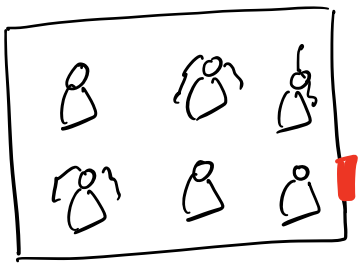
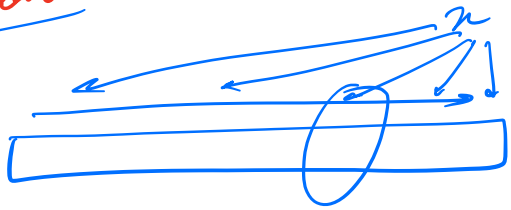
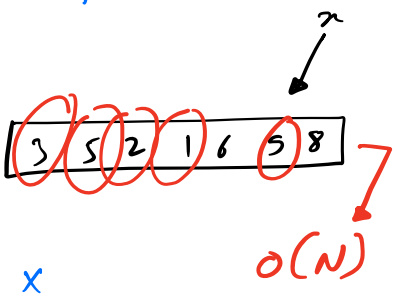


# of favs to look at to find a friend = N

Linear Search

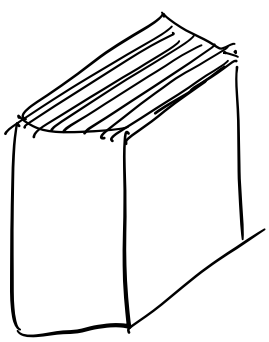




O(N)

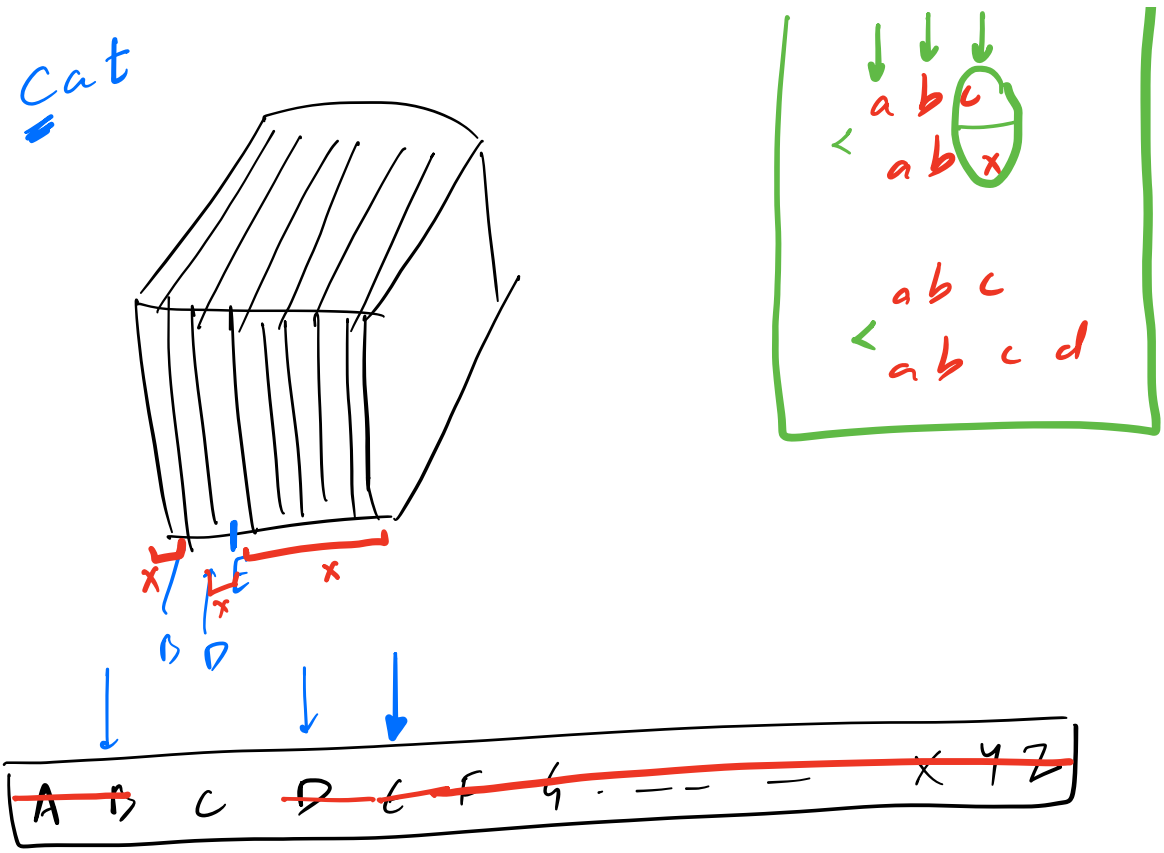
```
for( i = 0 ⟶ N-1) {
    if( A[i] == x) out (i);
}
out(-1);
```

DICTIONARY



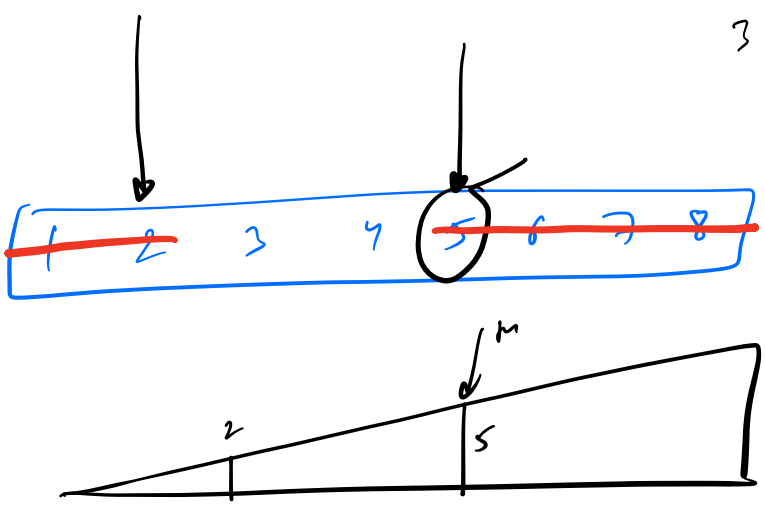
Lexicographical ordering

Alphabetical order!

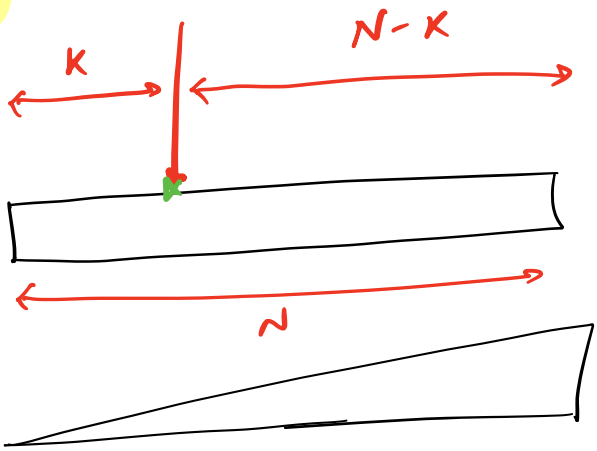


→ Reducing Search Space!

→ Searching in a DICT is easier because of ORDERING.



Why MID ?



$K < N-K$

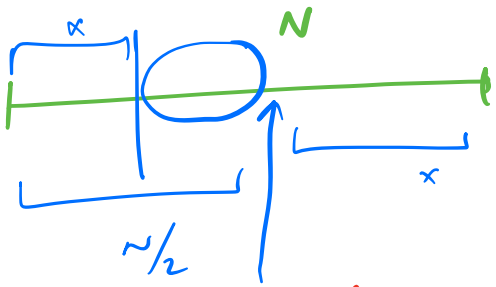
$K = N-K$

$2K = N$

$K = N/2$

Binary Search

Try to divide the search space into two halves & discard 1 half by using some condition.

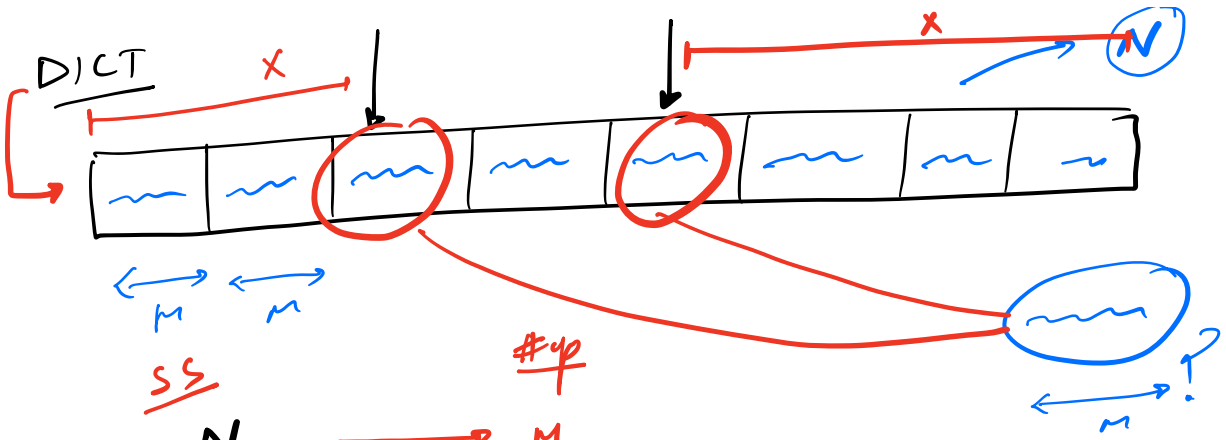


N → N/2 → N/4 → N/8 → ... → 1

# of iterations where SS = N

$= \log_2 N$

$\log_2$ (Search Space Size)



SS → #op

N → M

N/2 → M

N/4 → M

⋮ ⋮

1

#of it of BS = $f(N)$

#of ops/it of BS = $O(M)$

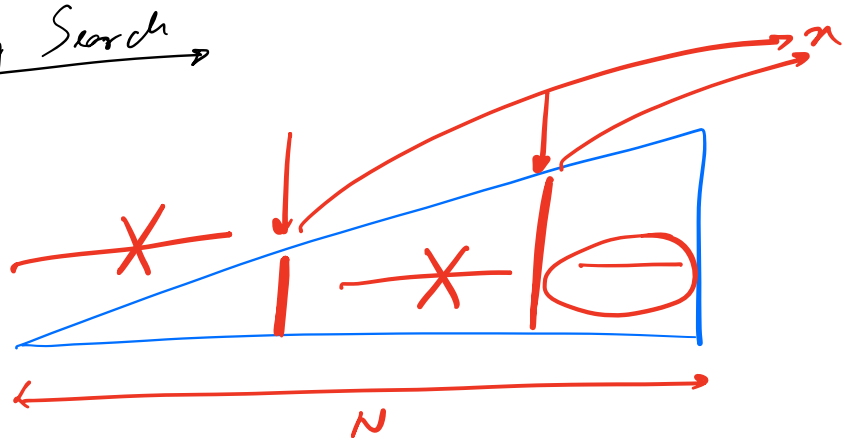
TC = $O(M \cdot f(N))$

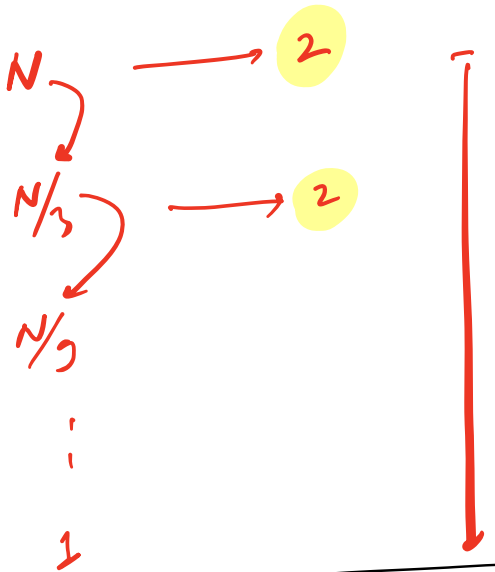
Search Space Size

TC of BS

↳ $O(\log(S.S) \times$ time per it$)$

Ternary Search →





it → $\log_3 N$

Q Given a sorted array with DISTINCT element
Search for the index of the element K.
If K is not present, then ret -1.

A: [3, 6, 9, 12, 14, 19, 20, 23, 25, 27]
(0 1 2 3 4 5 6 7 8 9)

K = 12 → 3

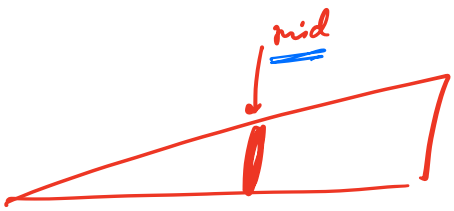
K = 13 → -1

1) Linear Search

TC = O(N)

SC = O(1)

2) B.S. → l = 0, h = N-1



CASES

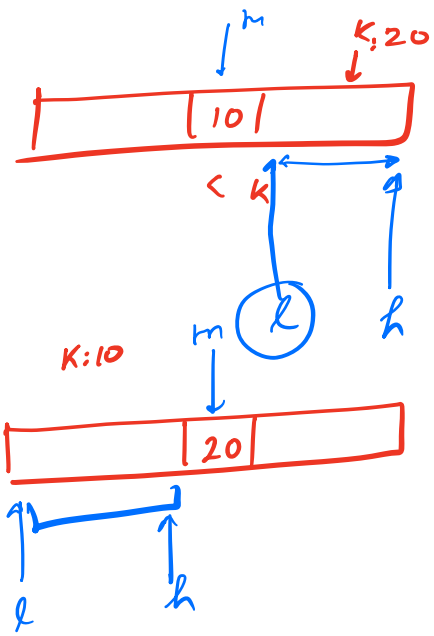
I) A[mid] == K — set mid

II) A[mid] < K — go right

l = m+1;

III) A[mid] > K — go left

h = m-1



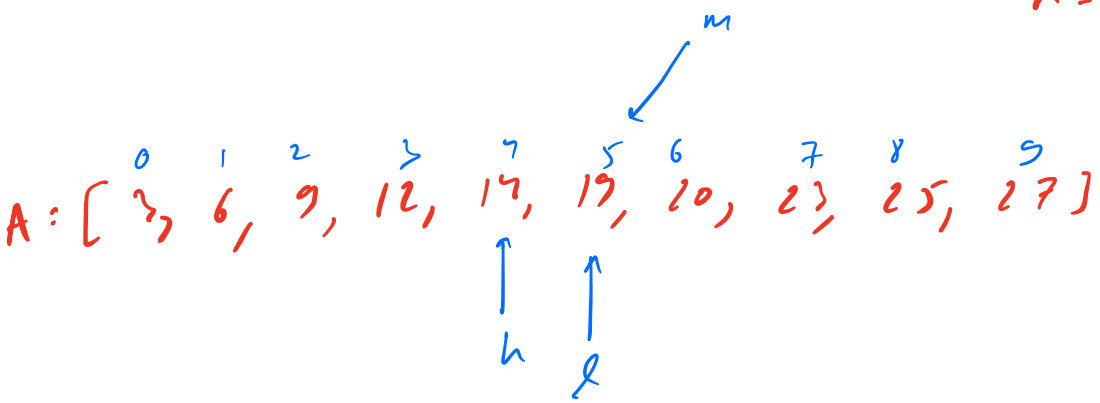
0	1	2	3	4	5	6	7	8	9
3	6	9	12	14	19	20	23	25	27

$(l+h)/2$

K = 12

l	h	mid	A[mid]
0	9	4	14 > K ; go left
0	3	1	6 < K ; go right
2	3	2	9 < K ; go right
3	3	3	12 == K ; get mid

K = 15



```
// A[], N, K
l = 0, h = N-1;
while ( l <= h ) {
    m = (l+h)/2
    if ( A[m] == K) ret m;
    else if ( A[m] < K) l = m+1;
    else h = m-1;
}
ret -1;
```

SS → N
#it → $\lg(N)$

→ O(1)

TC = $O(\lg N)$

SC = $O(1)$

DIFF STYLES of BS

```
l = 0, h = N-1
while ( l <= h) {
    m = -
    ret -
}
ret
```

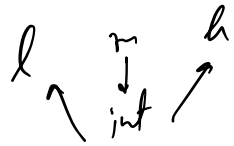
```
l = -1, h = N
while ( h - l > 1) {
    m - -
    if l = m
    - h = m
}
ret l;
```

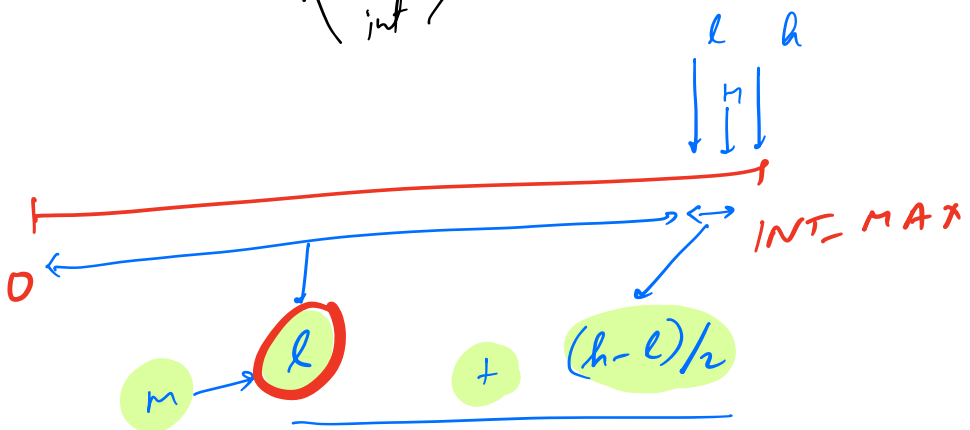
```
l = 0, h = N-1
while ( l < h) {
    m - -
    —
    —
}
ret (l+1);
```

④

$m = (l+h)/2$;

$m = l + (h-l)/2$;





Q Given a sorted arr. DISTINCT.

Find the floor of a no. K in the arr!

↳ greatest element $\leq K$

A: [-5, 2, 3, 6, 9, 10, 11, 14, 18]

K = 5 → 3

K = 4 → 3

K = 6 → 6

K = -7 → not INT_MIN !

CASES

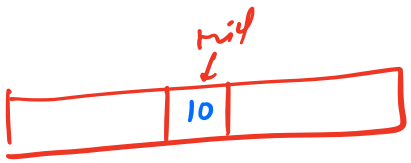
I) A[mid] == K ret k

K = 20

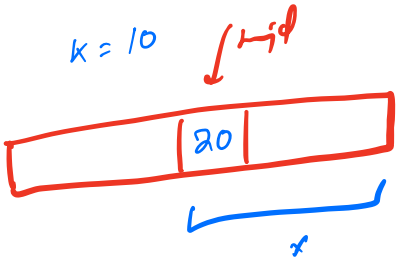
II) A[mid] < K

ans = A[mid] go right

l = m+1



K = 10



III A[mid] > K go left

h = m-1

0	1	2	3	4	5	6	7	8
-5	2	3	6	9	10	11	14	18

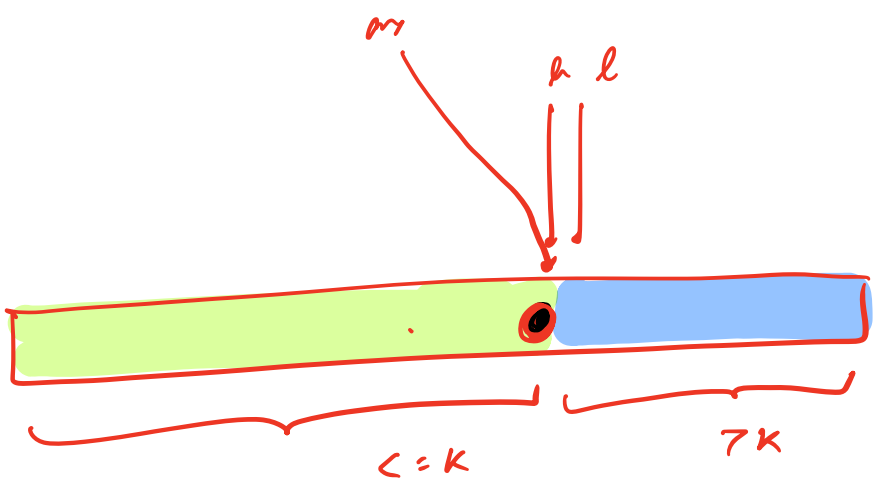
A: [-5, 2, 3, 6, 9, 10, 11, 14, 18] K = 5

l	h	m	A[m]	ans
0	8	4	9 > K	-∞
0	3	1	2 < K	2
2	3	2	3 < K	3
3	3	3	6 > K	3
3	2	STOP!		

```
l = 0,   h = N-1,   ans = -∞
while( l <= h) {
    m = (l+h)/2;
    if( A[m] == K) ret K;
    else if( A[m] < K) {
            ans = A[m], l = m+1;
    }
    else   h = m-1;
}
ret ans;
```

TC = O(lg N)

SC = O(1)




```
if( A[m] <= K) ans = A[m], l = m+1,
else   h = m-1
```

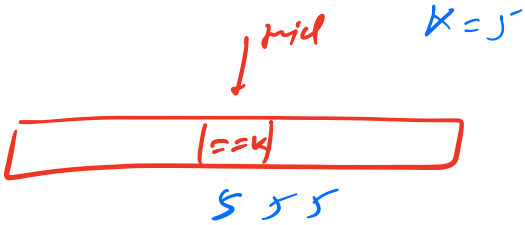
Q Given a sorted Array of K.
find the first occurrence of K. Return the index!
if it does not exist, return -1;

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A:	-5,	-5,	-3,	0,	0,	1,	1,	5,	5,	5,	5,	5,	8,	11,	11

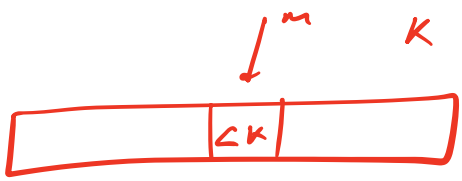
K = 5 ⟶ 7
K = 6 ⟶ -1

CASES

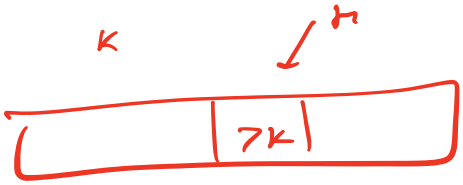
I) A[mid] == K
ans = mid
go left, h = m - 1

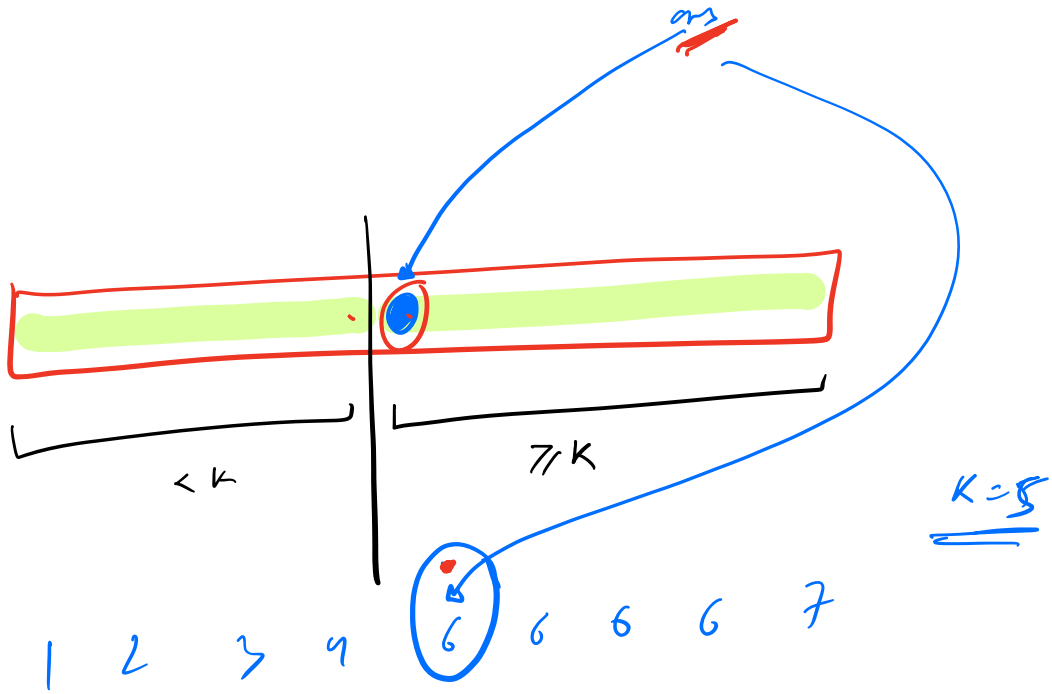


II) A[mid] < K
go right, l = m + 1



III A[mid] > K
go left; h = m - 1





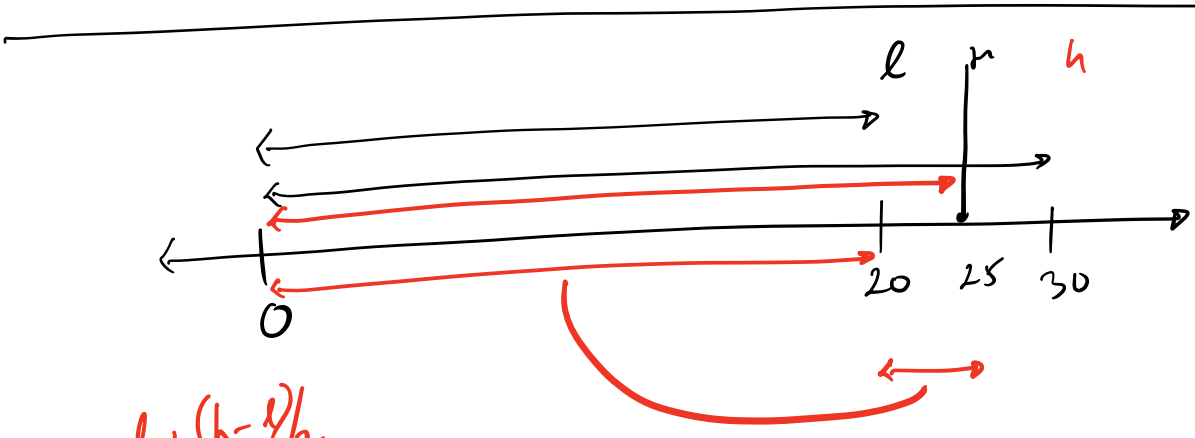
A[m] >= K ans = m, h = m-1

A[m] < K l = m+1

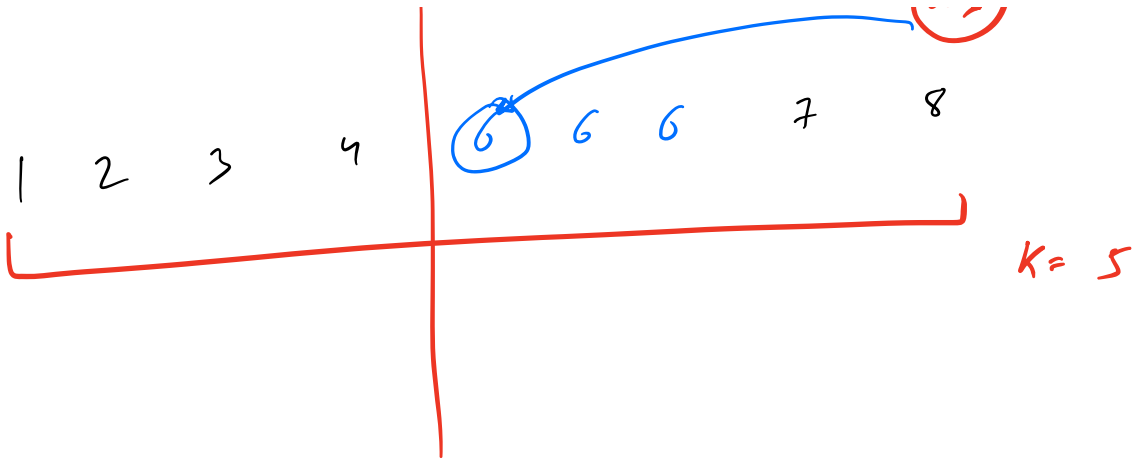
---

if (A[m] == K) → ret m;

ret -1;



l + (h - l)/2



< k

7 = K