Today's Content:

Searching Basics

Why mid at half?

Problems:

→ Search in sorted array

→ Finding floor in a sorted array

→ Finding the 1st occurence in sorted array

→ Search in 2D Sorted Matrix

→ Search in a Rotated Sorted Matrix

Search Space:

→ (Item/Pos) ⟶ police → (Target): What to Search

→ (Search Space): Where to Search

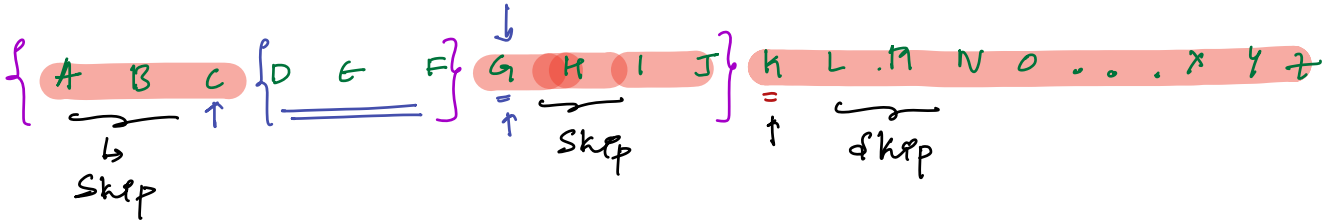
For Example:

word ⟶ {DICT / Book / News Paper}

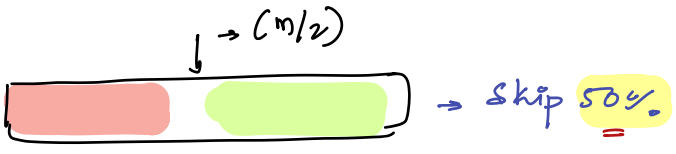
phonenumber ⟶ {Contacts / phone book}

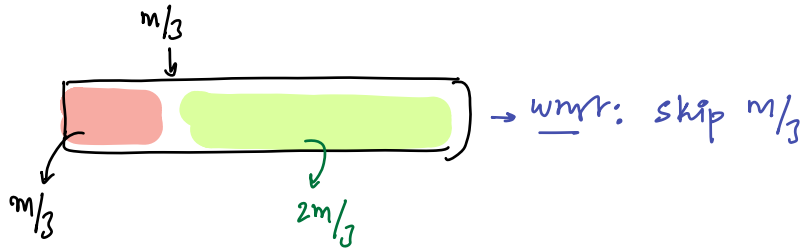
{Search space is reduced, Searching Becomes Easy}

→ Dog:




→ Load at mid?




Every time half of Search space is reduced

Idea: Binary Search

Binary Search : → Divide Search Space into 2 parts, & neglect 1 half of search space by using some condition.

Target Search Space :

Doubt: After dividing Search Space,

If we cannot neglect 1 half of space can we apply binary search : NO {Read flag}

28) Given a sorted array with distinct elements, search for index of an element k, if k is not present return -1

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| ar[10] : | 3 | 6 | 9 | 12 | 14 | 19 | 20 | 23 | 25 | 27 |

**Idea:**

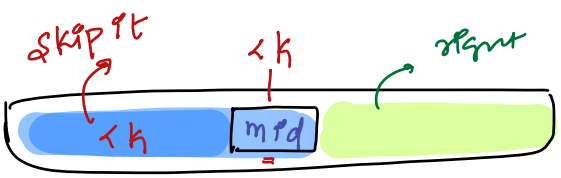
Iterate on array & Search for k.

TC: O(N)

SC: O(1)

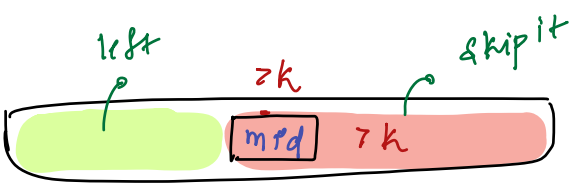
**Idea2:**
- Target : k
- Search Space : { Entire Array }


Case 1: ar[mid] == k return mid



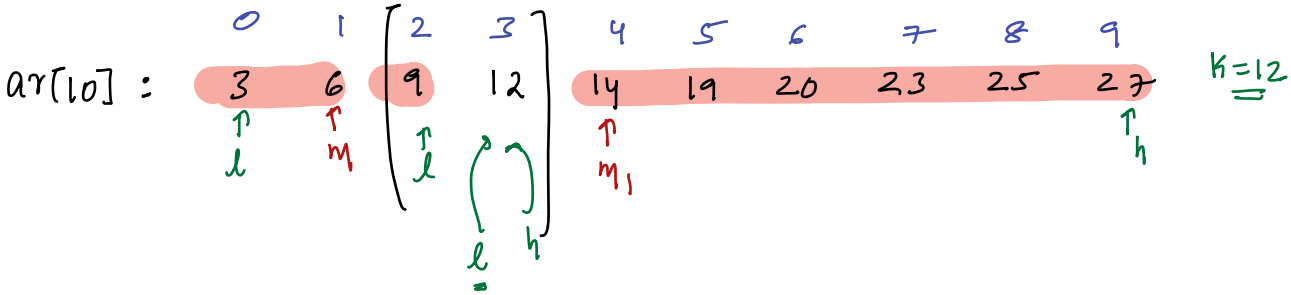

Case 2: ar[mid] < k :

Search in right




Case 3: ar[mid] > k :

Search in left

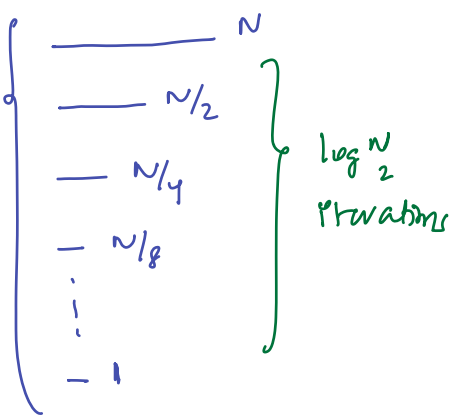



| l | h | m = (l+h)/2 | |
|---|---|---|---|
| 0 | 9 | 4, | ar[m] > k, goto left : h = m-1 |
| 0 | 3 | 1, | ar[m] < k, goto right : l = mid |
| 2 | 3 | 2 | ar[2] < k, goto right : l = m+1 |
| 3 | 3 | 3 | ar[3] == k, return m; |

```
int search(int ar[], int N, int k){
    l = 0, h = N-1;
    while( l <= h ) {
        m = (l+h)/2
        if (ar[m] == k) { return m }
        if (ar[m] < k ) { // goto right
                          l = m+1 }
        else { // goto left
               h = m-1 }
    }
    return -1
}
```

TC: $\longrightarrow O(\log_2 N)$

SC: $O(1)$



28) Given a sorted array, find floor of a given num k

↳ greatest ele $\le k$ in ar[]

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[9] : | -5 | 2 | 3 | 6 | 9 | 10 | 11 | 14 | 18 |

k = 12

k = 5 : 3

k = 4 : 3

k = 10 : 10

k = 20 : 18

k = -7 : -5*

↳ INT_MIN

TC: O(N) SC: O(1)

Idea: ans = INT_MIN = 9

= -5 = 10

= 2 = 11

= 3

= 6

{return ans}

11

Iterate in array

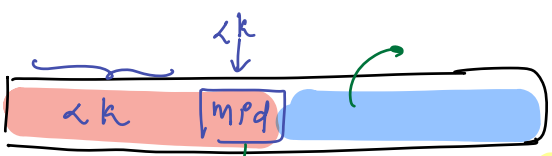
update ans,

look for a

better ans

target : (max $\le k$) Search Space : whole array


Case-I : ar[mid] == k : {return k}

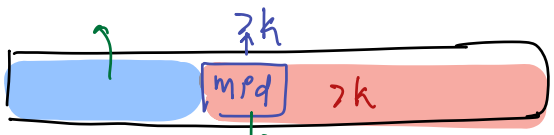



Case-[1]

ar[mid] < k : ans = ar[mid]

goto right

l = m+1




Case-[1]

ar[mid] > k : goto left

h = m-1

Tracing:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[9] : | -5 | 2 | 3 | 4 | 9 | 10 | 11 | 14 | 18 |

k = 5

ans = INT_MIN

| l | h | m | |
|---|---|---|---|
| 0 | 8 | 4 : | ar[4] > k : goto left ; h = m - 1 |
| 0 | 3 | 1 : | ar[1] < k : { ans = ar[1] = 2, goto right ; l = m + 1 |
| 2 | 3 | 2 : | ar[2] < k : { ans = ar[2] = 3, goto right ; l = m + 1 } |
| 3 | 3 | 3 : | ar[3] < k : { ans = ar[3] = 4, goto right ; l = m + 1 } |
| 4 | 3 | → (Break) | |

return ans; [4]

Pseudocode:

```
int floor(int ar[], int k){
    l = 0, h = N-1, ans = INT_MIN
    while(l<=h)
        m = (l+h)/2
        if(ar[m] == k){ return k }
        if(ar[m] < k){ ans = ar[m]
                       l = m+1 }
        else{ h = m-1 }
}
return ans;
```

TC: $O(\log N)$

SC: $O(1)$

10: 3pm

10: 38 hrs

38) Given a sorted array of N elements, find the first occurence index of given ele : k

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -5 | -5 | -3 | 0 | 0 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 8 | 10 | 10 | 15 | 15 |

Idea : Iterate & get first occurence index

TC : O(N) SC : O(1)

first occ index

0 : 3

5 : 7

10 : 15

2 : -1

Target : 1st occurence index of k

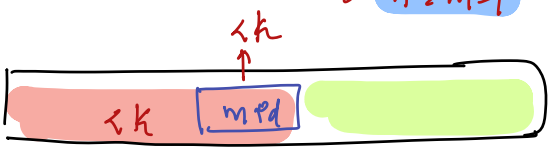
Search Space : Entire array


Case : 1 : ar[mid] == k

: ans = mid

: goto left, Better

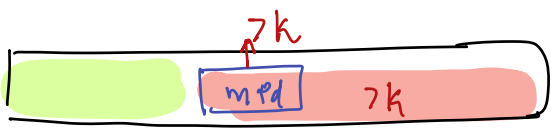
: h = m-1




Case : 2 : ar[mid] < k

: goto right

: l = m+1

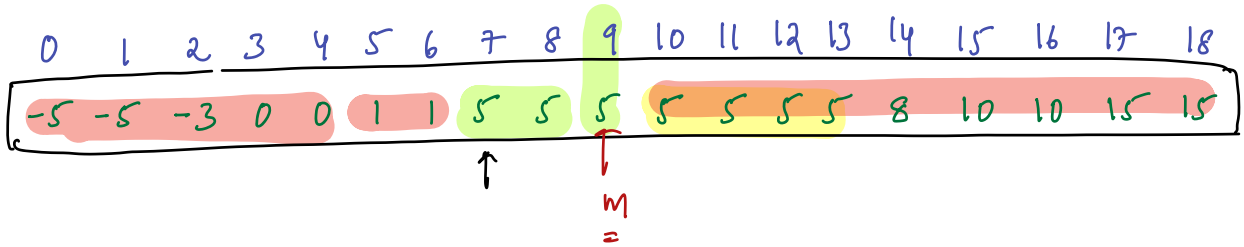



Case 3 : ar[mid] > k

: goto left

: h = m-1

Tracing




ans = -1;

| l | h | m | |
|---|---|---|---|
| 0 | 18 | 9, | if (ar[m] == k) { ans = mid; goto left; h = mid-1 } |
| 0 | 8 | 4, | if (ar[m] < k) { goto right; l = mid+1 } |
| 5 | 8 | 6, | if (ar[m] < k) { goto right; l = mid+1 } |
| 7 | 8 | 7, 2 | if (ar[m] == k) { ans = mid, ans = 7, goto left, h = mid-1 } |
| 7 | 6 | : { l > h } | → Break → { return ans } → (7) |

→ Code: (TODO)

→ Find last occurrence index of k in sorted array {TODO}

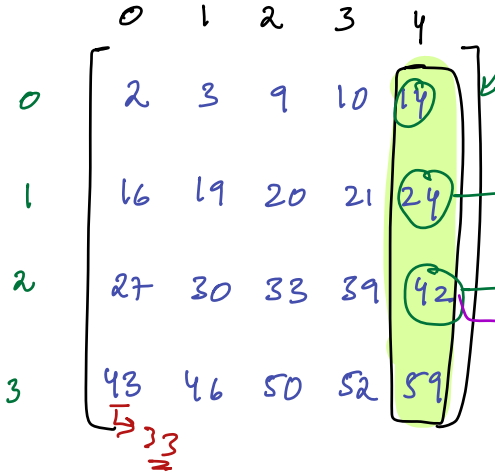
↳ Assignments:

4Q) Given a 2D sorted matrix, Search for k

N*M

→ Note: In a row : M Elements

In a col : N Elements




→ Smallest element >= k

k=10 → 14, k=21 → 24, k=33 → 42 } ceil

Idea1: Search entire Matrix

TC: O(N*M) SC: O(1)

Idea2: Elements can only be in a single row,

→ for every row compare its last element with k, based on that we can apply BS in a particular row

TC: N * log M (log base 2) ↷ Apply BS on a single row.

Idea2:

→ Apply BS in Every Row

TC: N * log M SC: O(1)

→ Apply BS in Every Column

TC: M * log N SC: O(1)

Idea4:

ceil ( )

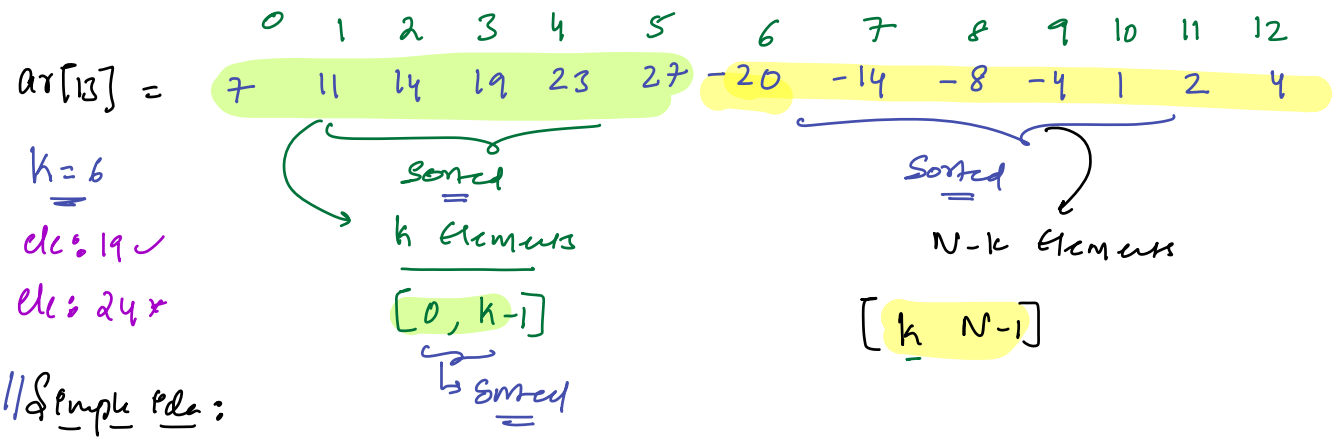
→ Apply ceil in last Column : log N

→ Apply BS in that row : log M

↳ search ( )

→ TC: O(log N + log M)

58] Given an array, which is formed by rotating a Distinct sorted array by k times (In clockwise), Search for a given Element




// Simple Idea:

→ Iterate & check

TC: O(N) SC: O(1)

Idea2:

(Worst Case k=N/2)

1) Apply BS in green part

l = 0, h = k-1 → : ($\log_2^N$)

2) Apply BS in yellow part

l = k, h = N-1 → : ($\log_2^N$)

TC: $O(\log_2^N + \log_2^N) \rightarrow O(\log_2^N)$

SC: O(1)

Twist:

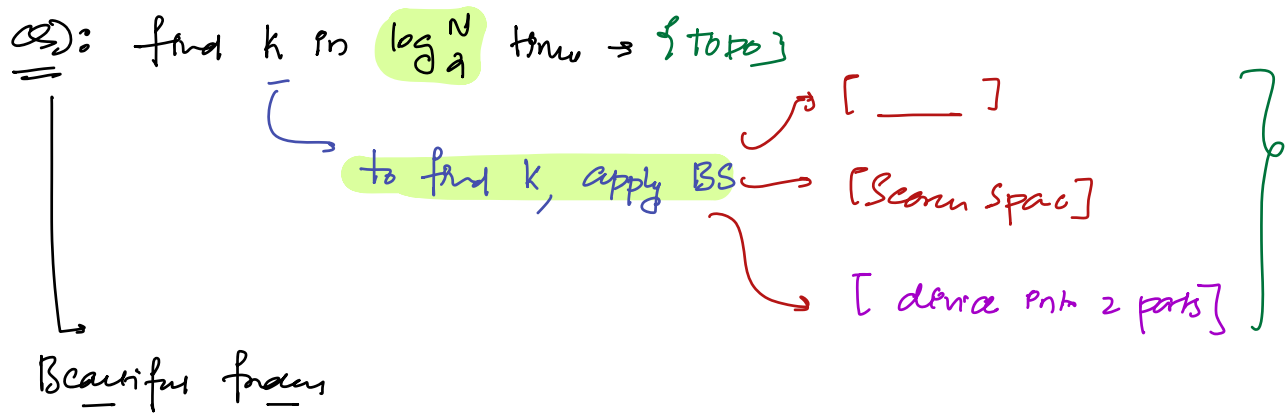
→ k is not given.

Q): given ar[] formed by rotating a distinct distinct sorted, (some time), In given array find x

Input: Just find ek — Expected TC: $\log_2^N$

Sol: Iterate & k value

TC: O(N)


find k in $\log_a N$ time → {TODO}
to find k, apply BS
[ ___ ]
[Search space]
[ divide into 2 parts ]
Beautiful Trees

5(8)