## E - 35. Search Insert Position

Given a sorted array of distinct integers and a target value, return the index if the target is found.
If not, return the index where it would be if it were inserted in order.
You must write an algorithm with O(log n) runtime complexity.

**Example 1:**
Input: nums = [1,3,5,6], target = 5
Output: 2

**Example 2:**
Input: nums = [1,3,5,6], target = 2
Output: 1

**Example 3:**
Input: nums = [1,3,5,6], target = 7
Output: 4

**Constraints:**
1 <= nums.length <= 104
-104 <= nums[i] <= 104
nums contains distinct values sorted in ascending order.
-104 <= target <= 104

Explanation :

- The given array is sorted so we can use *Binary Search* to find the target postiion.
- First, we initialize the start , end variable
- Then start the while loop with condition of start<=end , and inside that loop , we initialize the variable *mid=start+(end-start)/2*;
- We check the if condition for binary search
- The one different is here , in search we return -1 at the end , but here , we return the start varible because *we have to find the position where the loop break.*

## Solution

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        int start=0;
        int end=nums.length-1;
        while(start<=end){
            int mid=start+(end-start)/2;
            if(nums[mid]==target)
                return mid;

            if(nums[mid]>target){
                end=mid-1;
            else if(nums[mid]<target){
                start=mid+1;
        }
        return start;
    }
}
```