Agenda → Problem of searching

→ Linear Search

→ Binary Search

→ Modified formulae for BS

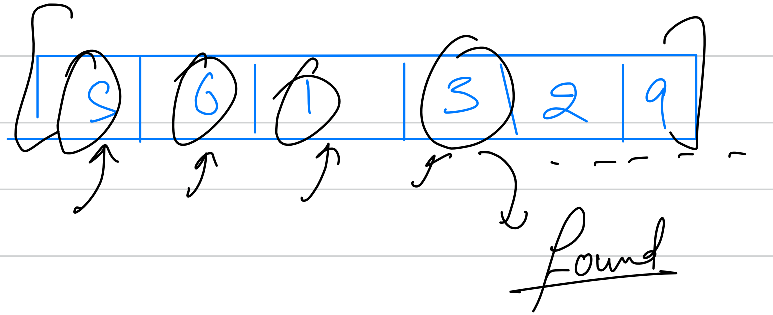
→ Time Complexity

→ Problem Solving.

Problem of searching

You have a target (the element to search)
You have a search space (It is the entire region where we can search for the target)

→ Linear Search → It one by one goes over all the
→
→ elements of the search space & checks if the current element is equal to target or not.

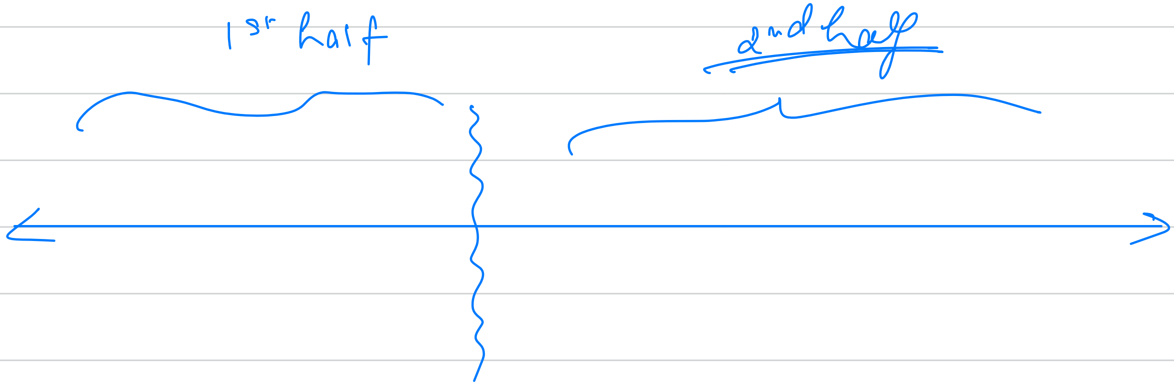


key = 3

Qn what is the time complexity of linear search?

O(N) when N is the size of search space

Any optimizations ??



You can distinguish between elements of first half & 2nd half based on some property.

→ **Binary Search** → Divide you whole search space into 2 equal halves such that first half is Differen 2nd half.

Application

→ Given a list of numbers arranged in ascending order & a target number. Find the position of target in list otherwise if not found return -1.

target = 5

[2, 5, 9, 13, 16, 23, 39, 65]

0 1 2 3 4 5 6 7

first find the middle →

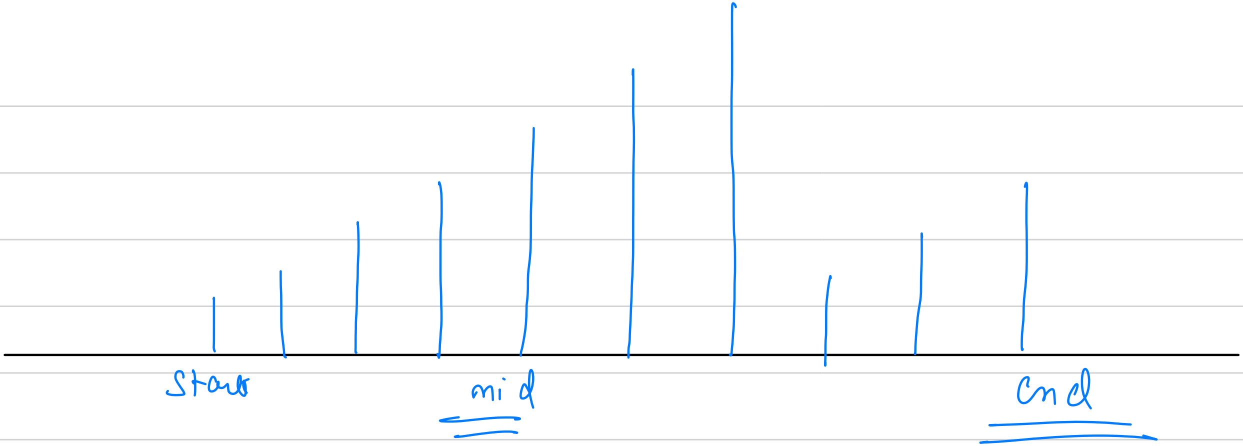
Q) Given a list of numbers which were initially sorted but now have been rotated. You also have a target, find the pos of target.

[4, 5, 6, 7, 0, 1, 2]

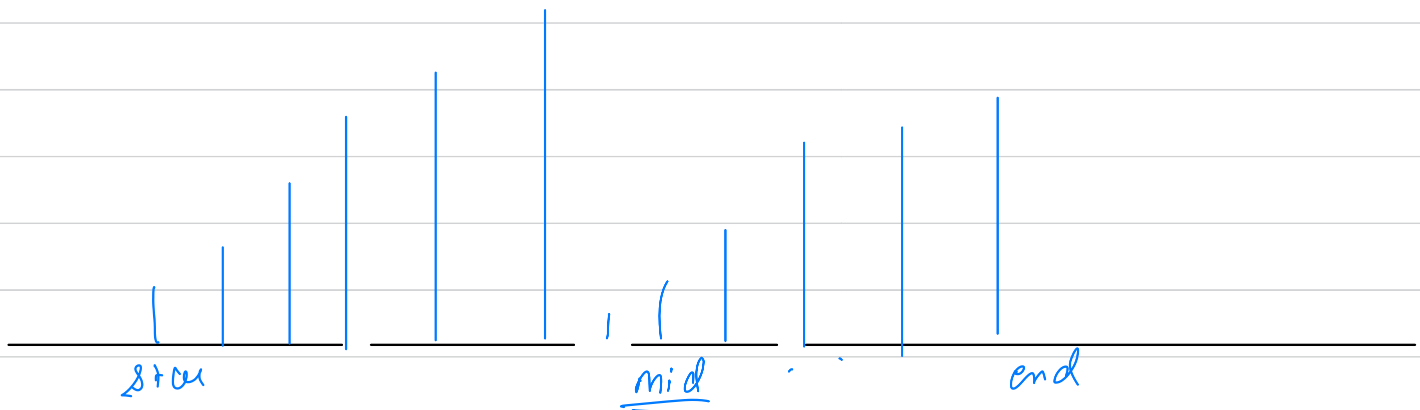
target = 0

$n \leq 10^7$

Case I



Case II



Time Complexity

→ similar

Ternary Search

comparison

$$T(n) = T(n/2) + O(1)$$

$$T(n/2) = T(n/4) + O(1)$$

$$T(n/4) = T(n/8) + O(1)$$

$$\vdots$$

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

K steps

$$T(n) = T(1) + \underline{K \times 1}$$

$k = ??$

$$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \frac{n}{8} \text{ - - - - - - - } \frac{n}{2^k}$$

last iter

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

$$n = 2^k$$

$$k = \log_2 n$$

total cases

$$T(n) = T(1) + k \times 1 \Rightarrow \underbrace{T(1)}_{\text{const}} + \underbrace{\log_2 n \times 1}_{n}$$

$$\Rightarrow O(\log_2 n)$$

$n \longrightarrow \frac{n}{3} \longrightarrow \frac{n}{9} \longrightarrow \frac{n}{27} \longrightarrow \frac{n}{81} \text{ - - - - } \frac{n}{3^k}$

$O(\log_3 n)$

$n \longrightarrow \frac{n}{2}$

$\lceil n \rightarrow \frac{n}{2^k}$

$O(\log_2 n) \longrightarrow O(\log_3 n)$

$64 \Rightarrow 2^6$

$\log_2 n$ is bigger

$81 = 3^4$

a) linear $\rightarrow$ $(n-1)$

b) binary $\rightarrow$ $n/2$ ✓

c) ternary $\rightarrow$ $n/3$

d) None

() $\rightarrow$ avoids constant term, IO effects

$$T(n) = \cancel{T(n/3)} + O(2)$$

$$\cancel{T(n/3)} = \cancel{T(n/9)} + O(2)$$

$$\cancel{T(n/9)} = \cancel{T(n/27)} + O(2)$$

$$\vdots$$

$$\cancel{T(3)} = T(1) + O(2)$$

$$T(n) = T(1) + \underline{2 \times k}$$

ten

K steps

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

$$k = \log_3 n$$

$$T(n) = T(1) + \underbrace{\log_3 n \times 2}_{\text{Total no of comparison}}$$

$2 \log_3 n$ =

$1 \times \log_2 n$

$$2 \times \frac{\log_2 n}{\log_2 3}$$

$\log_2 n$

lo hi

mid → $\boxed{\frac{lo + hi}{2}}$ → problematic

low + hi → overflow

Devis

$$lo + \frac{hi - lo}{2}$$

$$\frac{lo + hi}{2}$$

$$\frac{lo + hi + lo - lo}{2} \Rightarrow \frac{2lo + hi - lo}{2}$$

$$\Rightarrow lo + \frac{hi - lo}{2}$$

Q:- Given a number n, find integer part of squareroot of the number.

What is the range in which squareroot of n will lie?

range $\Rightarrow$ $(1 \longrightarrow n-1)$

R, J, U

$n-1$

mid

if $mid * mid > n$

1 5 ✗ 10

mid => 5

$5 * 5 == 10$ ✗

$5 * 5 > 10$ ✓

1 ✗ 4

2

$2 * 2 == 10$ ✗

$2 * 2 > 10$ ✗

$(3, \ldots\ldots)^2 == 10$

Pos

✗

3 4

$3 * 3 == 10$ ✗

$3 * 3 > 10$ ✗

$3 * 3 < 10$ ✓

HW 3rd class

Qn Given a +ve integer value 'n', find the squareroot upto 6 decimal precision

n => 36 → 6.000000

37 => 6.