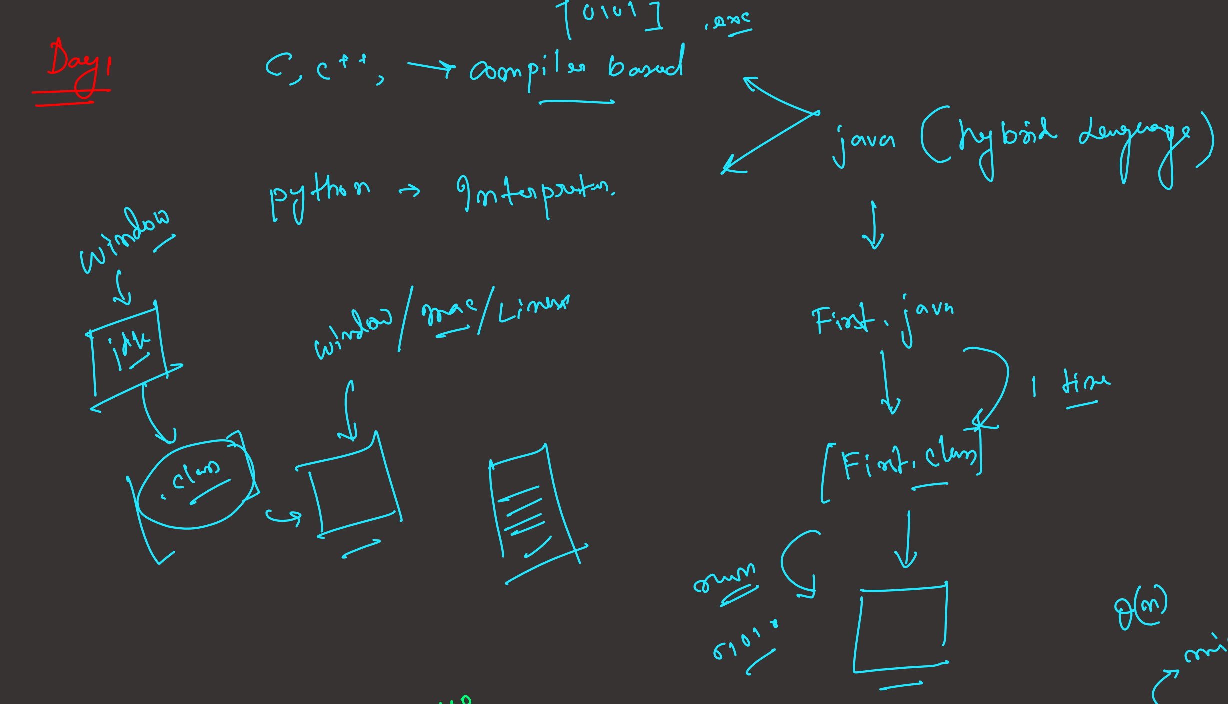


θ(n)

mid [avg] case

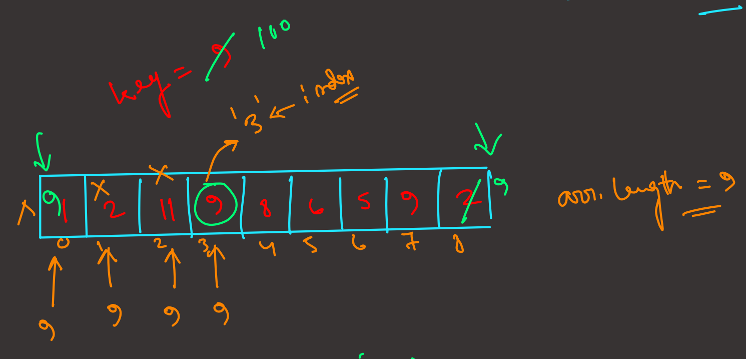
Ω(1)

key = first position O(1)

↓ last index → O(n)

not in array

Searching



arr → [n]

i++

for(int i=0 to arr.length) {
searching;
}

i--

for (i=arr.length-1 to 0) {
Searching;
}

Ω < θ < O

O(n)

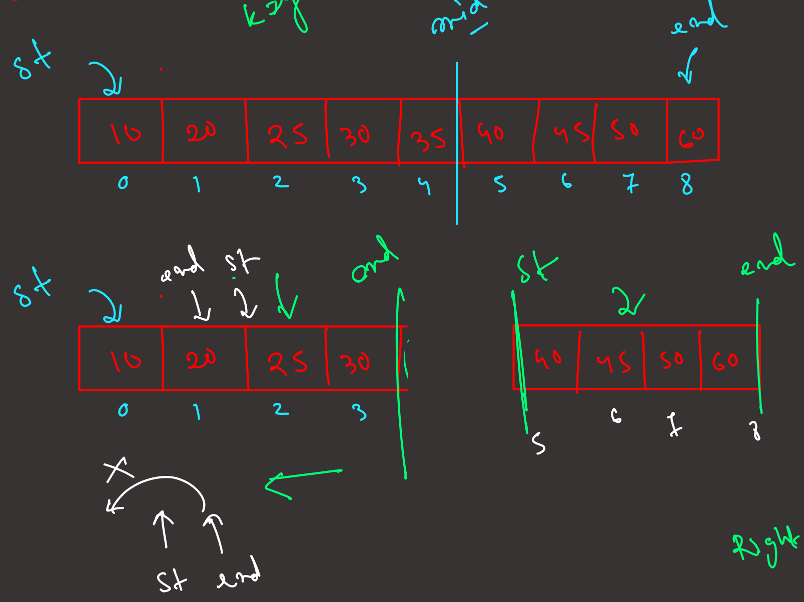
O(n log n)

arr (sorted)

{Arrays
Collections}

Binary Search → arr (sorted)

key = 25



(12)

$\to 7/2 = 3.5 = 3$

$mid = \frac{st + mid}{2}$

while(st ≤ end) $= \frac{0+8}{2} = 4$

index

if (arr[mid] == key) → T

return mid

Right half → if(arr[mid] < key)

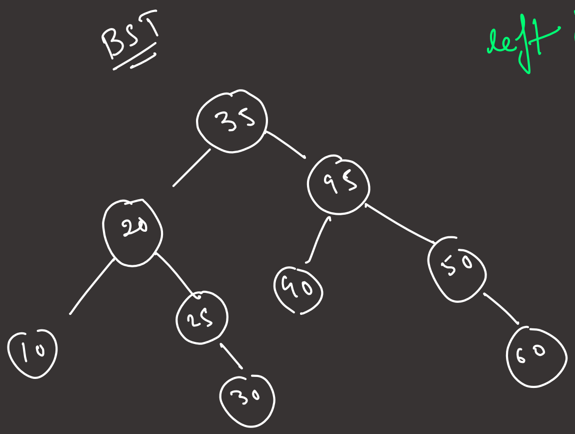
st = mid+1;

arr[mid] = 20 < key → RH

{25, 36}
  2    3

mid = $\frac{2+3}{2}$ = 2.

arr[mid] = 25 = key

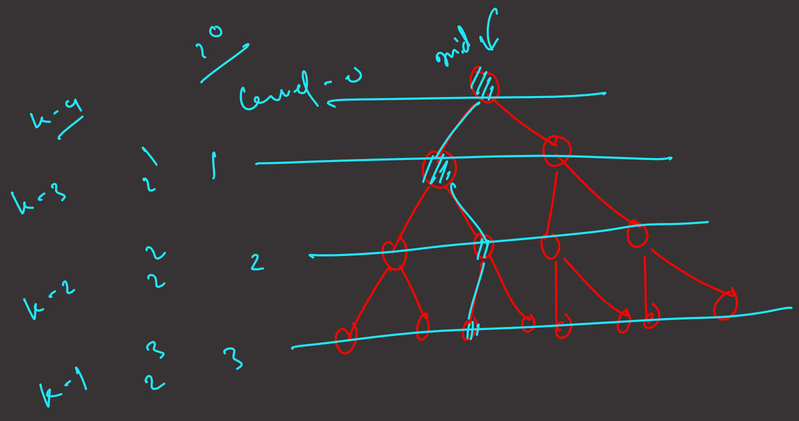


← Searching

T.C = O(log n)

left half if (arr[mid] > key)

end = mid - 1 ;



1+2+4+8 = 15

$\lceil \log_2(15) \rceil = 4$

total no of node

element

$2^k - 1 = n$

$2^k = n+1$

$k = \log_2(n+1)$

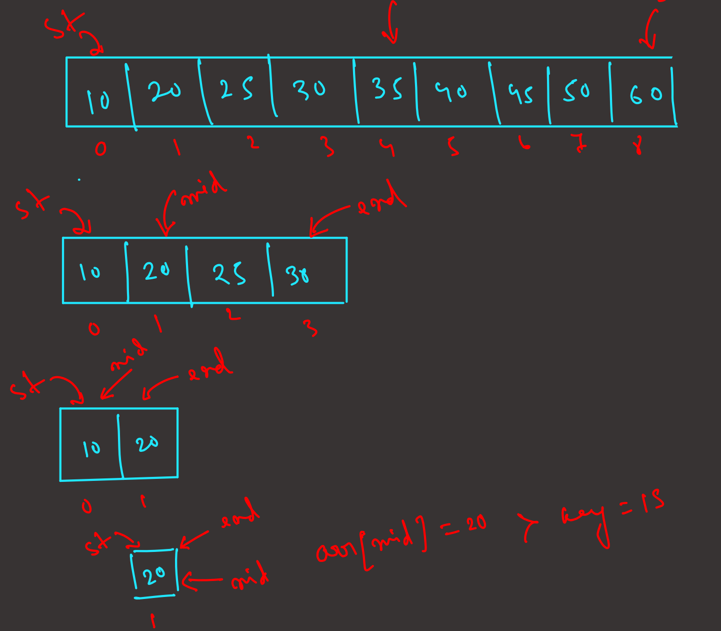
$= O(\log n)$

recursion

$1 < \log n < n < n\log n < n^2 < n^3 < n^n$

key = 15

mid = (st+end)/2



arr[mid] = 20 > key = 15

← left half

{end = mid - 1 = 1-1 = 0}

st = 1

end = 0

st <= end X

return = -1

H.W

key = 42

① Recursion → Binary Search (Array)

② Fibonachi Series → while / for loop , Recursion

{0 1 1 2 3 5 8 13, 21, ... }

③ key = 92 ← check Binary search.