LeetCode\_35. Search Insert Position

# 题目描述

题目：**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.

**Example 1**:

Input: [1,3,5,6], 5

Output: 2

**Example 2**:

Input: [1,3,5,6], 2

Output: 1

**Example 3:**

Input: [1,3,5,6], 7

Output: 4

**Example 4**:

Input: [1,3,5,6], 0

Output: 0

# 思路分析：

**思路分析：基于二分查找修改。**二分查找中target与arr[midIndex]比较有3种情况，当相等时，相等于在arr中存在target直接返回midIndex即可；显然需要修改的是大于和小于的情况。

* target>arr[midIndex]时，有两种情况下需要结束，一是：midIndex==high，代表target大于arr中任何一个元素，此时返回midIndex+1；二是：midIndex没有达到high，且target<arr[midIndex+1]，此时也返回midIndex+1;如果这两种情况都不满足，则需要进行low=midIndex+1，循环执行。
* target<arr[midIndex]时，同样有两种情况下需要结束，一是：midIndex==low，代表target小于arr中任何一个元素，此时返回midIndex；二是：midIndex没有达到low，且target>arr[midIndex-1]，此时也返回midIndex;如果这两种情况都不满足，则需要进行high=midIndex-1，循环执行。

**注意：这里一定要注意防止数组索引越界**。所以首先判断对应的第一种情况，如midIndex是否等于high或者low，然后进行arr[midIndex+1]或arr[midIndex-1]操作。利用||连接，前一个条件若满足了，后面的就不会执行。

# Java代码实现

## 二分查找算法的Java代码：

public int binarySearch(int[] arr,int low ,int high,int target){

while(low <= high){

**int midIndex = (low + high)/2;**

if(target < arr[midIndex] ){

high = midIndex -1;

}else if(target > arr[midIndex]){

low = midIndex + 1;

}else{

return midIndex;

}

}

return -1;//不存在返回-1

}

## 二分查找及插入的Java代码：

* // target > arr[midIndex];两种情况下对应跳出：1、midIndex达到high，target大于数组中任何一个值,返回midIndex+1；

2、arr[midIndex]< target < arr[midIndex + 1]，返回midIndex+1；

* // target < arr[midIndex];两种情况下对应跳出：1、midIndex达到low，target小于数组中任何一个值,返回midIndex；

// 2、arr[midIndex-1]< target < arr[midIndex]，返回midIndex；

* //target == arr[midIndex];直接返回midIndex

class Solution {

public int searchInsert(int[] arr, int target) {

int low = 0;

int high = arr.length - 1;

while(true){

int midIndex = (low + high)/2;

if(target > arr[midIndex]){

**if(midIndex == high||target < arr[midIndex + 1])**

**return midIndex+1;**

**low = midIndex + 1;**

}else if(target < arr[midIndex]){

**if(midIndex == low||target > arr[midIndex - 1])**

**return midIndex;**

**high = midIndex - 1;**

}else{

return midIndex;

}

}

}

}