid2) == key:
                return mid2
```

C

```
if key < arr(mid1):
    return ternarySearch(arr, i, mid1-1,
                         key)

elif key > arr(mid2):
    return ternarySearch(arr, mid2+1,
                         J, key)
```

1st Part

3rd Part

else :

return ternarySearch(arr, mid1+1, mid2-1, key)

return -1

Recurrence Relation

$$T(n) = T(n/3) + c$$

$$= T(n/3^2) + c + c = T\left(\frac{n}{3^2}\right) + 2c$$

$$= T(n/3^3) + c + c + c = T\left(\frac{n}{3^3}\right) + 3c$$

$\downarrow$ k times

$$\frac{n}{3^k} = 1$$

$$n = 3^k \implies k = \log_3 n$$

$$\Rightarrow T\left(\frac{n}{3^k}\right) + c \cdot k$$

$$\Rightarrow T\left(\frac{n}{3^{\log_3 n}}\right) + c \cdot \log_3 n$$

$$\Rightarrow T\left(\frac{n}{n \log_3 3}\right) + c \cdot \log_3 n$$

$$\Rightarrow O(\log_3 n)$$

Interview → Adobe

Binary Search	Ternary Search
$O(\log_2 n)$ >	$O(\log_3 n)$

*

Assignment Problem

↳ Binary search is more preferable as comparable to ternary search?

Sorting

Comparison-based sorting

Compare the elements inside the array

1) Bubble Sort
2) Selection Sort
3) Insertion Sort
4) QuickSort
5) MergeSort
6) HeapSort
7) ShellSort

non comparison based sorting

no comparison within the elements

1) Count sort
2) Radix sort
3) Bucket sort

Divide & conquer module

Stable & unstable sort

$20, 10^a, 5, 15, 10^b, 25, 45$ → original array

↓ sorting algo → MergeSort

$5, 10^a, 10^b, 15, 20, 25, 45$

↳ stable sorting algo

$5, 10^b, 10^a, 15, 20, 25, 45$

↳ not stable sorting algo

⟹ (QuickSort ⟸ HeapSort)

- sort()

X

	Name	Score
123	Priya	89 →
47	Priya	45 →
146	Priya	72 →
150	Priya	100 →
171	Priya	69

Inplace & Outplace sorting

arr = [_____]

↳ extra space

↳ new array ⟹ Outplace sorting

↳ MergeSort

Bubble Sort

n = 7

0	1	2	3	4	5	6
~~70~~	~~20~~	~~50~~	~~30~~	~~90~~	~~5~~	~~15~~
		~~70~~	70	5	~~90~~	90
20	~~70~~	30			15	
	50					

largest element

Pass 1: (20, ~~50~~, ~~30~~, ~~70~~, ~~5~~, ~~15~~) 90

(30 50 5 ~~70~~ 15 70)

Pass 2: (20, 30, ~~50~~, ~~5~~, ~~15~~) 70, 90

(5 ~~50~~ 50 15)

Pass 3: (20, ~~30~~, ~~5~~, ~~15~~) 50, 70, 90

(5 ~~30~~ 30 15)

Pass 4: (~~20~~, ~~5~~, ~~15~~) 30, 50, 70, 90

(5 ~~20~~ 20 15)

Pass 5: (5, 15), (20, 30, 50, 70, 90)

Pass 6: 5, 15, 20, 30, 50, 70, 90

⇓

Sorted array

n elements

⟶ Worst case ⇒ (n-1)

Passes

comparisons → Bubble sort → for every case

Total no. of comparisons

$(n-1) + (n-2) + (n-3) + \cdots + 1$

$\Rightarrow \frac{(n-1) * n}{2} = O(n^2)$

Sum of n natural numbers ⇒ $\frac{n(n+1)}{2}$

Swaps

Best case → 0

Total num. of swaps

Worst case → $(n-1) + (n-2) + (n-3) + \cdots + 1$

$\Rightarrow (n-1)n/2 = O(n^2)$

10, 20, 30, 40 → Best case for swap → 0

⇒ 40, 30, 20, 10 → Worst case for swap → n^2

Bubble sort

Time complexity → # Comparisons + # swaps

↓

$n^2 + n^2$

⇒ $O(n^2)$

Selection sort

0	1	2	3	4	5	6
~~70~~	20	50	30	90	~~5~~ 70	15

i=0 min = ~~0~~ 5

↳ to store the index of minimum element

Pass 1:

0	1	2	3	4	5	6
5	~~20~~ 15	50	30	90	70	~~15~~ 20

i=1 min = ~~1~~ 6

Pass 2:

0	1	2	3	4	5	6
5	15	~~50~~ 20	30	90	70	~~20~~ 50

i=2 min = ~~2~~ ~~3~~ 6

Pass 3:

0	1	2	3	4	5	6
5	15	20	30	90	70	50

i=3 min = 3

Pass 4

0	1	2	3	4	5	6
5	15	20	30	~~90~~ 50	70	~~50~~ 90

i=4, min = ~~4~~ ~~5~~ 6

Pass 5

0	1	2	3	4	5	6
5	15	20	30	50	70	90

$i = 5,\ min = 5$

Pass 6 $(5, 15, 20, 30, 50, 70, 90)$

$\Downarrow$

Sorted array

comparisons $\rightarrow (n-1) + (n-2) + (n-3) + \dots + 1$

$\Rightarrow \frac{(n-1)n}{2}$

$\Rightarrow O(n^2)$

$(n-1)$

swaps $\Rightarrow$ Pass 1 $\uparrow$ Pass 2 $\uparrow$ $1 + 1 + 1 + \dots + (n-1)$ $\downarrow$

Total number of swaps

$\Rightarrow O(n)$

Time complexity $\rightarrow$ # comparisons + # swaps

$\Rightarrow n^2 + n$

$\Rightarrow O(n^2)$