Todays Content

1. Searching Intro
2. Search in sorted array
3. Search first occurence of ele in sorted array
4. Search last occurence of ele in sorted array

Searching Basics

Bro/Sis missing → Police station
- Details : whom to search : Target
- location : where to search : Searchspace

Example:

Target ⟺ Searchspace

Word { Dict / Books / News Paper }

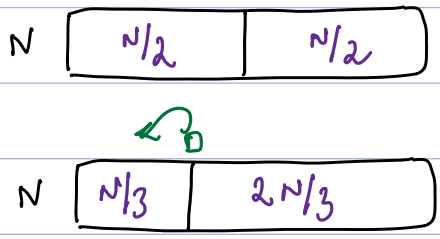
PhoneNo { Contact / Phonebook }

Obs: If Searchspace is Ordered, Searching becomes easier

Search Dog in Dictionary

Binary Search: Divide search space into 2 parts & we search into 1 part of search space.

How to where to land?



: Always Discard = N/2 Search space

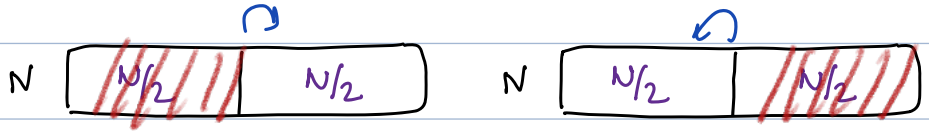
: Discard left: N/3

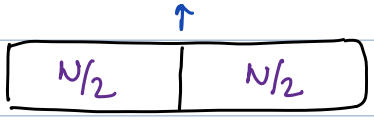
Discard right: 2N/3

Note: If we compare worst above will always discard N/2.

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: 

: If we cannot discard 1 side of searchspace, we cannot apply BS

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

	0	1	2	3	4	5	6	7	8	9
ar[10] = {	3	6	9	12	14	19	20	23	25	27 }

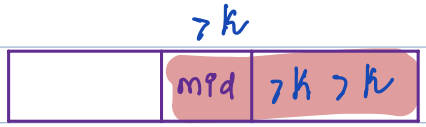
k = 12 Return True;

Idea1: Iterate on arr[] & search if k exists or not

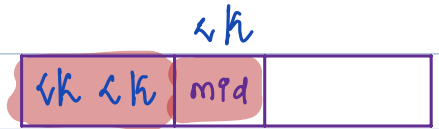
TC: O(N) SC: O(1)

Idea2: Target = k SearchSpace = In arr[]

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



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

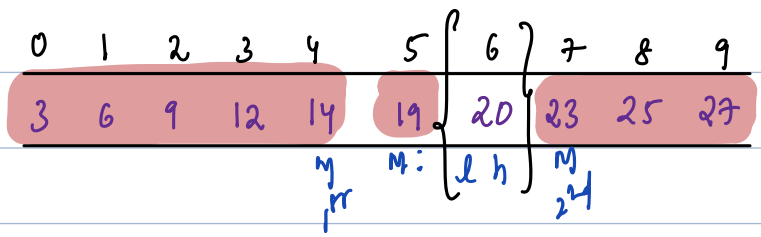


if(ar[m] < k) { goto right }

Tracking Search Space:

We can use s q e r l & h to indicate searchspace

Eg1: ar[10] =



l <= h m : (l+h)/2

0 <= 9 m = (0+9)/2 = 4 ar[m] < k : #discard left goto right l=m+1;

5 <= 9 m = (5+9)/2 = 7 ar[m] > k : #discard right goto left h=m-1;

5 <= 6 m = (5+6)/2 = 5 ar[m] < k : #discard left goto right l=m+1;

6 <= 6 m = (6+6)/2 = 6 ar[m] == k : return True;

Eg1: ar[10] =

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

m:1 (h) m:3 (l) m:2

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; h = 5-1 = 4

5 <= 4 l > h; stop process search space is completed. #return false.

```
bool search(vector<int> &ar, int k){   TC: O(log2 N)   SC: O(1)
    int l = 0, h = N-1;
    while(l <= h){
        int m = (l+h)/2;
        if(ar[m] == k){ return true; }
        else if(ar[m] > k){ #go to left
            h = m-1;
        }
        else{ # ar[m] < k go to right
            l = m+1;
        }
    }
    return false;
}
```

goto right l = m+1;
goto left h = m-1;

TC: Initial search space: $N \xrightarrow{1} N/2 \xrightarrow{2} N/4 \xrightarrow{3} N/8 \rightarrow \dots \xrightarrow{\log_2^N} 1$

Note: In BS, there are $\log_2^N$ iterating # N = Size of search space

29 Given a sorted ar[], find flow of a given ele k.

Floor: greatest ele <= k in ar[]

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

k ele

5 : 3

4 : 3

10 : 10

24 : 18

-7 : -∞

Idea: Iterate & Calculate flow & retun ans.

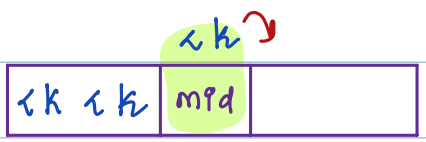
k = 8 ans = -∞ TC: O(N) SC: O(1)

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	Stop & retun ans = 6.

Idea2:

Target = greatest ele <= k SearchSpace = In ar[] ans = -∞;

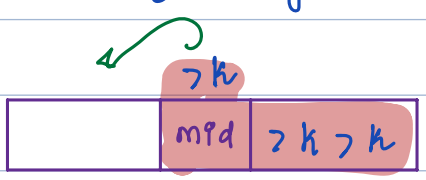
if ar[mid] == k : retun ar[mid]



if ar[mid] < k:

ans = ar[mid]

goto right l = m+1



if ar[mid] > k:

goto left h = m-1;

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

k = 20

l	h	m	ar[m] <= k	ans = -1;
0	7	3	ar[3] <= 20	ans = 12, l = m+1;
4	7	5	ar[5] <= 20	h = m-1;
4	4	4	ar[4] <= 20	ans = 15; l = m+1;
5 > 4			# Stop proces	retun ans = 15.

```
int floorValue(vector<int> &ar, int k){ TC: O(log N/2)  SC: O(1)

    int l=0, h=N-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;
        }
        else{ #ar[m]>k
            h=m-1;
        }
    }

    return ans;
}
```

TODO: Ceil(n): Smallest element >= n

If Ceil does not exist return +∞

38 Given an sorted arr[], find the first occurence index of given ele

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
arr[ ] = {	-5	-5	-3	0	0	1	1	5	5	5	5	5	5	8	10	10	15 }
k:								f					l				

c: {f..l} = l-f+1 = 6.

5 : 7

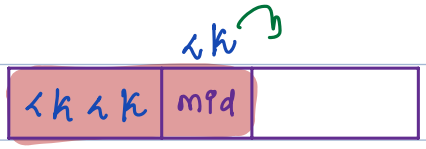
-5 : 0

20 : -1

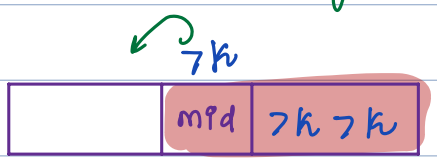
Ideal: Iterate & search for k & return 1st occurence index.

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

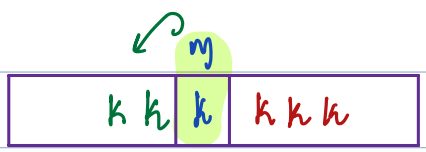
Target = 1st occurence index: k SearchSpace = In arr[] ans = -1;



if arr[mid] < k : goto right



if arr[mid] : goto left



if arr[mid] == k:

ans = m;

goto left;

Example

	0	1	2	3	4	5	6	7
Ex: arr[]: {	2	5	5	5	5	5	8	10 }
	m							
	h							
	l							

k = 5

ans = -1;

l	h	m	
0	7	3	arr[m] == k : ans = 3; h = m-1
0	2	1	arr[m] == k : ans = 1; h = m-1;
0	0	0	arr[m] < k : l = m+1
1	0		# stop & return ans = 1.

TODO: 1st occurence code

TODO: last occurence code & idea

TODO: In sorted arr[] find frequency of k:

f = first occurence of k

l = last occurence of k

c = l-f+1;

```
int firstOccurence(vector<int> &arr, int k){
```