Search Insert Position

Given a sorted array 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 may assume no duplicates in the array.

Here are few examples.
`[1,3,5,6]`, 5 → 2
`[1,3,5,6]`, 2 → 1
`[1,3,5,6]`, 7 → 4
`[1,3,5,6]`, 0 → 0

Solution 1

```
    public int searchInsert(int[] A, int target) {
        int low = 0, high = A.length-1;
        while(low<=high){
            int mid = (low+high)/2;
            if(A[mid] == target) return mid;
            else if(A[mid] > target) high = mid-1;
            else low = mid+1;
        }
        return low;
    }
```

written by AmmsA original link here

Solution 2

If there are duplicate elements equal to*target*, my code will always return the one with smallest index.

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int low = 0, high = nums.size()-1;

        // Invariant: the desired index is between [low, high+1]
        while (low <= high) {
            int mid = low + (high-low)/2;

            if (nums[mid] < target)
                low = mid+1;
            else
                high = mid-1;
        }

        // (1) At this point, low > high. That is, low >= high+1
        // (2) From the invariant, we know that the index is between [low, high+1
], so low <= high+1. Follwing from (1), now we know low == high+1.
        // (3) Following from (2), the index is between [low, high+1] = [low, low
], which means that low is the desired index
        //     Therefore, we return low as the answer. You can also return high+1
as the result, since low == high+1
        return low;
    }
};
```

written by a0806449540 original link here

Solution 3

I think the solution does not need a lot of if statement. Only two cases: 1 if found, just return current index 2 if not found, return next index where the search end

```
int search(int A[], int start, int end, int target) {
    if (start > end) return start;
    int mid = (start + end) / 2;
    if (A[mid] == target) return mid;
    else if (A[mid] > target) return search(A, start, mid - 1, target);
    else return search(A, mid + 1, end, target);
}
int searchInsert(int A[], int n, int target) {
    return search(A, 0, n - 1, target);
}
```

written by yuruofeifei original link here