Page :
Date :

BINARY SEARCH

⇒ Before talking about Binary Search lets talk about linear search

Consider an Array :-

i =	0	1	2	3	4	5
	3	7	0	-2	5	9

key = (5)

worst case - n comparison
If 1000 elements then 1000 comparison in worst case.

T.C = O(n) — worst case

T.C = O(1) — Best case.

⇒ Binary Search :-

condition :- element should be in monotonic function.

Ex :-

	0	1	2	3	4
	3	5	9	13	27

mid

key = 13

$m = \frac{0+4}{2}$ mid ≠ key

Key > mid

move to right part

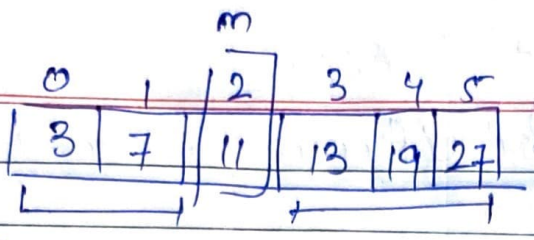
	3	4
	13	27

$m = \frac{3+4}{2} = 3$

13 = 13 — True

ans = 3

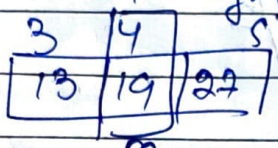
steps :-
1. find mid
2. compare mid & key
3. if equal return index
4. Decide part.



$m = \frac{0+5}{2} = 2$

$27 > m$

move to right.



$m = \frac{3+5}{2} = 4$

$27 > m$

again move to right

$m = \frac{5+5}{2} = 5$

$m = 27$

$m = key$

$27 = 27 \Rightarrow$ True

ans = 5

Benefit of using Binary Search over linear Search

Binary Search - T.C $\Rightarrow$ O(log n)
linear Search - T.C $\Rightarrow$ O(n).

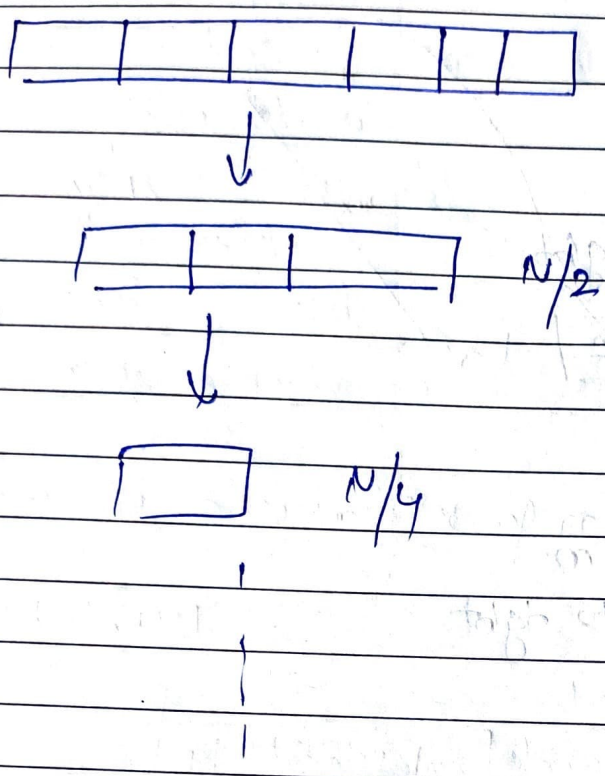
→ L.S → 1000 values in array $\Rightarrow$ 1000 comparisons.

→ B.S → 1000 value in sorted array → 500 – 250 – 125 – 62 – 31 – 15 – 7 – 3 – 1 – 0

10 comparisons

Page :
Date :

lets find T.C for B.S



$\frac{N}{2^K} = 1$

$N = 2^K$

$K = \log N$

$\therefore \boxed{TC = O(\log N)}$