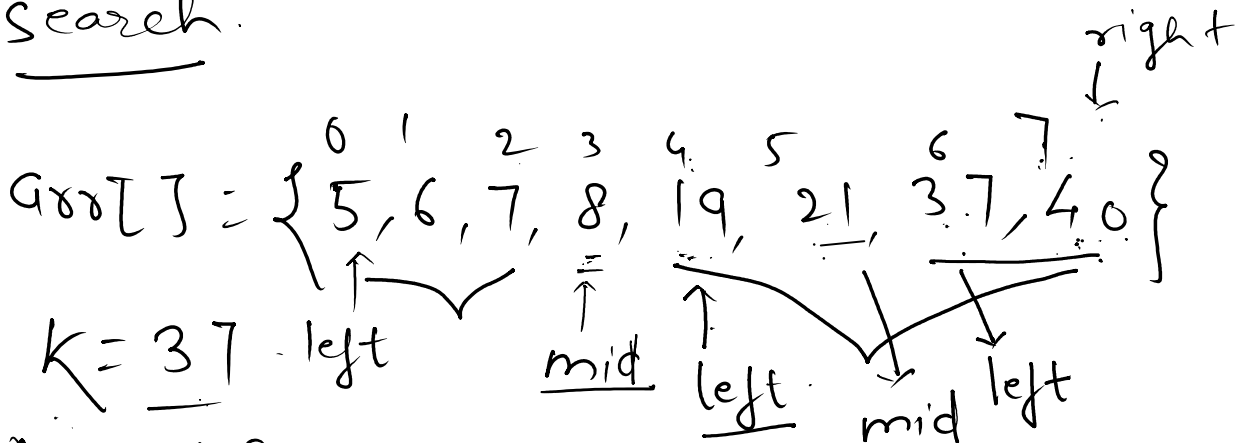
Binary Search → It is used to search an element in a sorted array

Time Complexity → log(n)

```
                0 1 2 3 4 5 6
arr[] = {5, 6, 7, 8, 19, 21, 37}
int target = 21        → using Linear Search
for(int i=0; i<arr.length; i++) {
    if (arr[i] == target) {
        S.O.P(i);
    }
}
```

Time Complexity → O(n)

Efficient approach to search is binary search.



Algorithm.

1. Find the mid of the array by having two pointers one pointer will be at 0

1. Find the index of the mid by using two pointer, one pointer will be at 0 another at n-1

int left=0, right=n-1;

int mid=(left+right)/2

$= (0+7)/2$

$= 7/2$

mid = 3

2. Check if arr[mid] is equal to target, if yes then, mid is the index

Otherwise,

if target is greater than arr[mid]

left will be mid+1.

if target is less than arr[mid]

then right will be mid-1

3. Repeat Step1 & Step2

Dry Run: K=37

5, 6, 7, 8, 19, 21, 37, 40 → (8)

1st Iteration

0 1 2 3 4 5 6 7

↑ left ↑ right

arr[mid] = 8.

2nd Iteration

4 5 6 7

[9, 21, 37, 40] → (4)

mid = (4+7)/2 = 11/2 = 5

left.

3rd Iteration

6 7

37 40 → (2)

mid = (6+7)/2 = 6

arr[mid] = arr[6] = 37

S.O.P(mid);

T.C. → 8 → 8/2/2/2 → $8/2^3$

$\log 8 = \log_2 2^3 = 3\log_2 2 = 3$

→ log(n)

Code.

```
int left=0, right=n-1;
While (left<=right){
int mid=(left+right)/2
if (k== arr[mid]){
 S.O.P ln (mid);
 return;
 }
else if (k> arr[mid]){
    left=mid+1;
}else{
    right=mid-1;
  }
}
```

Search character

0	1	2	3
a	b	c	d	e

Algorithm:-

1. Use binary search to check if character is present or not
   If present then again use modified binary search to find next greater char
   If not present then print -1.

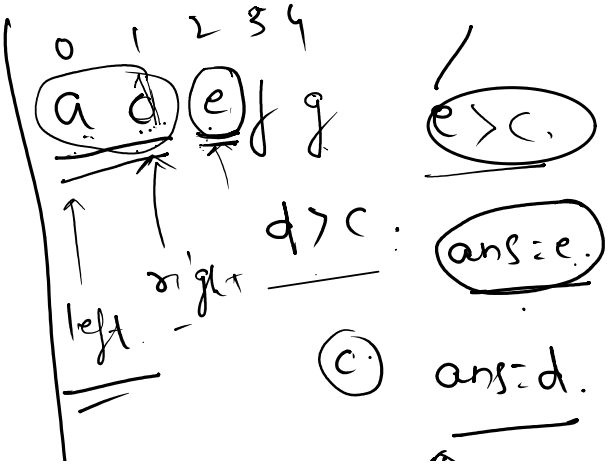
2. To find next greater character, we can use the below code.

↓

c → d

0	1	2	3	4
a	b	c	d	e

```
while(left < right) {
  if (arr[mid] <= k)
```



```
if (arr[mid] < target) {
    left = mid+1;
}else {
    right = mid-1;
    ans = arr[mid];
}
```

Here, we are checking

1. If arr[mid] is smaller or equal than target then we have to shift to right by moving left = mid+1

2. If arr[mid] is greater than target, possibility is left array can have value greater than target or it can not have any greater value.

Therefore, we will store current greater mid value, so that we won't loose it if there are no greater value in left.

```
        Scanner sc = new Scanner(System.in);
        char ch = sc.next().charAt(0);
        int n = sc.nextInt();
        char arr[] = new char[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.next().charAt(0);
        }
        int left =0, right = n-1;
        int res=-1;
        while(left<=right){
            int mid = (left+right)/2;
            if(arr[mid]==ch){
                res = mid;
                break;
            }else if(arr[mid]<ch){
                left=mid+1;
            }else{
                right = mid-1;
            }
        }
        if(res==-1){
             System.out.print(-1);

        }else{
            int left2=0, right2=n-1;
            char ans='/' ;
           while(left2<=right2){
               int mid = (left2+right2)/2;
               if(arr[mid]<=ch){
                   left2 = mid+1;
               }else{
                   ans = arr[mid];
                   right2 = mid-1;
               }
           }
            if(ans=='/'){
```

```
        }else{
            int left2=0, right2=n-1;
            char ans='/' ;
           while(left2<=right2){
               int mid = (left2+right2)/2;
               if(arr[mid]<=ch){
                   left2 = mid+1;
               }else{
                   ans = arr[mid];
                   right2 = mid-1;
               }
           }
            if(ans=='/'){
                System.out.println(-1);
            }else{
            System.out.println(ans);
            }
        }
    }
}
```