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
```

Example 4:

```
Input: nums = [1,3,5,6], target = 0

Output: 0
```

Example 5:

```
Input: nums = [1], target = 0

Output: 0
```

Constraints:

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

Program :

```
class Solution {
    public int searchInsert(int[] nums, int target) {
```

```
    int i=0;
    int j=nums.length-1;

    while(i<=j){
        int mid = (i+j)/2;

        if(target > nums[mid]){
            i=mid+1;
        }else if(target < nums[mid]){
            j=mid-1;
        }else{
            return mid;
        }
    }

    return i;



    }
}
```

Output :

Accepted

Runtime: 0 ms

Your input

```
[1,3,5,6]
5
```

Output

```
2
```

Diff

Expected

2