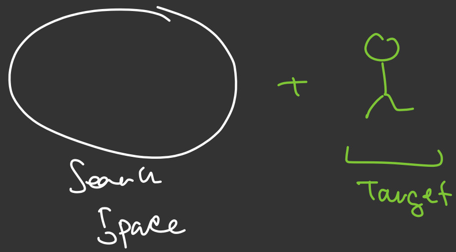
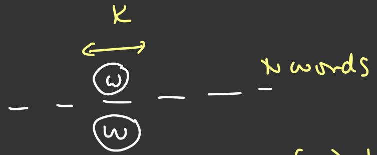
Binary Search

Searching



Word in Newspaper — (unsorted) [Newspaper] + [word] → Linear Search



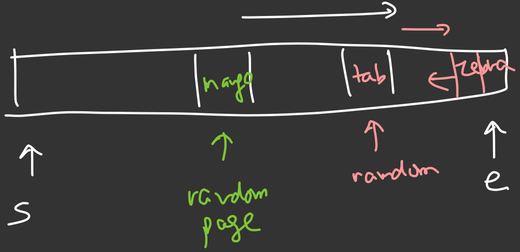
O(N) time

word in Dictionary — [Dictionary] (sorted) + [word]

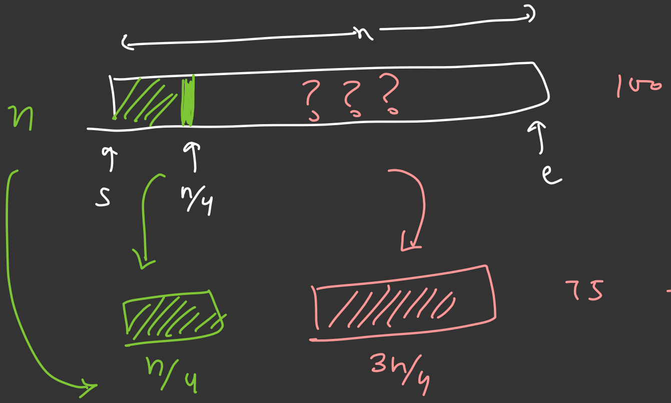
or name in Contact list

[app apple applet ... batman .... mango----- xmas --- zebra]

Better = Yes



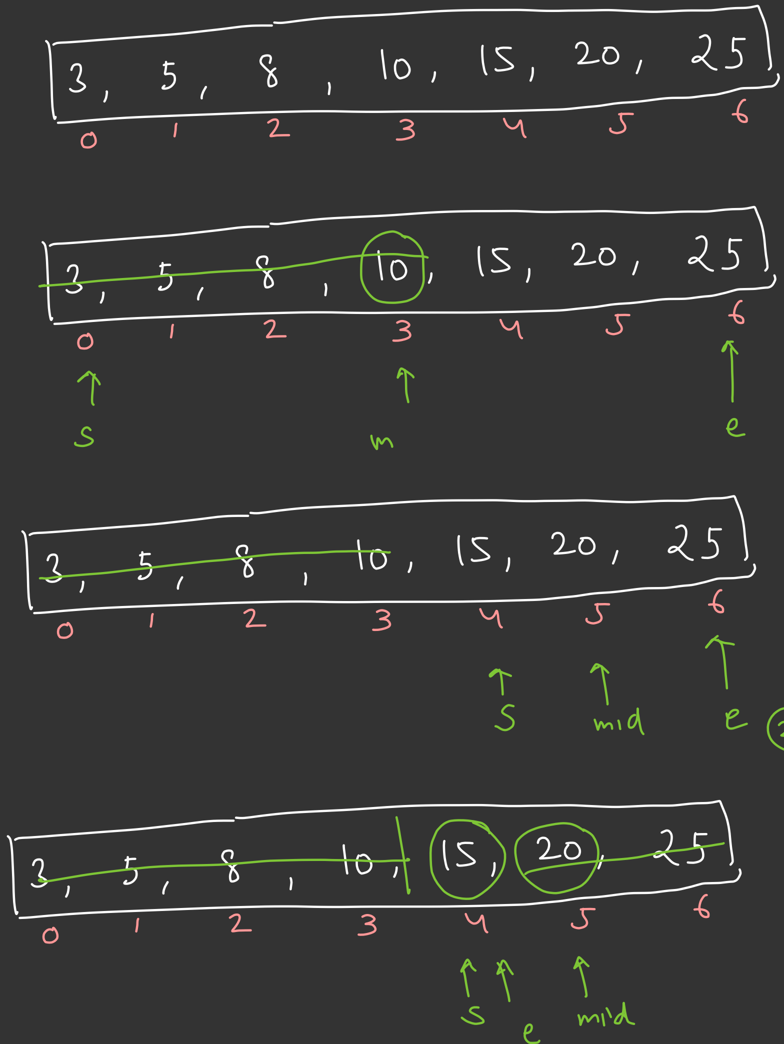
tree



75 → Unbalanced Split

Better → Divide at the middle

Sorted Array



Target = 15

① $\underline{10} < \underline{15}$

a[mid] < T

↳ S = mid + 1

② a[mid] > T

20 > 15

↳ e = mid − 1

S = 4

e = 4

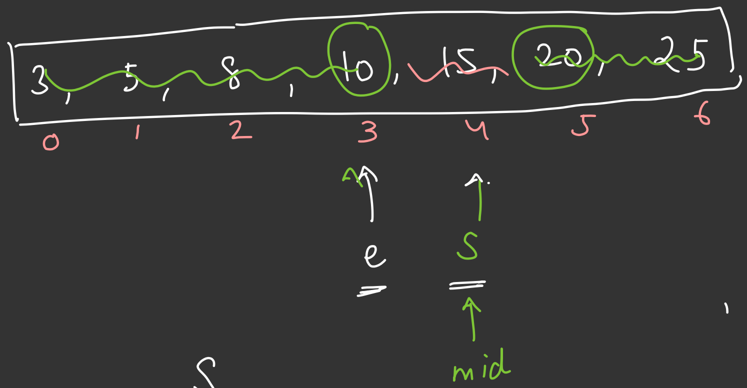
mid = 4

③ [arr[mid] == T

=> return mid

Element is Not present

$\boxed{T = 14}$

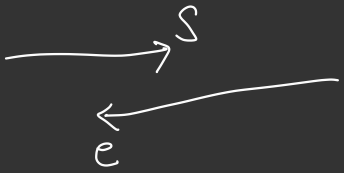


10< 1~~5~~4 Right s = mid + 1

~~20<~~

20> 14 e = mid - 1

15 > 14 => e = mid - 1



→ if(s>e)
return -1;

Pseudocode :

sorted

```
int binarySearch( int arr[] , int T ) {

    int s = 0 ,
    int e = arr.length - 1 ;

    while ( s<=e ) {
        mid = (s+e)/2
        if ( arr[mid] == T ) { return mid; }
        else if ( arr[mid] > T ) { e = mid - 1; }
        else { s = mid + 1; }
    }

    return -1;
}
```

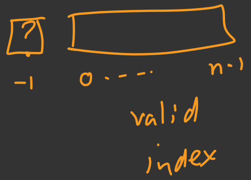
loop will stop if s>e

log N

O(1)

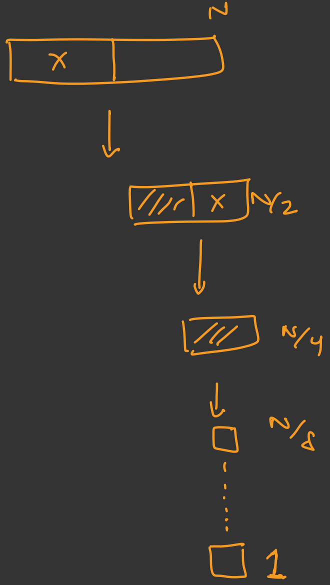
= O(Log N)

T = 10



Complexity

Visual way



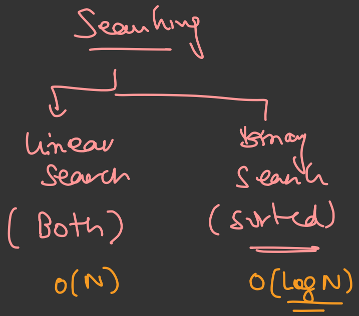
Step	Size
0	N
1	$N/2 = \frac{N}{2^1}$
2	$N/4 = \frac{N}{2^2}$
3	$N/8 = \frac{N}{2^3}$
⋮	
K	$\boxed{1 = \frac{N}{2^K}}$

$$\Rightarrow N = 2^K$$

$$\Rightarrow \log_2 N = \log_2 2^K$$

$$\Rightarrow K \log_2 2 = \log_2 N$$

$$\Rightarrow \boxed{K = \log_2 N}$$


Searching

Linear Search (Both) O(N)

Binary Search (sorted) O(Log N)

Substitution Method

n

$$\Rightarrow T(n) = k + \cancel{T(N/2)}$$

$$\Rightarrow \cancel{T(N/2)} = k + \cancel{T(N/4)}$$

$$\Rightarrow \cancel{T(N/4)} = k + \cancel{T(N/8)}$$

$$\vdots$$

$$\cancel{T(1)} = k$$

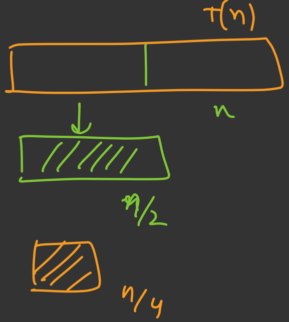
1

$$T(n) = \underbrace{k + k + k + \ldots\ldots k}$$

$$= k \cdot \log N$$

$$= O(\log_2 N) \quad \text{Time}$$

$$= O(1) \quad \text{Space}$$



$$\underbrace{n \rightarrow n/2 \rightarrow n/4 \rightarrow n/8 \dashrightarrow 1}_{\log N}$$

Doubt

Binay Search in Dictionary

(app apple bat batman ---- man mango ----- zebra)

s mid e

N = 50,000

Largest word · 100 chars

constant = c

T = "mango"

~~N log N~~

$O(c \log N)$

$\Rightarrow O(\log N)$

man ?== mango

String.compare(,)

linear in word length = c

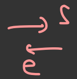
-1 0 +1

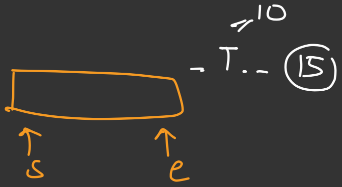
less eq greater

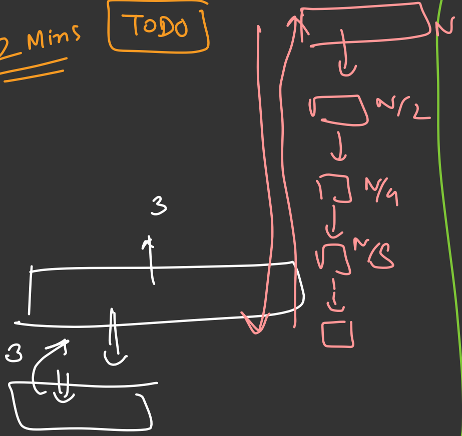
Recursive Binary Search

2 Mins TODO

```
int binary Search (arr, T, s, e ) {
  // Base Case
    if (s > e) {
        return -1;
    }

  // Rec Case

    mid = (s+e)/2

    if (arr[mid] > T) {
⇒       return binarySearch (arr, T, s, mid-1);
    }
    else if (arr[mid] == T) {
        return mid;
    }
    else {
        return binarySearch (arr, T, mid+1, e);
    }
}
```

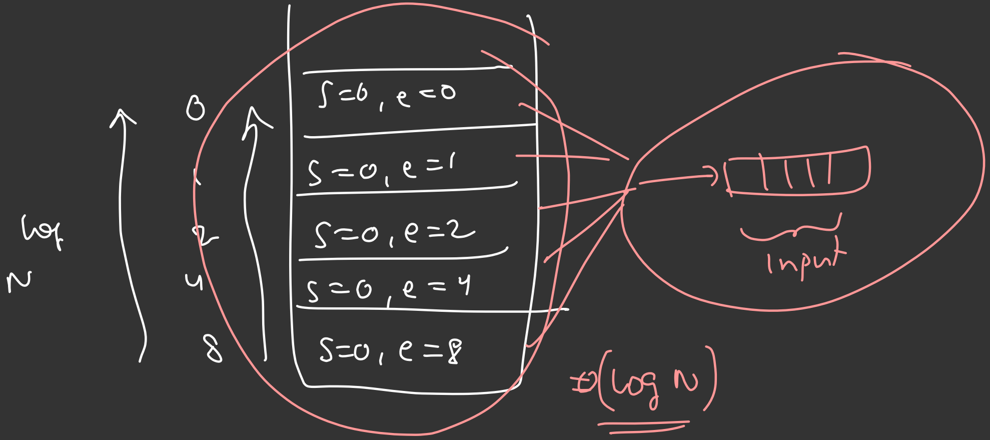




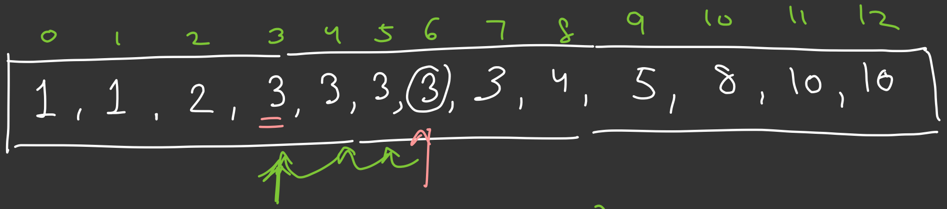


Time → O(logN)

Space → $\underbrace{O(\log N)}_{\text{stack Memory}}$


log N
0
1
2
4
8
s=0, e=0
s=0, e=1
s=0, e=2
s=0, e=4
s=0, e=8
Input
Θ(log N)

Binary Search on an array with duplicates



T=3

⇒ first occ

● → ~~Move linearly~~ → O(N) ← 3 3 3 3 3 3 3 3 3 3

→ Apply Binary Search in left & save current mid

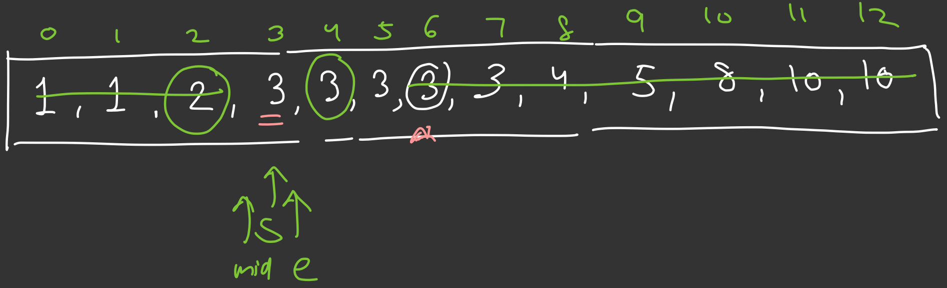
s=0

e=12

mid = $\boxed{6}$

Binary Search on an array with duplicates

	mid	ans
s=0, e=12	6	6
s=0 e=5	2	6



ans = 6

S=3 e=5 4 4 e =mid-1

s =3 e =3 3 3 e = mid -1

s=3 e=2 Stop

e =mid - 1

first occurence

```
int  binarySearch( int arr[] , int T){

      int s =0 ,
      int e = arr.length-1;
   int ans = -1;
      while(s<=e){
           mid = (s+e)/2
           if( arr[mid]==T) { ans = mid, e =mid-1 }
           else if (arr[mid] >T){ e = mid - 1; }
            else     { s = mid+1; }
      }

   return ans;

}
```

⇒ Amazon ~~Interview~~ Online Coding Round Prob

Time ↑ Accuracy

0 1 2 3 4 5

1, 1, 3, 3, 3, 3, 5, 7, 8, 10, 12

first 2

last 5

T = 3

⇒ frequency of T

○ linear Search → O(N), O(1) Space } unsorted array + 1 Query

○ Hashmaps →

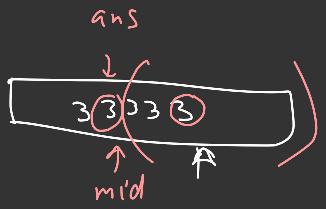
1	2
3	4
5	1
7	1
8	1
10	1
12	1

} O(N) time, O(N) Space } Multiple Queries O(N + Q)

Q → No of Queries

○ Binary search $last - first + 1 = \boxed{4}$] O(log N), O(1) space → 1 Query → O(Q log N)

```
          lastOcc
int       binarySearch ( int arr[] , int T) {

    int s =0 ,
    int e = arr. length-1 ;
  int ans = -1;
    while (s<=e) {
        mid = (s+e)/2
        if ( arr[mid] = = T)  { ans = mid ,  s=mid+1 }
        else if (arr[mid] > T) { e = mid - 1;  }
         else      {  s = mid+1; }
    }

  ⇒  return ans;
}
```

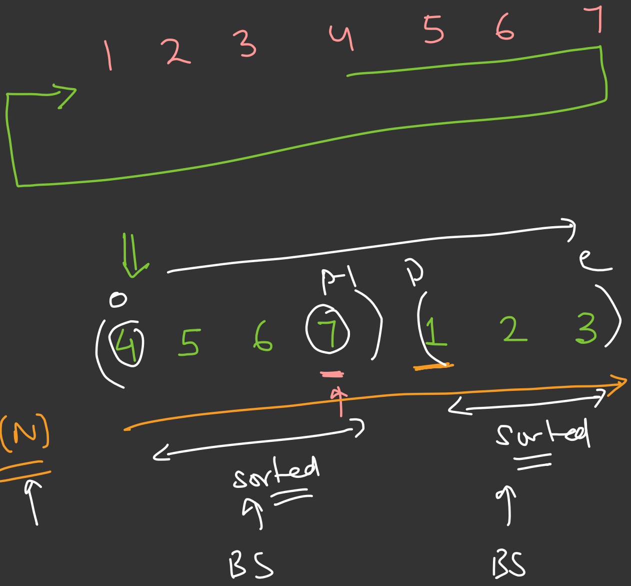


10.31

Q Sorted array which is K rotated, where K is not Known.

10 31

arr →



Linear Search (Largest/Smallest) O(N)

logN i) Find out the (pivot element)

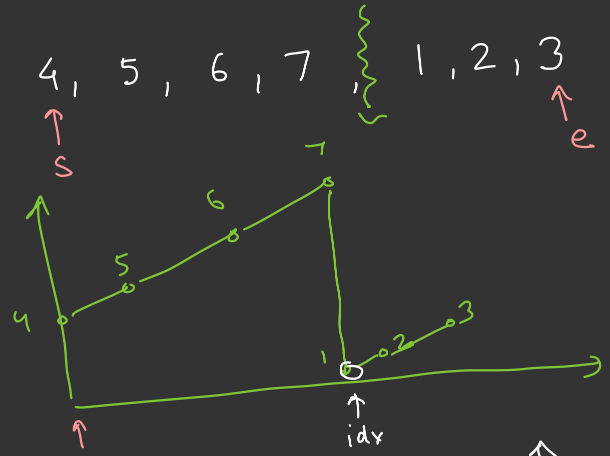
logN ii) Find out index of any element.

(0, n-1)

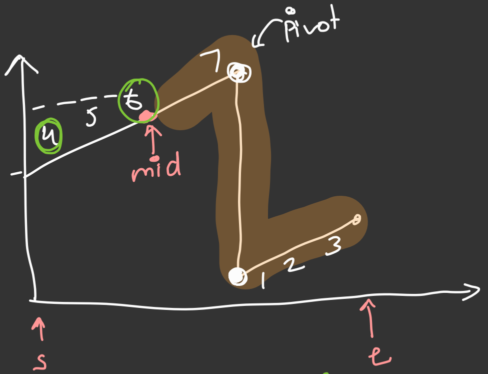
O(2 logN) = O(logN)

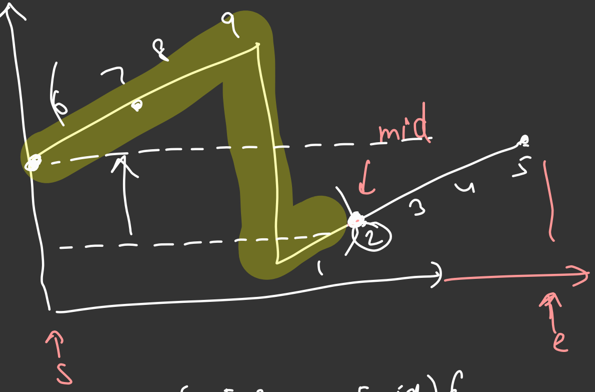
BS (0 — p-1)

BS (p — e)



Log N time





Case-I

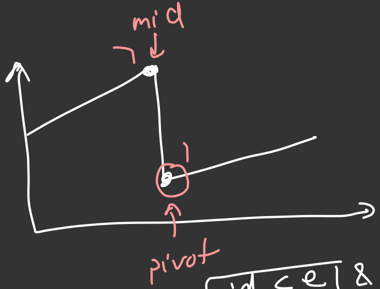
```
if ( a[s] < a[mid] ){
    binarySearch( mid+1, e)
}
```

Case II

```
if(a[s] > a[mid]){
    binarySearch(s, mid-1)
}
```

Rec Case

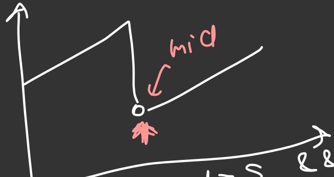
Case-III



mid < e &&

if (arr[mid] > arr[mid+1]) {
    return mid+1
}

Case-IV



mid > s &&

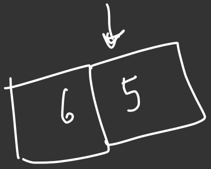
if(arr[mid] < arr(mid-1)]

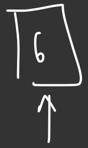
return mid,

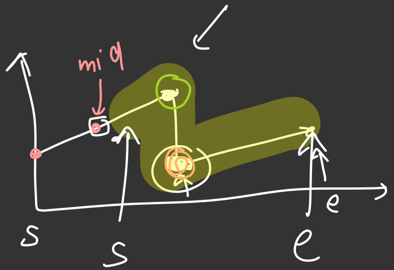
Spl case

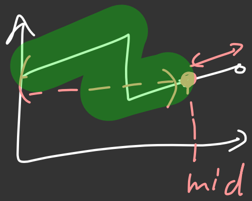
```
s = 0
e = n-1

while (s <= e) {

    mid = (s+e)/2 ;

    if ( mid < e && a[mid] > a[mid+1] )
        return mid+1
    else if ( mid > s && a[mid] < a[mid-1] )
        return mid,

    else if ( arr[mid] > arr [s] ) {
        s = mid + 1

}
```

out of bounds

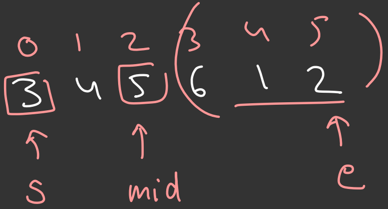


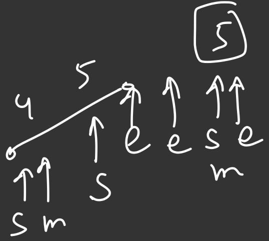


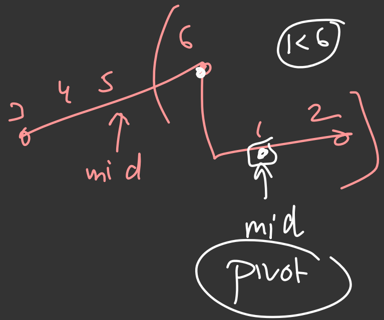


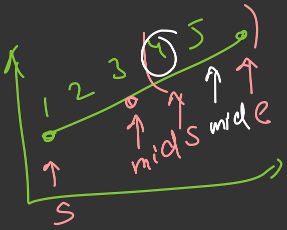


```
        else (
              e = mid - 1 ,
        }
    }
    return (1);
}
```

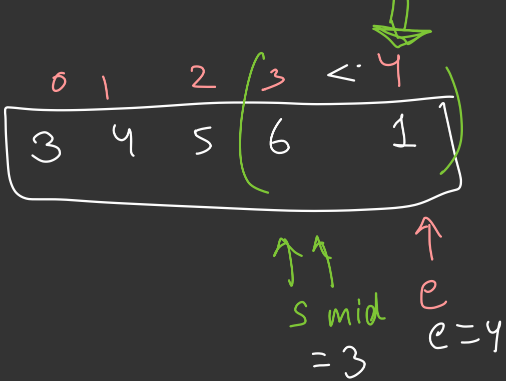


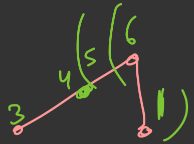






5 > 3





1 2 3 4 5

Physics

Pivot a point of Rotation