(1) K^{th} Smallest from array : After sorting the array, correct Element for index 'k'.

Prerequisite
→ Partitioning of array. { using pivot, we segregate arrays.

K=4

index-	0	1	2	3	4	5	6	7	8	9
arr →	35	33	42	10	14	19	27	44	26	31

Left

index-	0	1	2	3	4	5
arr →	10	14	19	27	26	31

Right

6	7	8	9
35	44	33	42

pivot

NOTE: pivot is present at arr[i]

pivot

pivot

return i-1;

partition Index = 5

pivot < ele.

i

j

out of Range

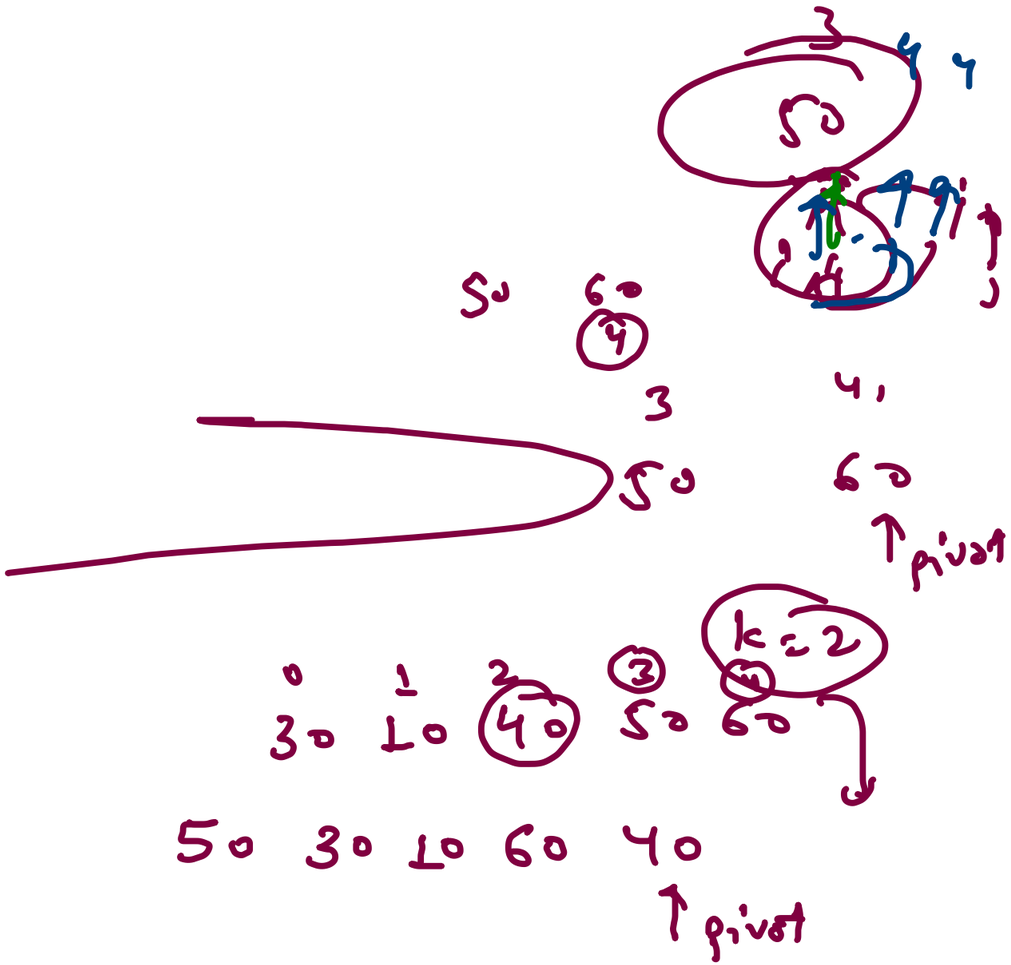
K' = K

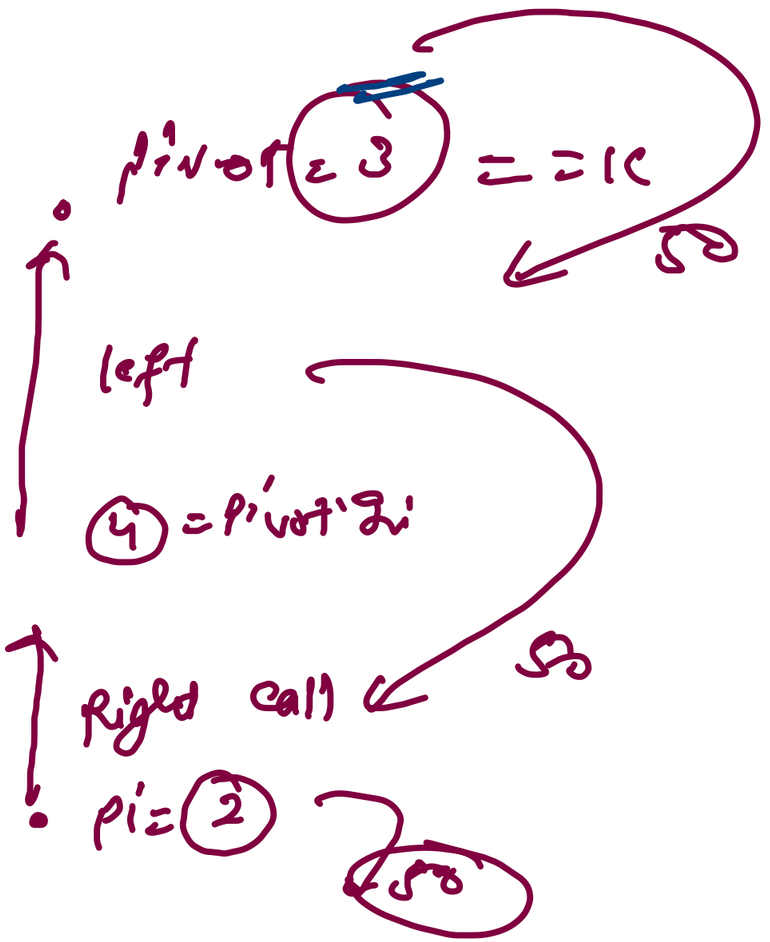
pi == k

k < pi → left

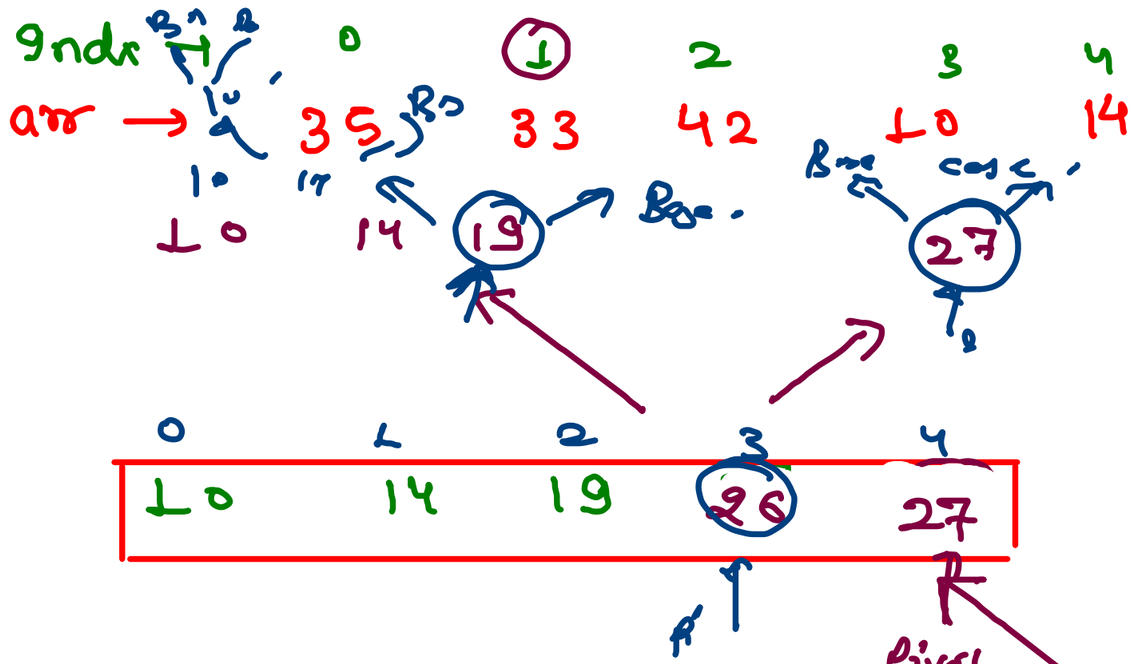
pi < k → right

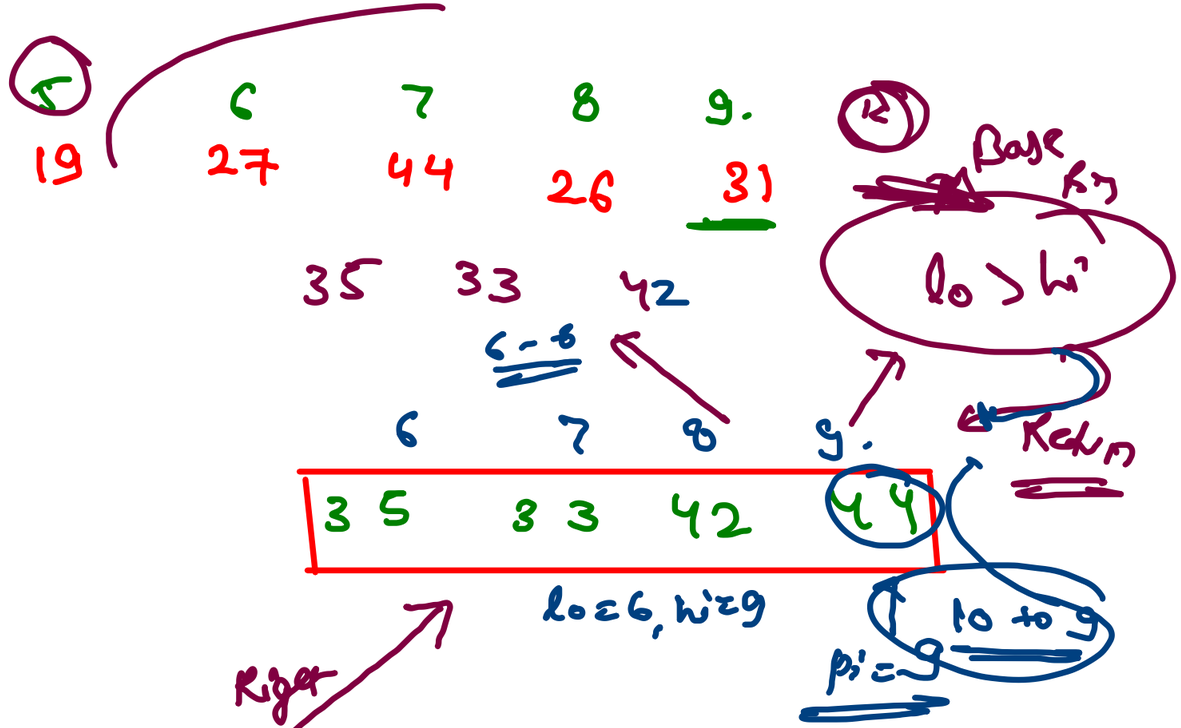
50 30 10 60 40 —Sort→ 10 30 40 50 60

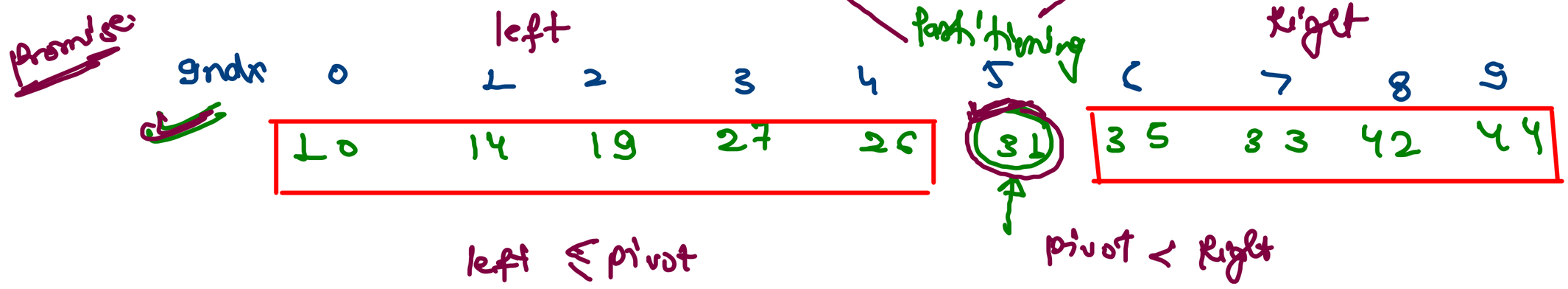


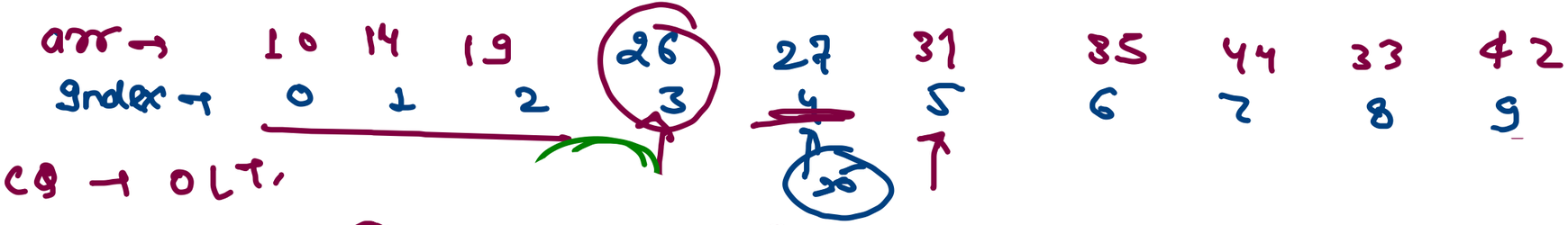


QuickSort → Inplace.



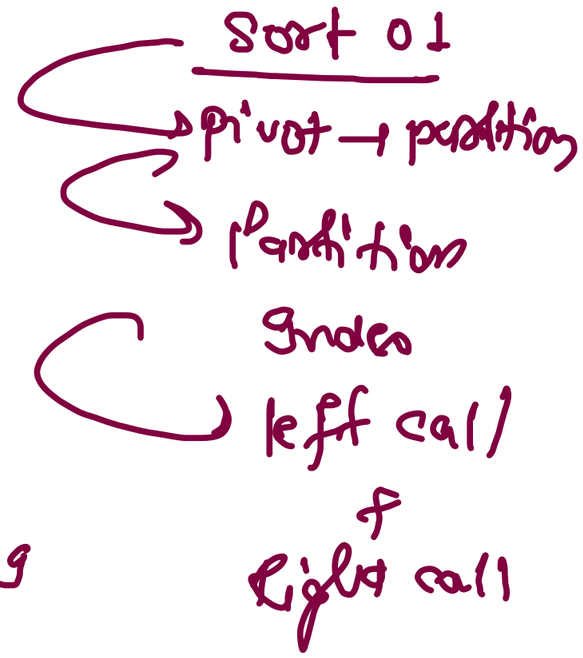







```
Array is : [10, 14, 19, 27, 26, 31, 35, 44, 33, 42] pi is : 5
Array is : [10, 14, 19, 26, 27, 31, 35, 44, 33, 42] pi is : 3
Array is : [10, 14, 19, 26, 27, 31, 35, 44, 33, 42] pi is : 2
Array is : [10, 14, 19, 26, 27, 31, 35, 44, 33, 42] pi is : 1
Array is : [10, 14, 19, 26, 27, 31, 35, 44, 33, 42] pi is : 0
Array is : [10, 14, 19, 26, 27, 31, 35, 44, 33, 42] pi is : 4
Array is : [10, 14, 19, 26, 27, 31, 35, 33, 42, 44] pi is : 8
Array is : [10, 14, 19, 26, 27, 31, 33, 35, 42, 44] pi is : 6
Array is : [10, 14, 19, 26, 27, 31, 33, 35, 42, 44] pi is : 7
Array is : [10, 14, 19, 26, 27, 31, 33, 35, 42, 44] pi is : 9
```

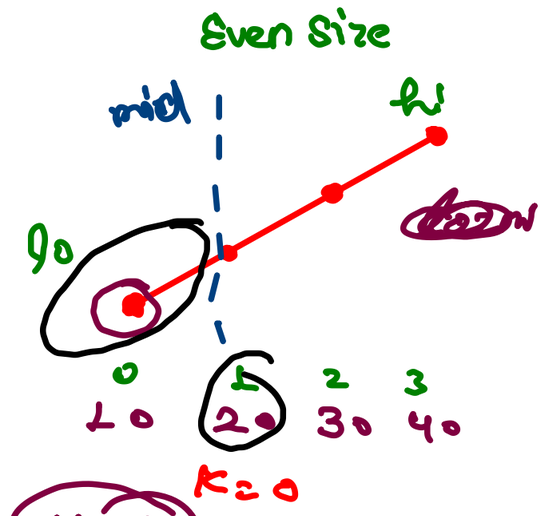
```java
public static void quickSort(int[] arr, int lo, int hi) {
    // base case
    if(lo > hi) {
        return;
    }

    // segregate array on the basis of pivot element
    int pivot = arr[hi];
    int pi = PartitionIndex(arr, pivot, lo, hi); // partitio

    String ar = Arrays.toString(arr);
    System.out.println("Array is : " + ar + " pi is : " + pi

    quickSort(arr, lo, pi - 1); // solve for left side
    quickSort(arr, pi + 1, hi); // solve for right side
}
```

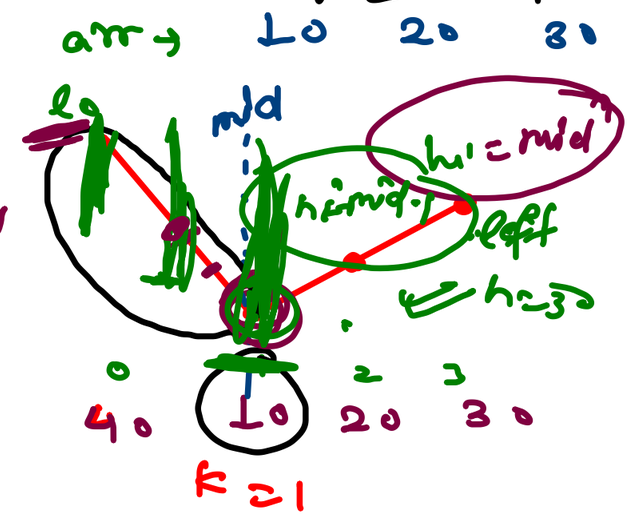



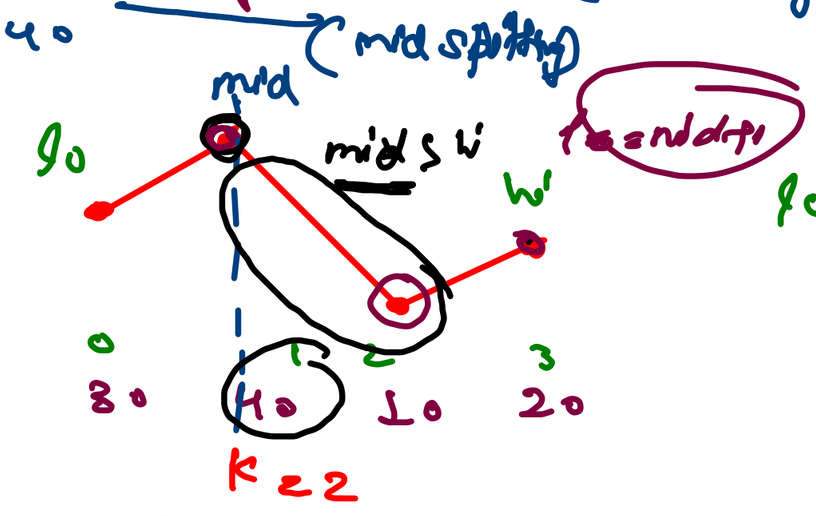
pivot in Sorted Rotated Array

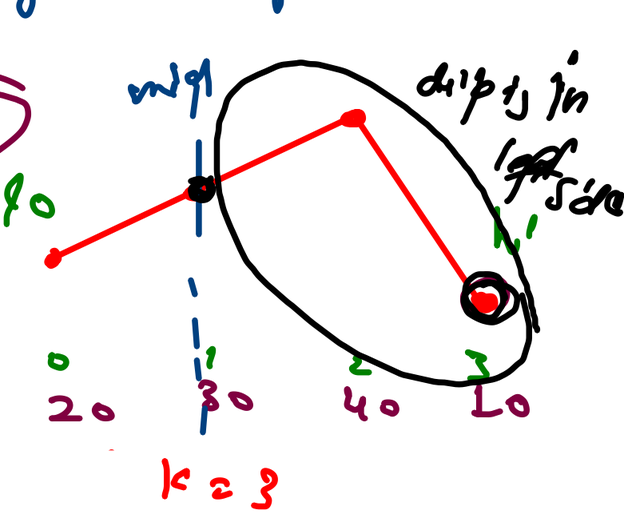
pivot → Smallest Element Time Complexity (searching) → log(n)

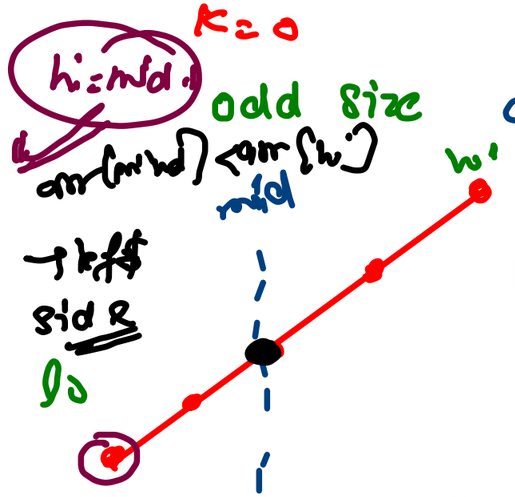
constraint → all no. are unique → Applicable for (Binary Search) (mid splitting)

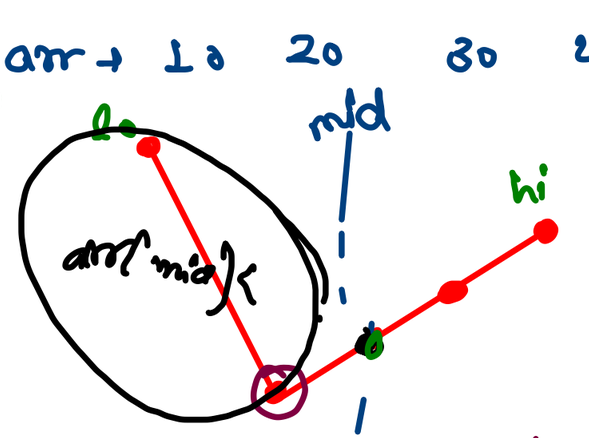


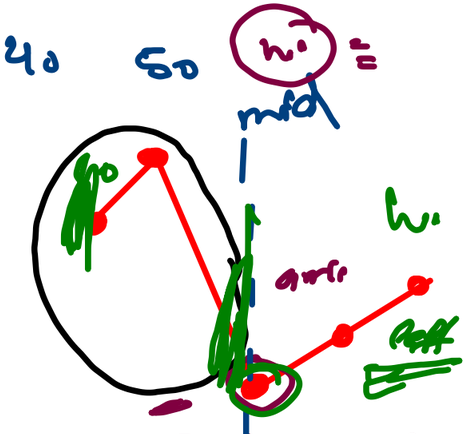


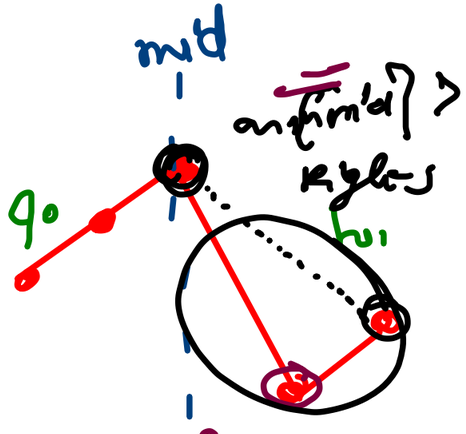


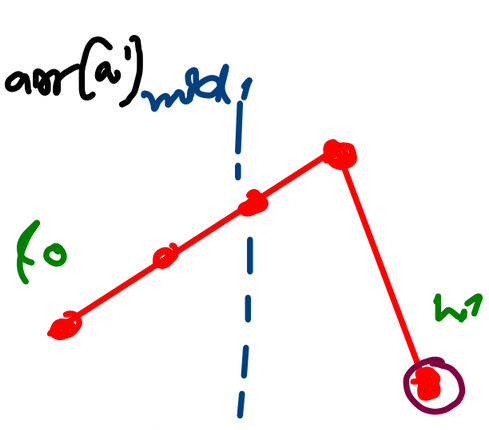












arr → 30 40 50 10 20

index → 0 1 2 3 4

mid = 2 (3)

lo = 3

hi = 3

all elements are

Dip found!

lo == hi

Dip

lo == hi

while (lo < hi)

arr[3] < arr[4] → Right

arr[2] ≥ arr[4] → Dip on Left

mid.

arr[

Complexities: $O(1)$, $O(\sqrt{n})$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$

↑ Arithmatic operations.

Complexity	Reoccurrence	Examples
$O(1)$		
$O(\sqrt{n})$		
$O(\log n)$	$T(n) = T(n/2) + k$	Binary Search
$O(n)$	$T(n) = T(n-1) + k$ $T(n) = 2T(n/2) + k$	Counting, printing array. → fake power.
$O(n \log n)$	$T(n) = 2T(n/2) + cn + k$	Merge Sort, QuickSort (worst case → $O(n^2)$, Best case → $O(n \log n)$)
$O(n^2)$	$T(n) = k(n-1) + T(n-1)$	BubbleSort, Selection, Insertion. (for Best case $O(n)$)
$O(2^n)$		

Binary Search →

$T(1) = 1$

L.H.S. R.H.S.

$T(n) = T(n/2) + k$ ——— (1) eq (1)

Smart power ⇒ $2^6 \{ T(n) = T(n/2) + k$

Replace $n \longrightarrow n/2$.

$T(n/2) = T(n/4) + k$ ——— (2)

Replace $n \longrightarrow n/2$

$T(n/4) = T(n/8) + k$ ——— (3)

Replace $n \longrightarrow n/2$

$T(n/8) = T(n/16) + k$ ——— (4)

$T(2) = T(1) + k$

add all eq's.

new problem is Reduce to find 'x'.

$T(n) = k \cdot x + T(1)$

$T(n) = kx + 1$

$k + k + k + k + \dots + k = kx$

$n/2, n/4, n/8, n/16, \dots, 1$ (x'th index)

$a, ar, ar^2, ar^3, \dots, ar^{n-1}$

$1, 2, 3, 4, \dots, n$

$2, 6, 18, 18\times 3, \dots$

$2, 2\cdot 3, 2\cdot 3^2, 2\cdot 3^3, \dots$

common ratio $= 3$

$ar^{x-1} = 1$

$\frac{n}{2}\left(\frac{1}{2}\right)^{x-1} = 1$

$\frac{n}{2} \times \left(\frac{1}{2}\right)^{x} \times 2 = 1$

$n = 2^x$

take $\log_2$ both side.

$\log_2 n = \log_2 2^x$

$\log n = x \equiv \log n$

$T(n) = k \cdot \log_2 n + 1$

$T(n) = O(\log n)$

$\log_a a^b = b \log_a a$

$= b$

n^{th} term of G.P $= ar^{n-1}$

First term $= a$

Common Ration $= r$

n^{th} index $= n$

Geometric Progression.

complexity analysis for O(n) :

print array from Recursion.

```
T(n)
print(arr, n) {
  (n<=0)
  sysO (arr[n]);   (T(n-1))
  print(arr, n-1)
}
```

Reoccurrence Relation :

$T(n) = k + T(n-1)$ —— (1) Replace n from (n-1)

$T(n-1) = k + T(n-2)$ —— (2) Replace n from (n-1)

$T(n-2) = k + T(n-3)$ —— (3)

⋮

$T(1) = k + T(1)$

$T(n) = k(n-1) + 1 = kn - k + 1 = kn + C$

$T(n) = O(n)$

$T(n) = k \cdot x + T(1)$ → Problem is Reduce to find value of (x)

put value of 'x'

$n-1, n-2, n-3, n-4, \ldots\ldots, 1$

$(n-2) - (n-1)$

$n-2-n+1$

$= -1$

$(n-4) - (n-3)$

$n-4-n+3$

A.P. $d = -1$

(Arithmatic Progression) $d = -1$

	(1)	(2)	(3)	(4)
	a	a+d	a+2d	a+3d
a=2, d=1 →	2	3	4	5
a=2, d=-1 →	2	1	0	-1

x^{th} term.

x^{th} term $= a + (x-1)d$

$= 2 + (x-1)1$

$\therefore 2 + (x-1)(-1)$

$a = n-1$

x^{th} term $= 1$

$d = -1$

$a + (x-1)d = 1$

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

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

$n-1-x+1 = 1$

$x = n-1$

fake smart power :

Pow(a, b){

log(n)

$2^0 T(n) = 2T(n/2) + 2^0 k$ —— (1)

Replace n from $n/2$ & multiply with '2'

$2^1 T(n/2) = 2^2 T(n/4) + 2k$ —— (2)

Replace n from $n/2$ & multiply with '2'

$2^2 T(n/4) = 2^3 T(n/8) + 2^2 k$ —— (3)

$\vdots$

$2^{x-1} T(n/2^{x-1}) = 2^x T(1) + 2^{x-1} k$ —— (i)

add

$T(n) = 2^x + k + 2k + 2^2 k + \cdots 2^{x-1} k$

$T(n) = 2^x + k(1 + 2 + 2^2 + \cdots + 2^{x-1})$

Sum of G.P. $= \frac{a(r^n - 1)}{r - 1}$

$= a + ar + ar^2 + \cdots \to ar^{n-1}$

$1 + 2 + 2^2 + \cdots + 2^{x-1}$

$a = 1$, $r = 2$

$\text{sum} = \frac{1 \times (2^x - 1)}{2 - 1}$

$\text{sum} = 2^x$

$T(n) = 2^x + k(2^x)$

$\boxed{T(n) = 2^x (1 + k)}$

$\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots$ x^{th} term, $\downarrow$

$x = \log n$

$T(n) = 2^{\log_2 n} (1 + k)$

$= n(1 + k)$

$= n \cdot k!$

$T(n) = O(n)$

$a^{\log_a b} = b$

n log n Complexity Analysis:

pivot → mid for quick sort

$T(n) = \underbrace{T(n/2)}_{\text{sorting call}} + \underbrace{T(n/2)}_{\text{sorting call}} + \underbrace{c \cdot n}_{\text{merging}} + \underbrace{k}_{\text{Arithmatic operation}}$

$T(n) = 2T(n/2) + cn + k$ ——— (1) $n \to n/2$ } ×2.

$2T(n/2) = 2^2 T(n/2^2) + 2cn/2 + 2k$ ——— (2) $n \to n/2$] ×2.

$2^2 T(n/2^2) = 2^3 T(n/2^3) + 2^2 cn/2^2 + 2^2 k$ ——— (3)

$2^{x-1} T(n/2^{x-1}) \quad 2^x T(n/2^x) + 2^x c \cdot n/2^{x-1} + 2^{x-1} \cdot k$

pivot → O(n)

n times.

constant quicksort - O(n²)

add

$T(n) = 2^x T(1) + x \cdot cn + k(1 + 2 + 2^2 + \dots 2^{x-1})$

Solve for 'x'

$T(n) = 2^x T(1) + x \cdot cn + k(2^x - 1)$

$\frac{n}{2^x} = 1$

$n = 2^x$

$\log n = x$

Replace 'x' from log n.

$T(n) = 2^{\log_2 n} + n * c * \log n + k(2^{\log_2 n} - 1)$

$T(n) = n + c \cdot n\log(n) + k(n-1)$

$T(n) = n(k+1) - k + c \cdot n\log(n)$ $\quad n\log(n) > n$

$T(n) = O(n\log n)$

Complexity analysis for $O(n^2)$

$$T(n) = K(n) + T(n-1) \quad \text{—} (1)$$

$$T(n-1) = K(n-1) + T(n-2) \quad \text{—} (2)$$

$$T(n-2) = K(n-2) + T(n-3)$$

$$\vdots$$

$$T(2) = K(2) + T(1)$$

$$T(n) = K(\underbrace{2+3+4+\cdots+n}) + C$$

add 1 & subtract 1

$$= \underbrace{1+2+3+4+\cdots+n} - 1$$

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

$$T(n) = K\left(\frac{n(n+1)}{2} - 1\right) + C$$

$$T(n) = K\left(\frac{n^2}{2} + \frac{n}{2} - 1\right) + C$$

$$T(n) = \frac{Kn^2}{2} + \frac{Kn}{2} \underbrace{-K + C}_{K'}$$

$$T(n) = \frac{Kn^2}{2} + \frac{Kn}{2} + K'$$

$n^2 > n$

$$T(n) = O(n^2)$$

add
=