Today's content

- Searching Basics
- Why mid at half?

Problems.

- Search in sorted array
- finding floor in sorted array
- finding first occurence in sorted array
- search in 2D sorted matrix.
- search in rotated sorted array.

friend is missing → Police → (target) What to search
→ (search space) Where to search

for example →

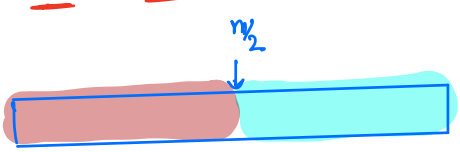
word → { Dictionary / Book / Newspaper }

phone number → { contacts / phone book }

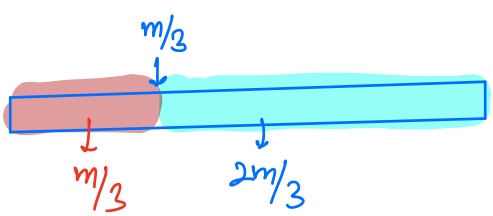
observation → If search space is ordered, searching becomes easy.

Dog.

why mid should be at half?



→ discarding 50%.



→ worst → discarding $\frac{m}{3}$.

{ idea → Every time if your search space is reduced to half. → Binary Search }

Binary Search. ⇒ Divide search space into 2 parts, neglect half of the search space.

target

Search space.

Q: If after diving the search space, if we can't neglect half of the space, can we apply binary search? → {NO}

red flag.

Q:] Given a sorted array with distinct elements. We need to search for index of an element K.

Note:- if K is not present in the array, return -1.

K=12

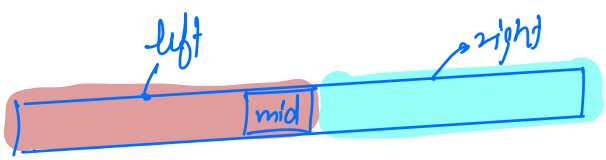
arr[] = { 3 6 9 12 14 19 20 23 25 27 }
0 1 2 3 4 5 6 7 8 9

Idea 1:- Iterate on the array & search for element K.

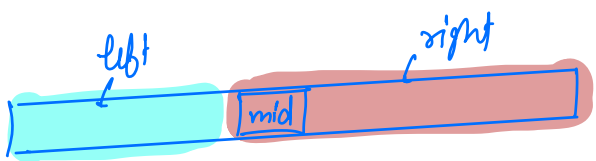
{ T.C → O(N)
S.C → O(1) }

Idea 2:- target : K
search space : entire array.

Case 1: arr[mid] == K { return mid }

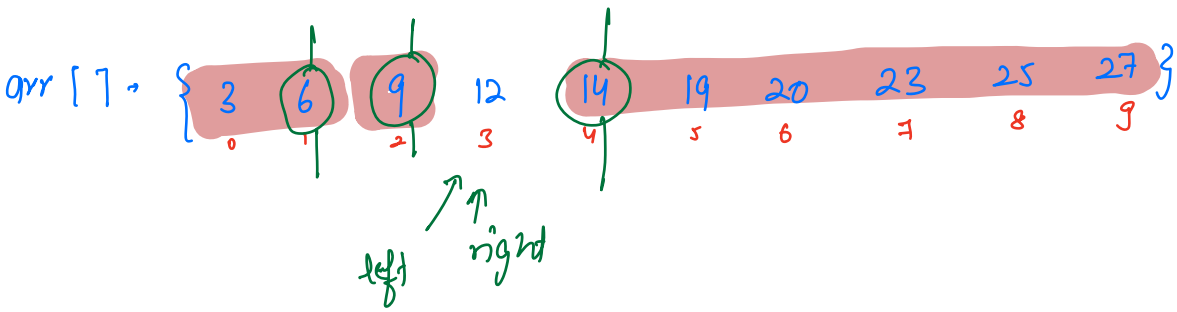


Case 2: arr[mid] < K
→ search K in right



Case 3: arr[mid] > K
→ search K in left.

K = 12.



left	right		
0	9	mid = $\frac{0+9}{2}$ = 4.	arr[mid] > K. go to left. (right = mid - 1)
0	3	mid = $\frac{0+3}{2}$ = 1	arr[mid] < K. go to right. (left = mid + 1)
2	3	mid = $\frac{2+3}{2}$ = 2	arr[mid] < K. go to right (left = mid + 1)
3	3	mid = $\frac{3+3}{2}$ = 3.	arr[mid] == K. { return mid. }

pseudo-code

```
int search ( arr , N , K ) {
    left = 0 , right = N-1
    while ( left <= right ) {
        mid = (left + right) / 2 ;
        if (arr[mid] == K) { return mid }
        else if (arr[mid] < K) { left = mid + 1 }
        else { right = mid - 1 }
    }
    return -1
}
```

N
↓
N/2
↓
N/4
↓
N/8
|
|
1

T.C → $O(\log_2 N)$

S.C → $O(1)$

Q) Given a sorted array, find **floor** of a given number K.

⇓

{greatest number ≤ K. in arr[ ]}

arr[9]: [-5 (0), 2 (1), 3 (2), 6 (3), 9 (4), 10 (5), 11 (6), 14 (7), 18 (8)] , K = 12.

K=5 : 3

K=4 : 3

K=10 : 10

K=20 : 18

K=-7 : -5 ✗

↳ INT_MIN

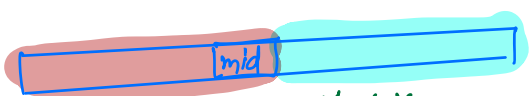
idea-1 → iterate and update ans:

ans = INT_MIN = 9
= -5 = 10
= 2 = 11
= 3 {return ans}
= 6

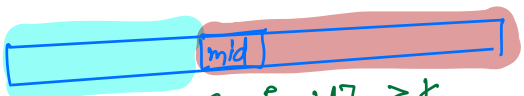
{T.C → O(N), S.C → O(1)}

target : max <= K search space : entire array

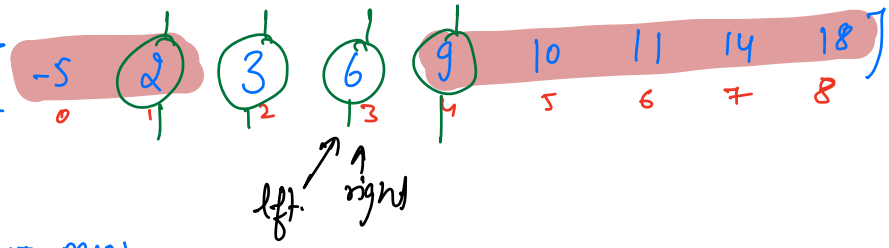
Case-1 : arr[mid] == K {return mid}



Case-2 : arr[mid] < K ans = arr[mid] go to right



Case-3 : arr[mid] > K go to left.

arr[9]: 

K=5.

ans = INT_MIN.

lft.	right.	mid.		
0	8	$\frac{0+8}{2} = 4$	arr[mid] > K	⇒ go to left right = mid-1
0	3	$\frac{0+3}{2} = 1$	arr[mid] < K	ans = 2. ⇒ go to right lft = mid+1.
2	3	$\frac{2+3}{2} = 2$	arr[mid] < K	ans = 3 ⇒ go to right left = mid+1
3	3	$\frac{3+3}{2} = 3$	arr[mid] > K	⇒ go to left right = mid-1
3	2	{ break }		

pseudo-code.

```
int search ( arr , N , K ) {
    left = 0 , right = N-1 , ans = INT. MIN
    while ( left <= right ) {
        mid = (left + right) / 2 ;
        if (arr[mid] == K) { return K }
        else if (arr[mid] < K) { ans = arr[mid] , left = mid + 1 }
        else { right = mid - 1 }
    }
    return ans
}
```

{ T.C → $O(\log_2 N)$
S.C → $O(1)$ }

Q) Given a sorted array of N elements. Find the first occurence (index) of given element K.

arr[ ] : {	-5	-5	0	1	1	5	5	5	5	5	5	8	10	10	15	15 }
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

K=5 : 5

K=0 : 2

K=10 : 12

K=2 : -1

idea-1.

iterate & get first occurence.

{ T.C → O(N) S.C → O(1) }

idea-2. target : 1st occurence of K.

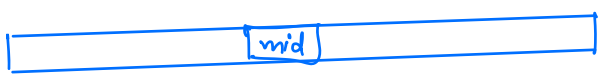
search space : Entire array.

Case 1.: arr[mid] == K

→ ans = mid
→ go to left
→ right = mid - 1

Case-2.: arr[mid] < K

→ go to right
→ left = mid + 1



Case-3.: arr[mid] > K

→ go to left.
→ right = mid - 1

Tracing.

arr[ ]: { -5 -5 0 1 1 5 5 5 5 5 8 10 10 15 15 }

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

left right

{K=5}

ans = -1

left	right	mid		
0	15	7	arr[mid] == K	ans = 7 right = mid-1
0	6	3	arr[mid] < K	go to right left = mid+1
4	6	5	arr[mid] == K	ans = 5, go to left. right = mid-1
4	4	4	arr[mid] < K	go to right left = mid+1
5	4		(left > right) break.	

→ {# todo → code}

→ {# todo → find last occurence of K in a sorted array}

Q) Given a sorted matrix. Search for K.

(N*m)

	0	1	2	3	4
0	2	3	9	10	14
1	16	19	20	21	24
2	27	30	33	39	42
3	43	46	50	52	59

K=10 (14) , K=21 (24) , K=33 (42)

→ {smallest element >= K}

↓

ceil of K.

Idea-1. → Search entire matrix

T.C → O(N*m) , S.C → O(1)

idea-2. Apply B.S on every row

T.C → O(N log m) , S.C → O(1)

Apply B.S on every column

T.C → O(m log N) , S.C → O(1)

Idea-3.

Element can only be present in a single row.

iterate on the last column & check which row should be selected for applying B.S.

T.C → O(N+log m) , S.C → 1.

idea-4.

⇒ Apply B.S in last column to find ceil of K : log N

⇒ Apply B.S in that particular row. : log M

{# todo → code}

{T.C → O(log m + log N)
S.C → O(1)}

Q) Given an array, which is formed by rotating a sorted array K times in clockwise direction.

→ Array contains distinct elements.

→ search for a given element (ele)

K=6, ele=24

arr[13]: [7 11 14 19 23 27 -20 -14 -8 -4 1 2 4]

0 1 2 3 4 5 6 7 8 9 10 11 12

B.S B.S

idea-1. iterate & search for element.

T.C → O(N), S.C → O(1)

idea-2. Apply B.S in yellow region. → log N

Apply B.S in green region. → log N

{ T.C → O(log N) , S.C → O(1) }

---

twist. → K is not given?

Q) Given a sorted & rotated array containing distinct elements.

Search x.

{ Hint → first find K } Expected T.C → $O(\log_2 N)$

Divide integer

$N = 35$, $K = 6$

$$\frac{35}{6} = \frac{24+11}{6} = \frac{24}{6} + \frac{11}{6}$$

$$= 4 + \frac{11}{6}$$

$$\frac{11}{6} = \frac{6+5}{6} = \frac{6}{6} + \frac{5}{6}$$

$$= 1 + \frac{5}{6}$$

In binary, we are dealing with powers of 2.

$a << i \Rightarrow a * 2^i$

$N = 35$, $K = 6$.

$q = \underset{3}{0}\ \underset{2}{1}\ \underset{1}{0}\ \underset{0}{1}$

$\{q = 5\}$

$K << 0 \Rightarrow K * 2^0 = 6 * 1 \leq 35$

$K << 1 \Rightarrow K * 2^1 = 6 * 2 \leq 35$

$K << 2 \Rightarrow K * 2^2 = 6 * 4 \leq 35$

$K << 3 \Rightarrow K * 2^3 = 6 * 8 \leq 35$

$N = 35 - (K << 2)$, $K = 6$.

$= 35 - 24 = 11$.

$K << 0 \Rightarrow K * 2^0 = 6 * 1 \leq 11$

$K << 1 \Rightarrow K * 2^1 = 6 * 2 \leq 11$

$N = 11 - (K << 0)$

$= 11 - 6 = 5$, $K = 6$.