

1. Merge Sort
2. Quick Sort

What logic??

X { let
 Const
 Var }

$$\varphi \left\{ \begin{array}{l} arr1 = [1 \ 2 \ 3 \ 4 \ 5] \\ arr2 = [1 \ 2 \ 4 \ 3 \ 5] \end{array} \right.$$

lang)

logic

Golang

- net core

Python.






③ / C++ ⑤

Node

dependencies

Linux, Apache

Byte Code
Machine Code
Assembly Code

Lines Traveled

Hosting

Dedicated Cloud hosting

AWs
Azure
GCP

Interact.

CLI

Command line

GUI (Graphical User Interface)

Command line interface.

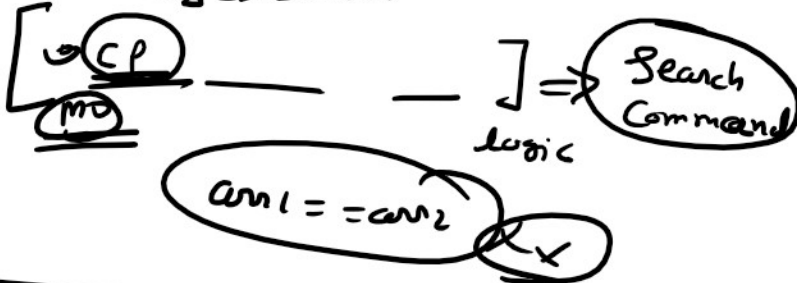
Graphical User Interface

AWS

↓

Commands

↓
Linux ⇒ Secure



Quick Sort:-

Merge Sort:-

Best - $O(n \log n)$

Avg - $O(n \log n)$

Worst - $O(n \log n)$

Quick Sort

Best - $O(n \log n)$

Avg - $O(n \log n)$

Worst - $O(n^2)$

* Quick Sort: } partition
↓
Sort an array.

→ Quick Select Sort:-

Search for a
Single specific Elem

Q 1 3 2 4 8 5 } k = 2nd
↓
Quick Sort } Smallest Element

[1 2 3 4 5 8]

① Naive approach

Q. 1 3 2 4 8 5
 low high
 i j
 Pivot
 k = 2nd Smallest elem.

1 < 5 ✓
 3 < 5 ✓
 8 < 5 ✗
 i++
 Swap arr[i] & arr[j]

arr[i+1] arr[pivot]
 0 1 2 3 4 5
 1 3 2 4 5 8
 X X

k = 2nd Smallest elem

low high
 1 3 2 4 5 8
 i j
 Pivot

0 1 2 3 4 5
 1 3 2 4 5 8
 X



1 3 2
 i j
 Pivot
 0 1 2 3 4 5
 1 3 2 4 5 8
 1 < 2 ✓
 3 < 2 ✗
 k = 2nd Smallest

1 3 4 5 8 2
 Best 1st Iteration
 Pivot

Code

Quick

Partition

↳ index of Pivot elem.

k = 2 2-1 = 1

1 3 2 4 8 5
 low 1 3 2 4 5 8
 i j
 Pivot

Pi = 4 k = 1

low 0 1 2 3 4 5 6
 1 3 2 4 5 8
 function Partition(arr, low, high) {

pi = 4 k = 1

function Partition(arr, low, high) {

→ return
 pivot's index

{

function kthSmallest(arr, low, high, k) {

let pi = Partition(arr, low, high)

if (pi == k-1) {
 return arr[pi]
 } // Base case

k = 2

if (pi < k-1) {
 return kthSmallest(arr, pi+1, high, k)
 }

else {
 return kthSmallest(arr, low, pi-1, k)
 }

}

find out a specific Pos's element

Medians

0	1	2	3	4	5	6	7	8
7	4	18	-6	1	12	14	28	19

-6 1 4 7 12 14 18 19 28

naive
approach

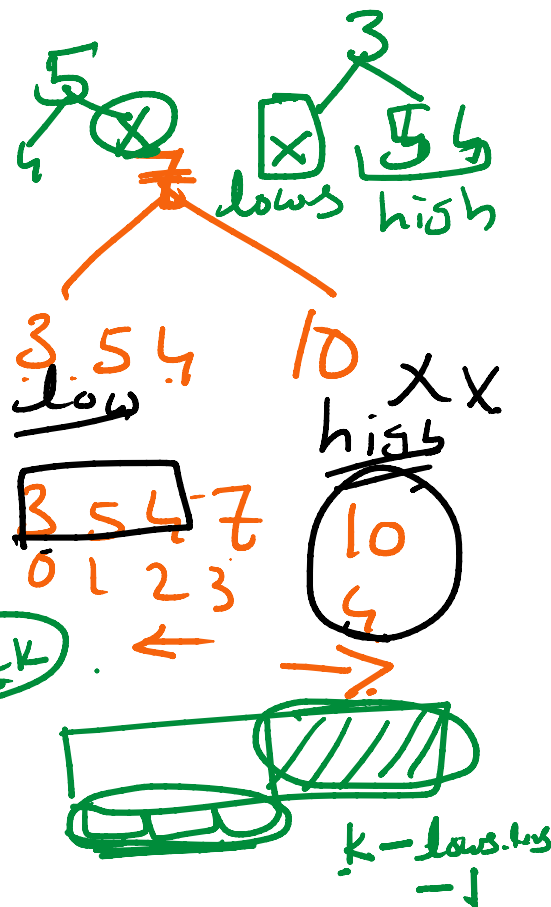
even $\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right) - 1$
odd $\Rightarrow \frac{n}{2}$

```
function QuickSelect_Median(arr) { // 7,3,5,4,10
  let len = arr.length; // 5
  let halfLength = parseInt(len / 2); // 2
  if (len % 2 !== 0) {
    return QuickSelect(arr, halfLength); // (arr, 2)
  }
  else {
    return (QuickSelect(arr, halfLength-1) + QuickSelect(arr, halfLength)) / 2;
  }
}
```

```
function QuickSelect(arr, k) {
  if (arr.length == 1) {
    return arr[0];
  }
  else {
    let pivot = arr[0];
    let lows = arr.filter((e) => e < pivot);
    let highs = arr.filter((e) => e > pivot);

    if (k < lows.length) {
      return QuickSelect(lows, k);
    }
    else if (k < lows.length + 1) {
      return pivot;
    }
    else {
      return QuickSelect(highs, k - lows.length - 1);
    }
  }
}
```

console.log(QuickSelect_Median([7,3,5,4,10]));



Q

$$T(n) = T(n-1) + O(n) \quad \underline{RR}$$

$$T(n-1) = T(n-2) + n-1$$

$$\boxed{\frac{n(n-1)}{2}} \quad \underline{\underline{n^2}}$$

Q

book = [
{

]