# Binary Search

**Sudo Code:**

low=0; high=n-1;

while(low <= high){

mid = (low + high)/2 [if there is possibility of overflow then “ mid=low+(high-low)/2”]

if(arr[mid] == target) return mid;

else if(arr[mid] < target) low = mid+1;

else high = mid - 1;

}

return -1;

**Recursive Binary Search:**

int bs(vector<int>&nums, int low, int high, int target){

if(low > high) return -1;

int mid = (low+high)/2;

if(nums[mid] == target) return mid;

else if(nums[mid] > target) bs(nums, low, mid-1, target);

return bs(nums, mid+1, high, target);

}

**Resource:**  BS-1

<https://www.youtube.com/watch?v=MHf6awe89xw&list=PLgUwDviBIf0pMFMWuuvDNMAkoQFi-h0ZF&index=3>

# Lower Bound & Upper Bound

**Lower Bound:** smallest element where arr[i] >= target

Sudo code:

low=0; high=n-1; ans = n;

while(low <= high){  
int mid=(low+high)/2;

if(arr[mid] >= target){

high = mid -1;

ans = mid;

}

else low = mid + 1;

}

return ans;

**Using Built in function:**

lb = lower\_bound(v.begin(), v.end(), target) – v.begin()

FOR array

lb = lower\_bound(arr, arr+n, target) – arr;

For specific range

lb =lower\_bound(arr+2, arr+7, target) – arr; [where bs will be apply in range of index 2 to 6]

**Upper Bound:** smallest element where arr[i] > target

**Sudo code:**

int low=0, high=n-1, ans=n;

while(low <= high){  
int mid = (low+high)/2;

if(arr[mid] > target){

ans = mid;

high = mid – 1;

}

else low = mid + 1;

}

return ans;

**Using Built in Funcion:**

ub = upper\_bound(v.begin(), v.end(), target) – v.begin();

# Floar and Ceil in sorted array

**Floar:** largest element where arr[i]<=target

ans=-1

if(arr[mid] <= target){

ans = arr[mid]; low=mid+1;

Ceil: smallest element where arr[i]>=target [we can say Lower Bound]

**Resource:** BS-2

<https://www.youtube.com/watch?v=6zhGS79oQ4k&list=PLgUwDviBIf0pMFMWuuvDNMAkoQFi-h0ZF&index=3>