⇒ **Searching** is most commonly used operation CS.

↳ Amazon | Myntra | FB | Instagram / Netflix.

Search :-

↳ **Target** :- Something that has to be searched.

Search Space :- Where to search for the target

En

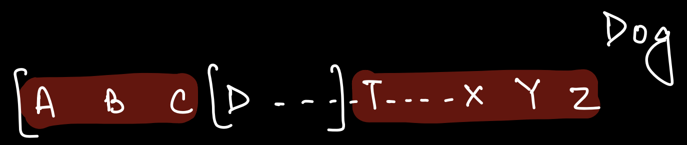
1) Search a **word** in **newspaper**.

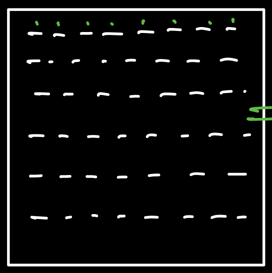
2) Search a **word** in a **dictionary**

3) Search a given **value** in **Array**

4) Search a given **value** in a **sorted Array**.

Quiz Searching in dictionary is faster because dictionary stores the word in lexicographic order.





⇒ N words.

⇒ linear Search

	0	1	2	3	4	5	6	7	8	9
A:	3	6	9	12	14	19	20	23	25	27
								↑ > K		

Sorted Array.

$K = 12$

Best choice will be middle

s	e	randomIndex	
0	9	7	⇒ A[7] > K
0	6	4	⇒ A[4] > K
0	3	2	⇒ A[2] < K
3	3	3	⇒ A[3] == K → 3

		m	s	e						
		↓	⇓	⇓						
	0	1	2	3	4	5	6	7	8	9
A:	3	6	9	12	14	19	20	23	25	27

$K = 9$

s	e	mid	
0	9	$\frac{9}{2} = 4$	⇒ A[4] > 9
0	3	$\frac{3}{2} = 1$	⇒ A[1] < 9
2	3	$\frac{5}{2} = 2$	⇒ A[2] == 9. ↳ 2

				me ↓⇓	s ⇓						
	0	1	2	3	4	5	6	7	8	9	
A:	3	6	9	12	14	19	20	23	25	27	$k = 13$

s	e	mid	
0	9	4	⇒ $A[4] > 13$
0	3	1	⇒ $A[1] < 13$
2	3	2	⇒ $A[2] < 13$
3	3	3	⇒ $A[3] < 13$.
4	3		

$\underbrace{4 \qquad 3}$

$s > e$ ⇒ Break.

⇒ Binary Search.

$$\underbrace{N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{4} \longrightarrow \frac{N}{8} \longrightarrow \text{- - - - - -} \; 1/0}_{\boxed{\log N}}$$

```
                              Sorted
int  binarySearch( A[], N, K){
        //define the search Space
        s = 0, e = N-1
        while ( s<=e ){
              mid = (s+e)/2
              if( A[mid] == K) return mid;
              if ( A[mid] > K){
                    //Go to left
                    e=mid-1;
              }
              else
                    s = mid+1;
              }
        }
        return -1;
}
```

TC: O(logN)

SC: O(1)

Recursive BS

TC: O(logN)

SC: O(logN)

Q. Given a sorted array (ascending), find the floor of a given value k.

floor(k) ⇒ largest no. less than or equal to k.

	0	1	2	3	4	5	6	7	8
A:	-5	2	3	6	9	10	11	15	18

floor(20) = 18

floor(2) = 2

floor(6) = 6

floor(14) = 11

floor(8) = 6

floor(-5) = -5

floor(-10) = X

$-5 > -10$

↓ s e

	0	1	2	3	4	5	6	7	8
A:	-5	2	3	6	9	10	11	15	18

k=4

floor(4)

s	e	mid	ans = -∞
0	8	4	-∞
0	3	1	2
2	3	2	3
3	3	3	3
3	2		

s > e ⇒ Break.

```
int   floor ( A[] , N, K) {
      s = 0, e = N-1;
      while ( s <= e) {
            m = (s+e) /2;
            if (A[mid] == K)
                  return A[mid];
            else if ( A[mid] < K) {
                  ans = A[mid];
                  s = mid+1; // go to right, to find
                                 better ans.
            }
            else {
                  e = mid-1;
            }
      }
      return ans;
}
```

TC: $O(\log N)$

SC: $O(1)$

Q. Given an Array of size N sorted in ascending order. Find the frequency of a given target K.

Amazon
Paytm.

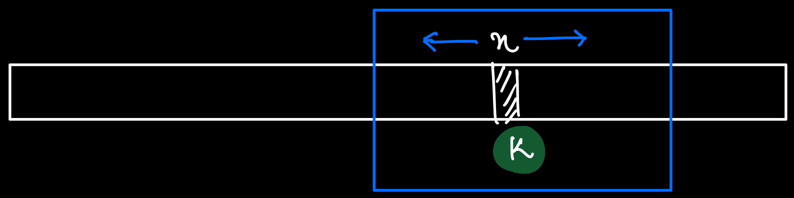
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A:	-5	-5	-3	0	0	1	1	1	5	5	5	5	5	5	9	10
									⇑ i					⇑ j		

freq(0) = 2

freq(5) = 6

freq(-1) = 0

freq(5) = j-i+1



$O(\log N) + O(N) \Rightarrow O(N)$ { Worst Case }

⇒ i ⇒ first occurrence of K in Array A.

j ⇒ last occurrence of K in Array A.

$$\boxed{freq = j - i + 1}$$

First Occurrence index

									m s							e
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A:	-5	-5	-3	0	0	1	1	1	5	5	5	5	5	5	9	10

K = 5

s	e	mid	first_occ
0	15	7	-1
8	15	$\frac{23}{2} = 11$	11 move to left
8	10	9	9 move to left
8	8	8	8 move to left
8	7		

S>e ⇒ Break.

```
int firstOcc( A[], N, K){
        s=0, e=N-1
        while(s<=e){
            mid=(s+e)/2;
            if(A[mid]==K){ // Store mid as an
                                   ans & move to
                                   left
                i=mid;
                e=mid-1;
            }
            else if (A[mid]<K)  s=mid+1;
            else   e=mid-1;
        }
        return i;
}
```

last Occurrence

```
{ if (A[mid] == K){   // Store mid as an ans
          j = mid;      // & move to right
          s = mid + 1;
  }
```

$$freq = j - i + 1$$

$$TC: \underbrace{O(\log N)}_{\text{first occ}} + \underbrace{O(\log N)}_{\text{last occ.}}$$

$$SC: O(1)$$

HW Solve above problem in one log N iteration

What is the condition to apply Binary Search?

Binary Search can be applied when we can come up with a logic to discard one half of the search space in every iteration.

—— * ——