Binary Search

Binary search is technique of searching element into list/array. Binary search is based on divide conquer technique.

Binary search is only applied in sorted array/list of elements.

Time complexity – O(log n)
Space complexity – O(1)

Algorithm

1. start
2. key = element to be search in array
3. Initialize two pointer ‘left’=0 and ‘right’=n-1 (n is the number of element in array/list)
4. find middle of left and right mid=(left+right)/2
5. compare if arr[mid]==key than return mid (means element present at mid index)
6. compare if arr[mid] > key than modify right pointer right=mid-1
7. compare if arr[mid] < key than modify left pointer left=mid+1
8. repeat step 3 to 6 until we get the key array or left pointer is than or equal to right pointer
9. if left pointer cross right pointer means element not found than we return -1 (denotes element not found)
10. stop

Pseudo Code

```
binary_search(A,x) {
  left := 0
  right := A.length-1
  while(left<=right) {
    mid := (left+right)/2
    if (A[mid]==x)
```

```
      return mid
    else if (A[mid]>x)
      right := mid-1
    else
      left := mid+1
  }
  return -1
}
```

Implementation Java

```
class Solution {
    public int search(int[] arr, int target) {
        int n=arr.length;

        int left=0;
        int right=n-1;

        while (left<=right) {
            int mid=(left+right)/2;
            if (arr[mid]==target) {
                return mid;
            }

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

        return -1;
    }
}
```

Implementation C/C++

```
int search(int[] arr, int target) {
      int n=arr.length;

      int left=0;
      int right=n-1;

      while (left<=right) {
          int mid=(left+right)/2;
          if (arr[mid]==target) {
              return mid;
```

```
            }

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

        return -1;
    }
```