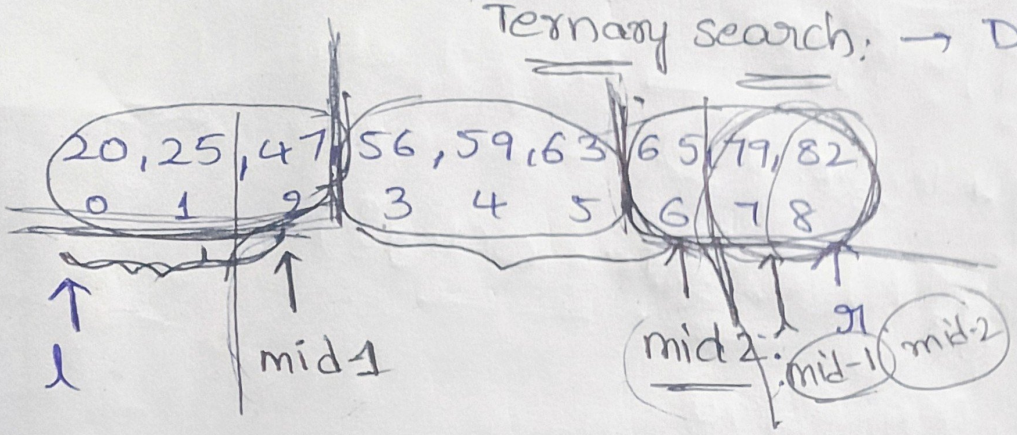
Ternary search: → Dividing array into three halfs.



l=0

r=8 0 + (8-0)/3 = 2

mid 1 = l + (r-l)//3

mid-2 = r-(r-1)//3

8-(8-0)//3

8-2 (6)

Key = 79

ternary Search (arr, l, r, key):

while (if arr[mid1] == key

i<=j: return mid1

(82)

mid-1 = 7 + (8-7)//3

= 7

mid2 = 8

if arr[mid2] == key

return mid2

if key < arr[mid1]: ↗ 1st part

return ternary Search (arr, l, mid1-1, key)

elif key > arr[mid1]: ↗ 3rd part

return ternary Search (arr, mid2+1, r, key)

↗ 2nd part 7 8

else:

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

return -1:

Time Complexity: $O(\log_3^n)$.

Recurrence Relation: of Ternary search:

$$T(n) = T(\frac{n}{3}) + c$$

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

$$= T(\frac{n}{3^3}) + c + c + c \quad = T(\frac{n}{3^3}) + 3c$$

$\}$ k-times

$= T\left(\frac{n}{3^k}\right) + kc$

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

$\therefore k = \log_3^n$

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

$= T\left(\frac{n}{n}\right) + \log_3^n c.$

$= c \cdot \log_3^n$ $\quad \therefore TC = O(\log_3^n)$

Interview:

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

lower the base, higher the value.

Assignment problem:

Binary Search is more preferrable as comparable to ternary Search? Why?

1. The no of comparisions in Ternary in each pass → 4
   Whereas, in Binary Search it's only - 2

Eventhough TernarySearch reduces the problem size faster it needs more Comparions per step.

2. Binary Search is easier to implement & reason about

∴ For searching in Sorted arrays/lists → binary Search is preferable.

For finding extrema of mathematical functions → ternary Search can be used.