Todays Content

1. Searching Intro
2. Search in sorted array
3. search floor of ele in sorted array
4. Search 1st occurence of ele in sorted array

Searching Basics

Bro/Sis missing → Police station → Photo : Whom to search: Target

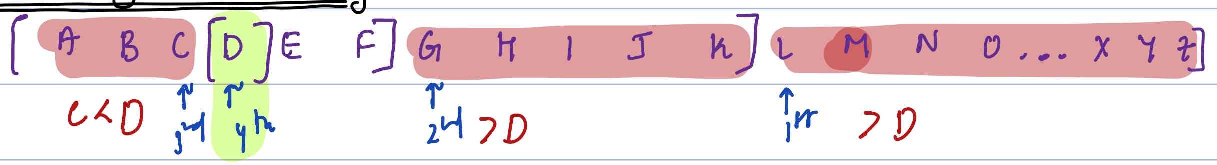
→ location : where to search: Search space/ Answer space

Example:

Target	⇄ SearchSpace
Word	{Dict / Books / News Paper}
PhoneNo	{Contact / Phonebook}

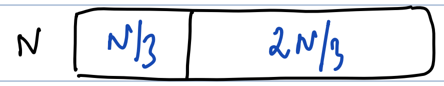
Obs: If SearchSpace is Ordered, Searching becomes easier.

Search Dog in Dictionary



How to where to land? mid?

: Always Discard = N/2 ; always N/2 better N/3 {worst}.

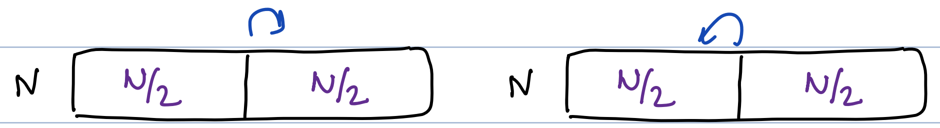


: Discard left : N/3 land at mid.

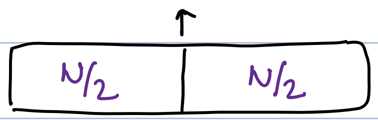
Discard right: 2N/3

When to apply BS:

After dividing search space into 2 parts, if we can discard 1 half of searchspace using some conditions, than we can apply binary search



Note:



Q) Given a sorted ar[N] search if k is present or not?

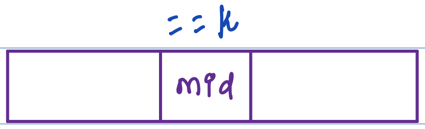
ar[10] =

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

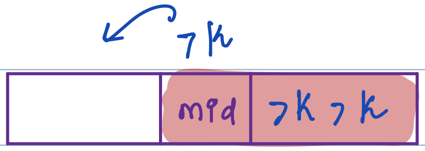
k = 12 : 44

Idea1: Iterate on arr[] & search for k: (True/False) TC: O(N) SC: O(1)

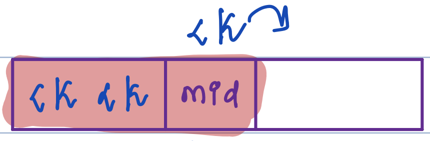
Idea2: Target = k SearchSpace = In arr[]



if(ar[m] == k) { return True; }



if (ar[m] > k) { goto left }



if (ar[m] < k) { goto right }

Tracking Search Space:

We can use s & e to indicate searchspace

Ex1: ar[10] = k = 20

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

l	h	m : (l+h)/2		
0 <=	9	m = (0+9)/2 = 4	ar[m] < k :	goto right l = m+1;
5 <=	9	m = (5+9)/2 = 7	ar[m] > k :	goto left h = m-1;
5 <=	6	m = (5+6)/2 = 5	ar[m] < k :	goto right l = m+1
6 <=	6	m = (6+6)/2 = 6	ar[m] == k :	return true;

Ex1: ar[10] =

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

$l <= h$	$m : (l+h)/2$	$k = 15$
$0 <= 9$	$m = (0+9)/2 = 4$	$ar[m] < k$: goto right $l = m+1;$
$5 <= 9$	$m = (5+9)/2 = 7$	$ar[m] > k$: goto left $h = m-1;$
$5 <= 6$	$m = (5+6)/2 = 5$	$ar[m] > k$: goto left $h = m-1$
$5 <= 4$	# $l > h$ stop process.	

```
bool search(vector<int> &arr, int k){ TC: O(log_2^N) SC: O(1)
    int l=0, h=arr.size()-1;
    while( l<=h ){
        int m=(l+h)/2;
        if( ar[m]==k){ return true;}
        else if( ar[m]>k){
            h=m-1; #goto left
        }
        else{ # ar[m]<k goto right
            l=m+1;
        }
    }
} return false;
```

goto right : $l = m+1$
goto left : $h = m-1$

TC: Initially Search Space = $N \xrightarrow{1^{st}} N/2 \xrightarrow{2^{nd}} N/4 \xrightarrow{3^{rd}} N/8 \longrightarrow \dots\dots 1$:

$\log_2^N$ iterations

29 Given a sorted arr[], find flow of a given num k

Floor: greatest ele <= k

	0	1	2	3	4	5	6	7	8
arr[9] = {	-5	2	3	6	9	10	11	14	18}
	↑	↑	↑		↑				

k	ele
5 :	3
4 :	3
10 :	10
24 :	18
-7 :	-∞;

If flow doesn't exist return -∞ // INT_MIN;

Idea: Iterate in arr[] & update ans;

k=8 ans = -∞ TC: O(N) SC: O(1)

i	ar[i]	ans=
0	-5<=8	ans=-5; update & look for better
1	2<=8	ans= 2; update & look for better
2	3<=8	ans= 3; update & look for better
3	6<=8	ans= 6; update & look for better
4	9<=8	9 not possible, Right ele not possible. *

return ans = 6

Idea2:

Target = greatest ele <= k Search Space = In arr[] ans = -∞;

==k

	mid	

if ar[mid]==k : return k;

<k

<k <k	mid	

if ar[mid] < k:

  ans = ar[mid];

  goto right

	mid	>k >k

if ar[mid] > k:

  goto left

	0	1	2	3	4	5	6	7
Ex: ar[] = {	2	4	8	12	15	22	24	26 }
					h M	l		

k=20

l	h	m	
0	7	3	ar[m]<k: ans = ar[m] goto right
4	7	5	ar[m]>k : goto left
4	4	4	ar[m]<k: ans = ar[m] goto right
5	4		# stop & return ans = 15.

```
int floorIndex(vector<int> &arr, int k){   TC: O(log_2^N)   SC: O(1)

    int l=0, h=arr.size()-1, ans = -∞;
    while( l<=h){
        int m = (l+h)/2;
        if( ar[m]==k){
            return k;
        }
        else if( ar[m] < k){
            ans = ar[m];
            l=m+1;   #   goto right;
        }
        else{ #ar[m]>k
            h=m-1;   # goto left
        }
    }
    return ans;
}
```

TODO: Ceil(n) : Smaller ele >= k.

3Q) Given an sorted arr[], find the first occurence index of given element

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ar[ ] = {	-5	-5	-3	0	0	1	1	5	5	5	5	5	5	8	10	10	15 }
								P_1		12 - 7 + 1 = 6			P_2				

occ = [P_1 P_2] = $P_2 - P_1 + 1$

k:

5 : 7 Idea1: Iterate in arr[] & get 1st occurence of k.

-5 : 0 TC: O(N) SC: O(1)

20 : -1; If element doesn't exist

Idea2:

Target = 1st ocurence index of k Search Space = In arr[] ans = -1;

>k

	mid	>k > k

if ar[mid] > k : goto left

<k

<k <k	mid	

if ar[mid] < k : goto right

m

kk	k	k k k

if ar[mid] == k:

ans = m; #update ans

goto left; #better ans

Example:

	m 0	✓ 1	2	✓ 3	4	5	6	7
Ex: ar[ ] = {	2	5	5	5	5	5	8	10 }
k = 5	h	l						

l <= h	m	
0 <= 7	3	ar[m] == k : ans = 3; goto left h = m-1;
0 <= 2	1	ar[m] == k : ans = 1; goto left h = m-1;
0 <= 0	0	ar[m] < k : goto right l = m+1;
1 <= 0	✗	Stop & return ans = 1;

int firstOcurence(vector<int> &arr, int k){ TODO

TODO: 1st ocurence of k in sorted arr[] ✓ TC: $O(\log_2 N)$

TODO: last ocurence of k in sorted arr[] ✓ TC: $O(\log_2 N)$

TODO: Frequency of k in sorted arr[] ✓ { P_2[last_occ index] - P_1[1st_occ index] + 1 }
TC: $O(\log_2 N + \log_2 N) = O(\log_2 N)$