琪石第五期算法小组
Binary Search

Group A-Jiahao Zhu, Apr 20

Time Complexity

时间复杂度可以衡量算法的efficiency。 但是在面试中，我们有时可以通过时间复杂度来倒推所需要的算法。

复杂度	我的思路
O（1）	很少见， bit manipulation….
O (log(n))	**二分法**
O（n）	遍历
O (n*log(n))	可能和排序有关，quick, merge….
O(n^2)	Matrix相关操作, LU, Cholesky decomposition

二分法 O(Log(n))

- T(N) = T(N/2)+O(1)
- 通过O(1)的时间，把size N的问题变成Size N/2
- T(N) = T(N/2) +O(1) = T(N/4) + O(1) + O(1) +……
- 那么我们要这样操作多少次呢？ N/(2^k) =1, solve k, k = log(N)
- 所以我们得到 T（N） = O(log(N)) , log(N)个O(1)

- Question: 如果需要O(N)的时间把Size N 的问题变成 Size N/2。那时间复杂度是什么？ N*Log(N)？ NO!

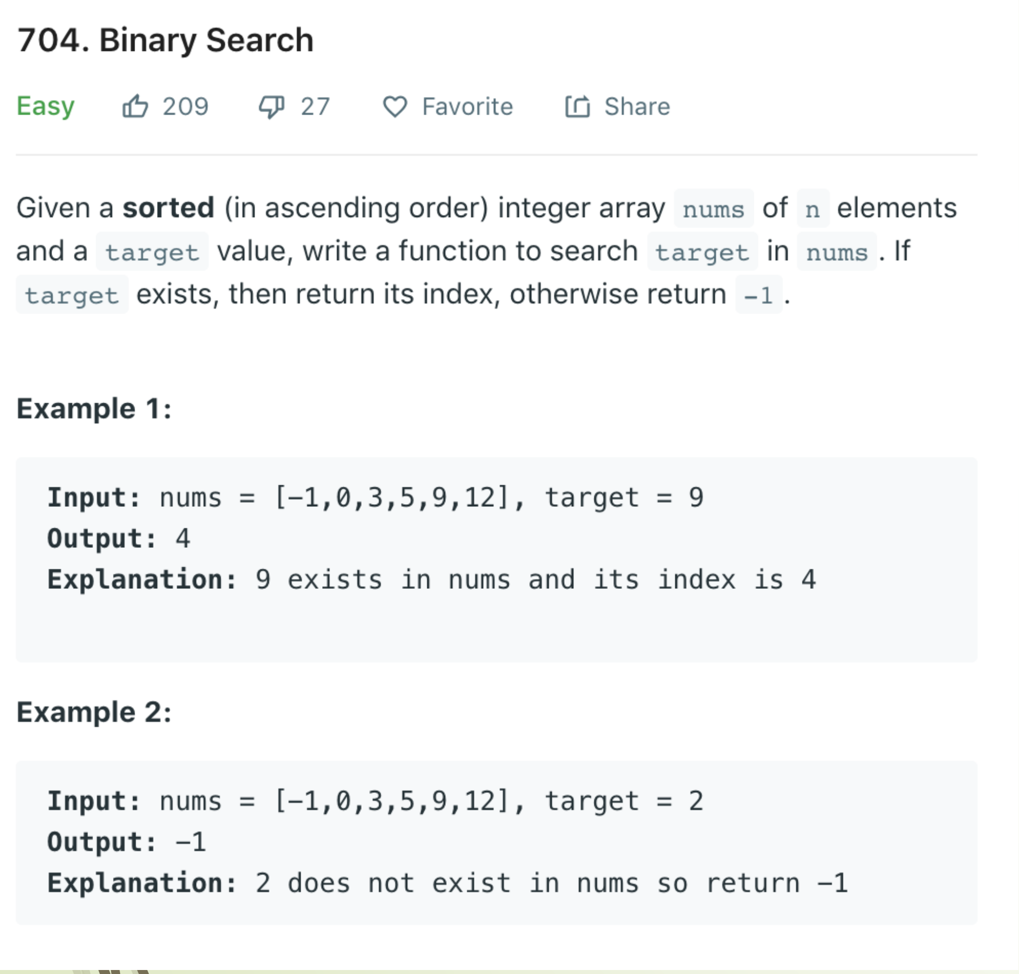
二分法两种基本Implementations

Recursive & iterative

Pay Attention to Boundary Condition !!

- LeetCode 704: Binary Search

面试中合适地选择这两种方法之一



704. Binary Search

Easy 209 27 Favorite Share

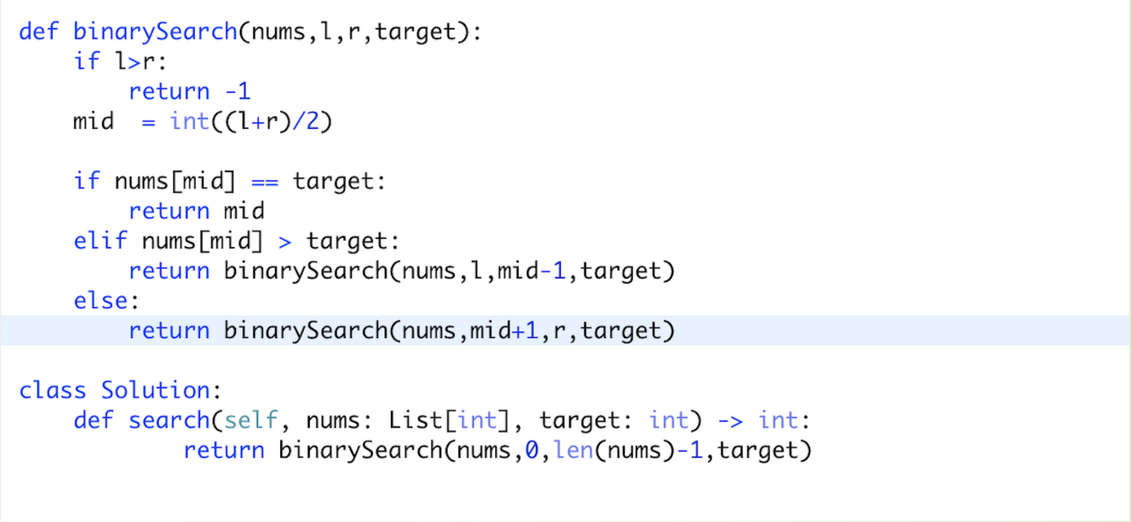
Given a **sorted** (in ascending order) integer array `nums` of `n` elements and a `target` value, write a function to search `target` in `nums`. If `target` exists, then return its index, otherwise return `-1`.

Example 1:

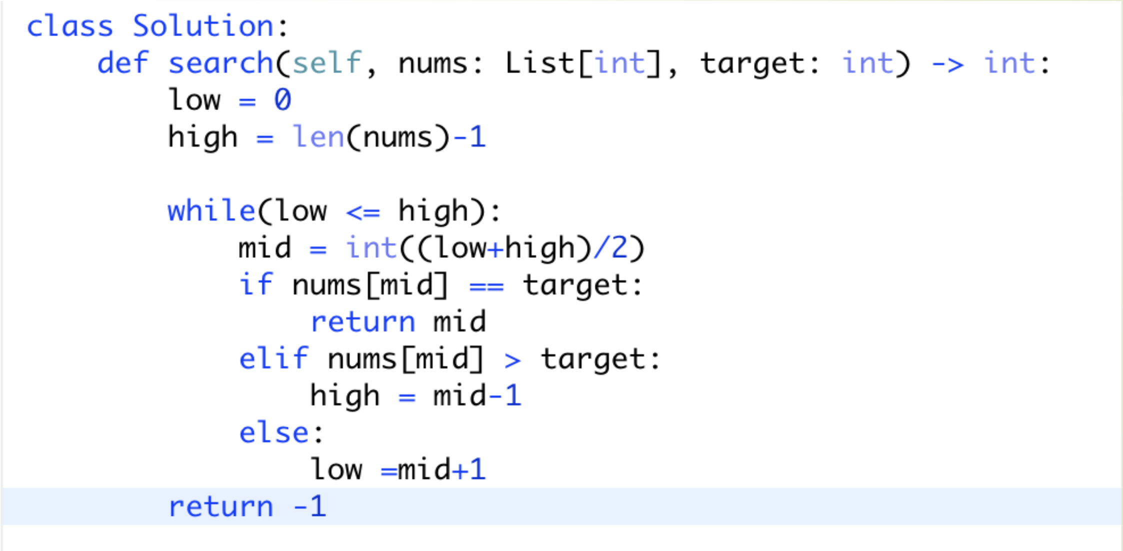
```
Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4
```

Example 2:

```
Input: nums = [-1,0,3,5,9,12], target = 2
Output: -1
Explanation: 2 does not exist in nums so return -1
```



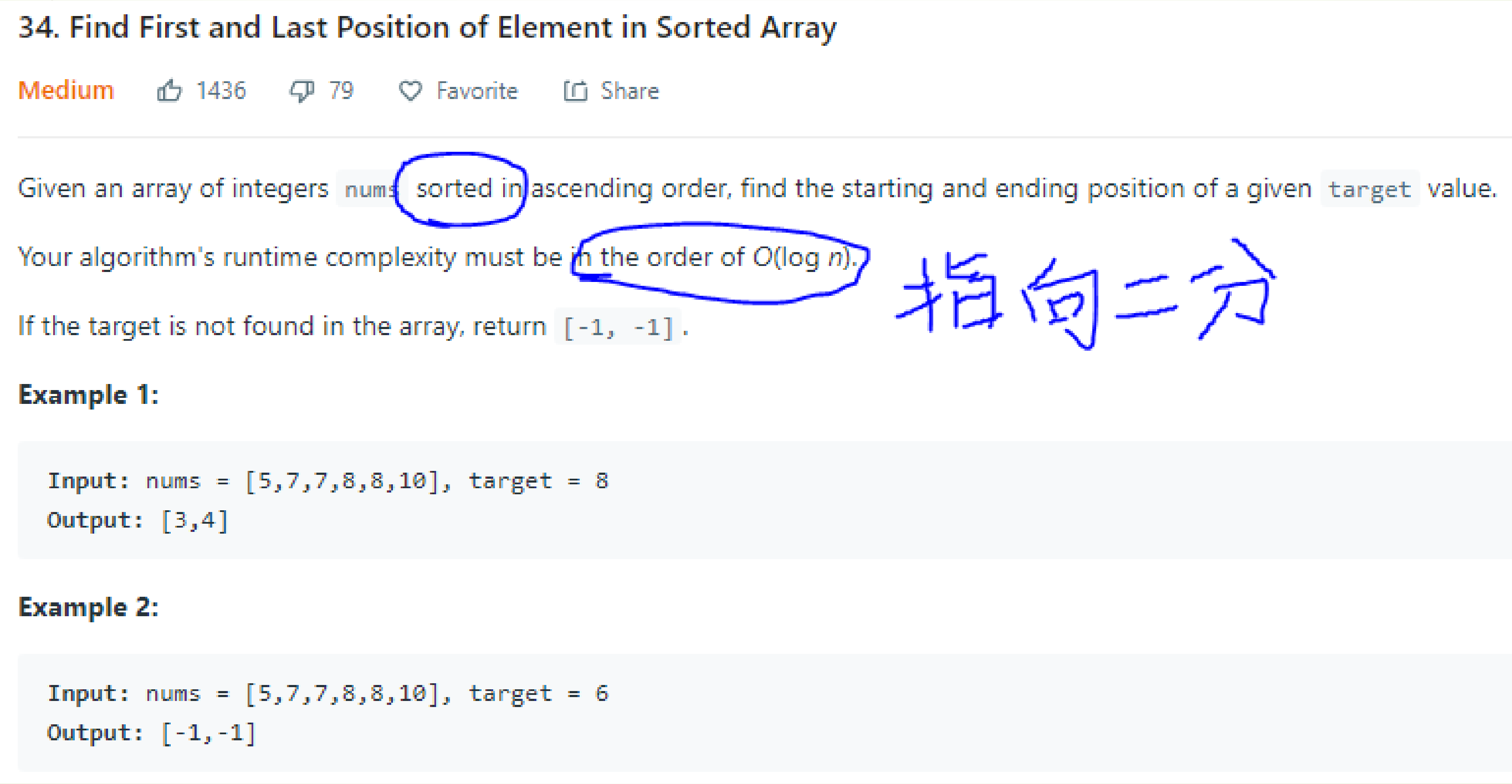
```
def binarySearch(nums,l,r,target):
    if l>r:
        return -1
    mid  = int((l+r)/2)

    if nums[mid] == target:
        return mid
    elif nums[mid] > target:
        return binarySearch(nums,l,mid-1,target)
    else:
        return binarySearch(nums,mid+1,r,target)

class Solution:
    def search(self, nums: List[int], target: int) -> int:
            return binarySearch(nums,0,len(nums)-1,target)
```



```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        low = 0
        high = len(nums)-1

        while(low <= high):
            mid = int((low+high)/2)
            if nums[mid] == target:
                return mid
            elif nums[mid] > target:
                high = mid-1
            else:
                low =mid+1
        return -1
```

Example1: Find the First/Last appearance

- LeetCode 34



34. Find First and Last Position of Element in Sorted Array

Given an array of integers `nums` sorted in ascending order, find the starting and ending position of a given `target` value.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

指向二分

If the target is not found in the array, return `[-1, -1]`.

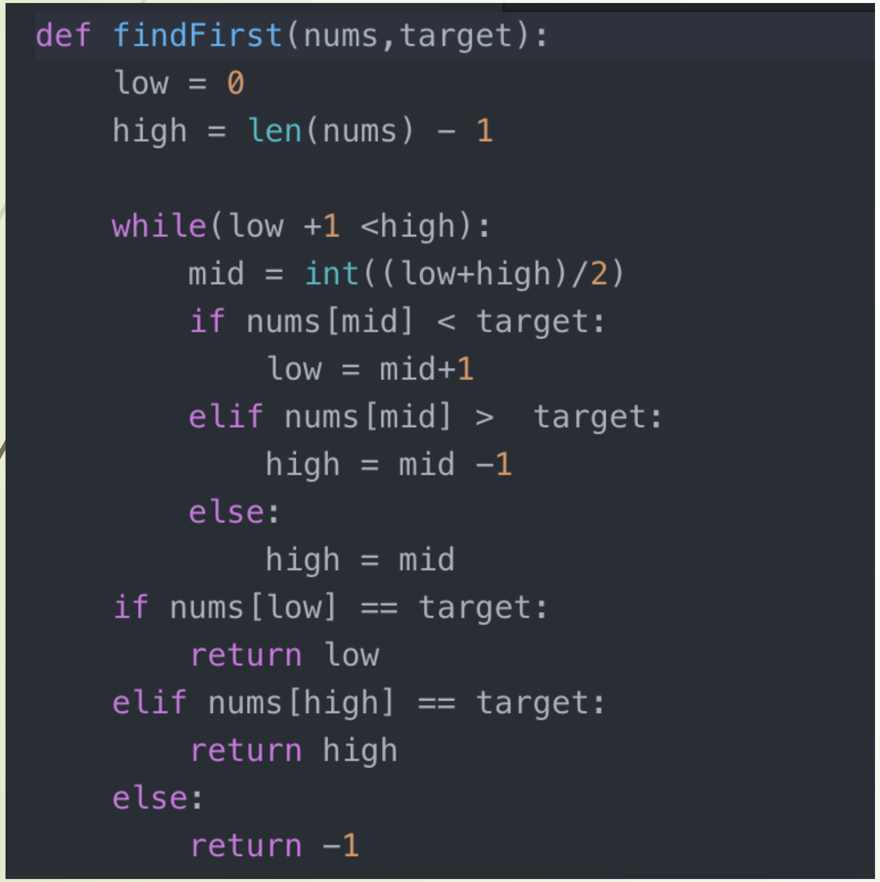
Example 1:

```
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
```

Example 2:

```
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
```

Breakdown: [5,7,7,8,8,10,10] find the index of first 10 and last 10

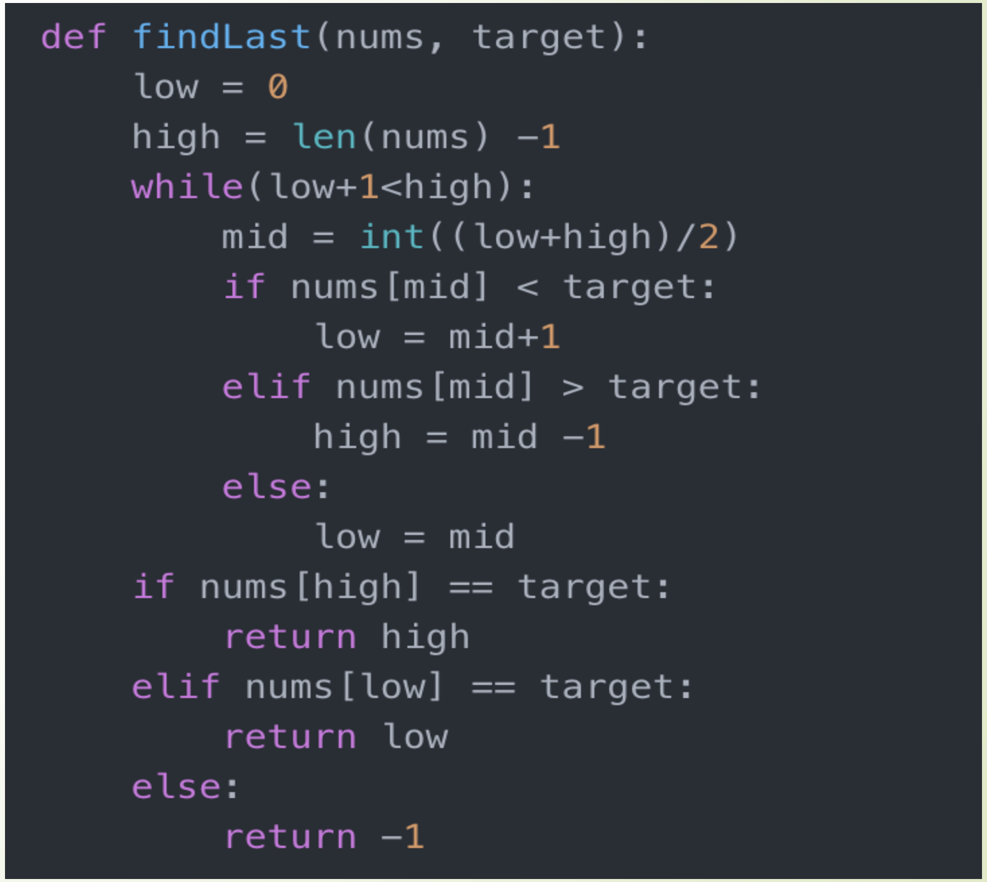


```
def findFirst(nums,target):
    low = 0
    high = len(nums) - 1

    while(low +1 <high):
        mid = int((low+high)/2)
        if nums[mid] < target:
            low = mid+1
        elif nums[mid] >  target:
            high = mid -1
        else:
            high = mid
    if nums[low] == target:
        return low
    elif nums[high] == target:
        return high
    else:
        return -1
```

Again, Please pay attention to the boundary condition.

In my implementation, “low < high” works fine for findFirst, but will result in infinite loop for findLast.

Instead, I used a more general boundary “low+1>high” , meaning to stop when low and high are next to each other, and to check low, high with preference.

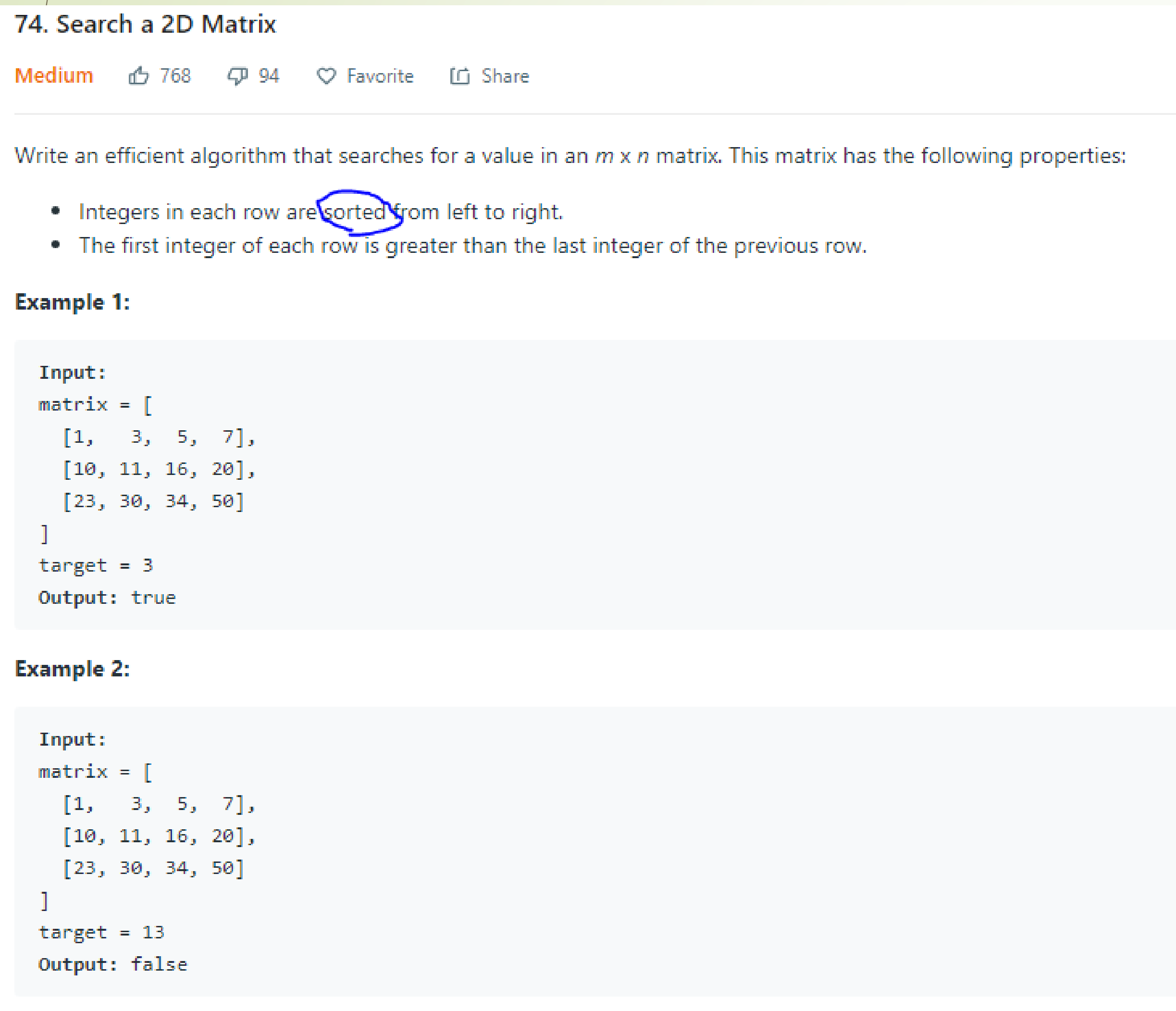


```
def findLast(nums, target):
    low = 0
    high = len(nums) -1
    while(low+1<high):
        mid = int((low+high)/2)
        if nums[mid] < target:
            low = mid+1
        elif nums[mid] > target:
            high = mid -1
        else:
            low = mid
    if nums[high] == target:
        return high
    elif nums[low] == target:
        return low
    else:
        return -1
```

Example 2: Extend to Matrix

Fun Fact: horse
racing in green book

- LeetCode 74



74. Search a 2D Matrix

Medium 768 94 Favorite Share

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

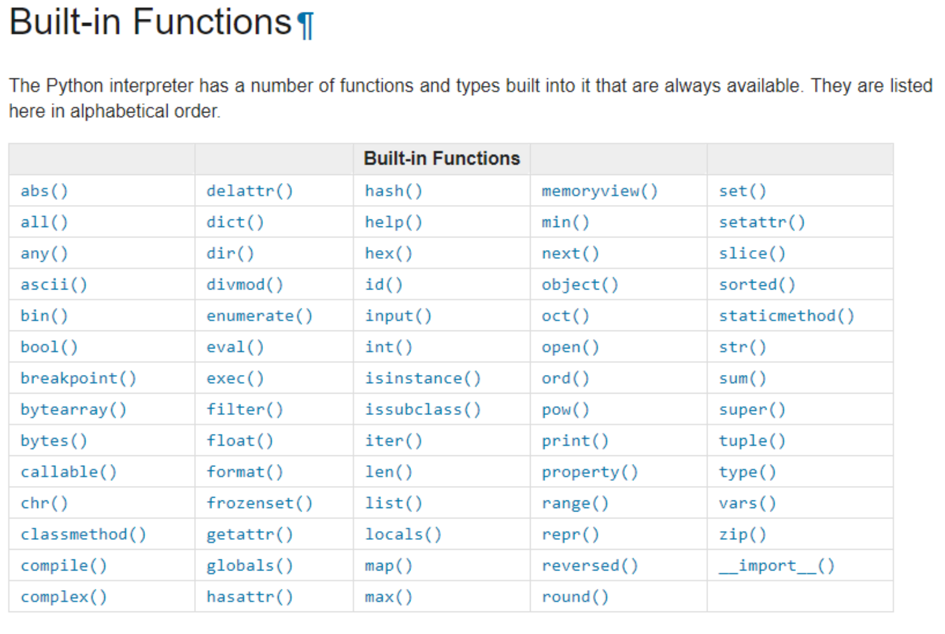
Example 2:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 13
Output: false
```

Well, it's Python:
flatList = [item for item in row for row in Matrix]

It is **"Pythonic"**, faster than a traditional double loop. But still, you need to go through all elements, at least O(n).

A little note on "Pythonic":



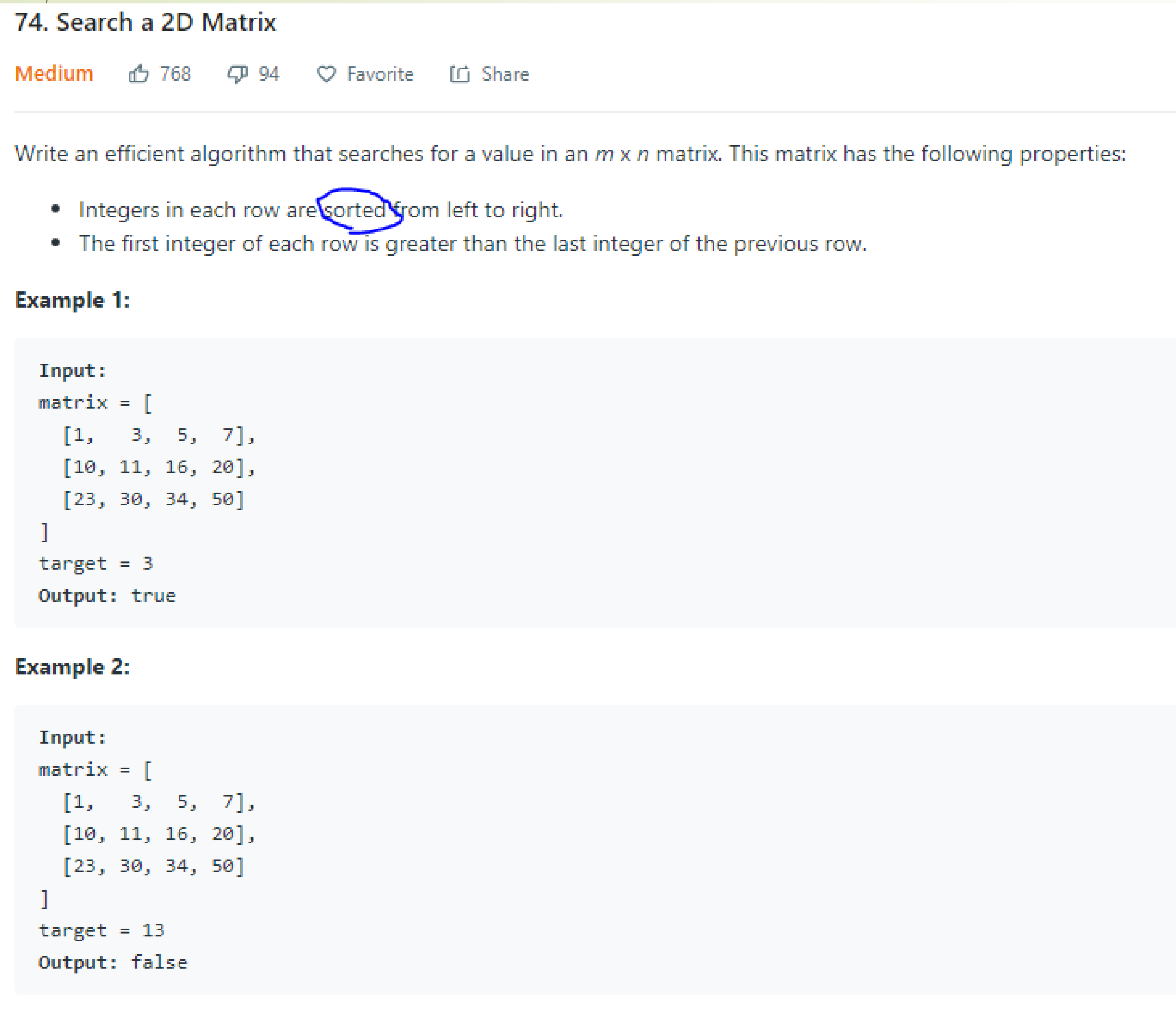
Built-in Functions¶

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Example 2: Extend to Matrix

- LeetCode 74



74. Search a 2D Matrix

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

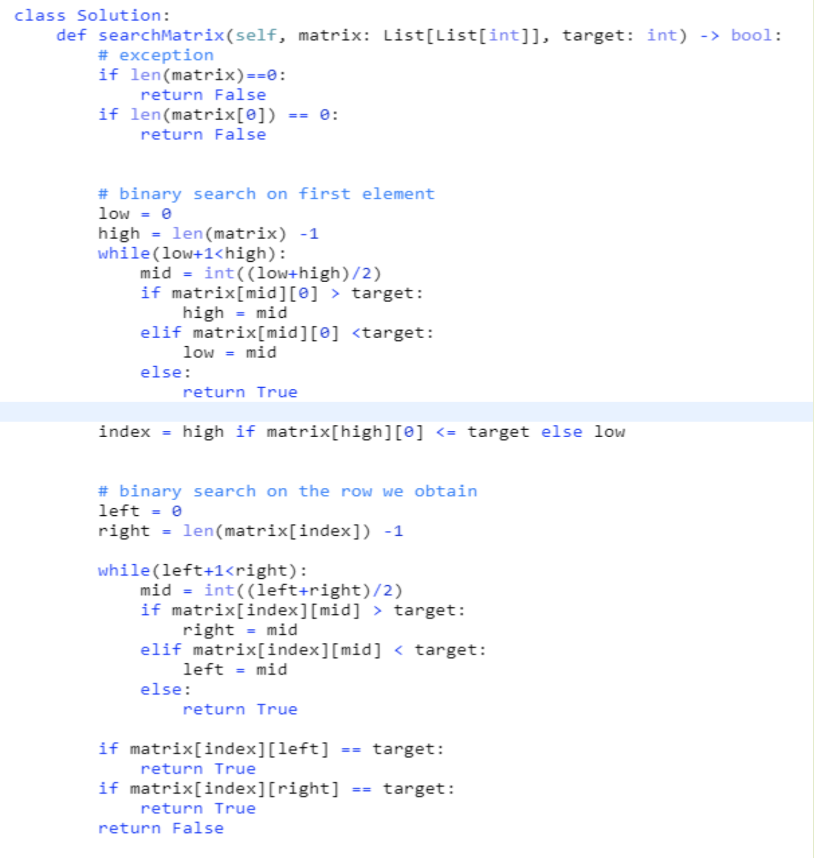
Example 2:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 13
Output: false
```

Let's get back to this problem, and use Binary Search on the matrix directly

Idea:

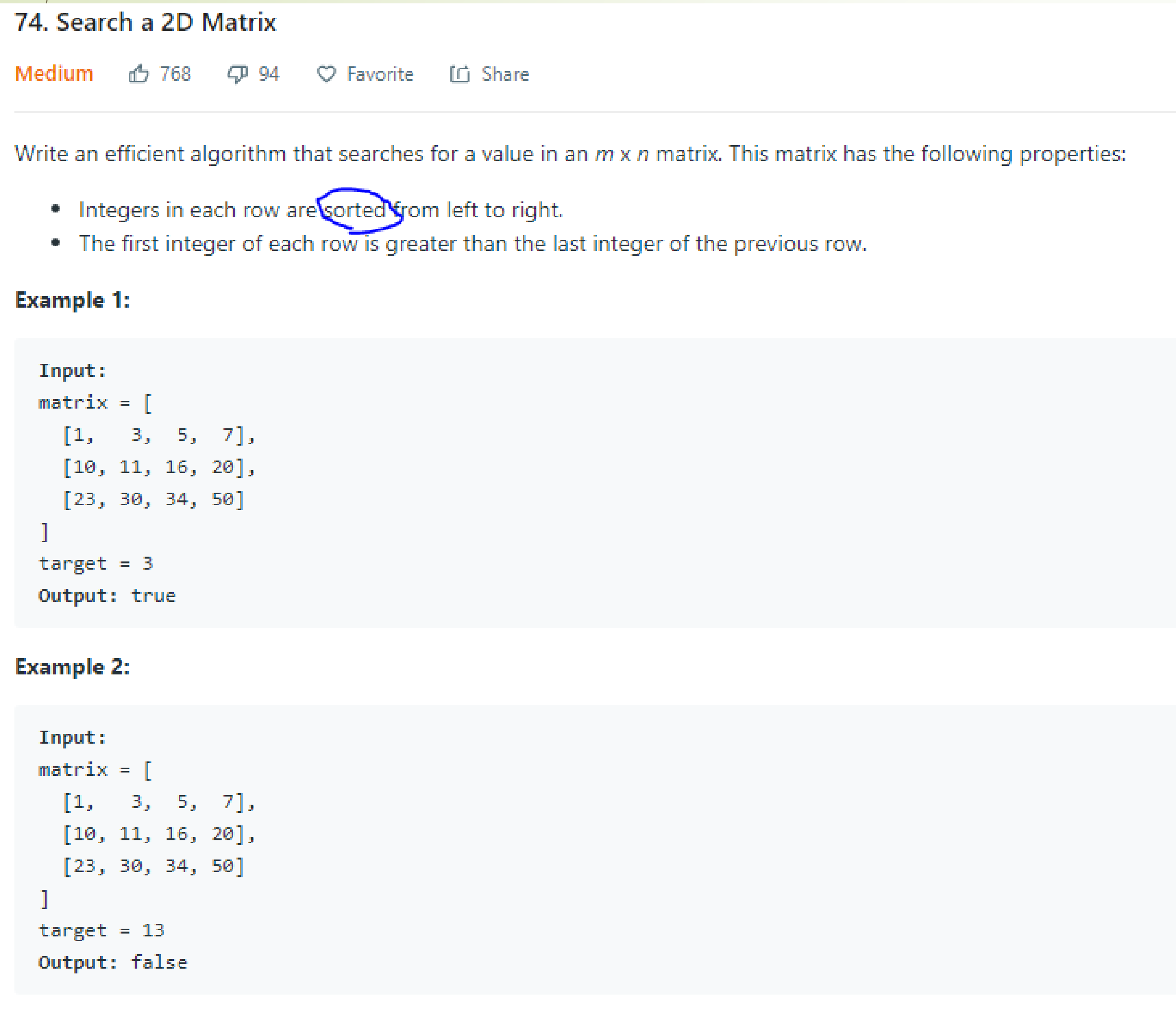
1. For each row, find the last element smaller than target
2. Do a binary search on that row



```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        # exception
        if len(matrix)==0:
            return False
        if len(matrix[0]) == 0:
            return False


        # binary search on first element
        low = 0
        high = len(matrix) -1
        while(low+1<high):
            mid = int((low+high)/2)
            if matrix[mid][0] > target:
                high = mid
            elif matrix[mid][0] <target:
                low = mid
            else:
                return True

        index = high if matrix[high][0] <= target else low


        # binary search on the row we obtain
        left = 0
        right = len(matrix[index]) -1

        while(left+1<right):
            mid = int((left+right)/2)
            if matrix[index][mid] > target:
                right = mid
            elif matrix[index][mid] < target:
                left = mid
            else:
                return True

        if matrix[index][left] == target:
            return True
        if matrix[index][right] == target:
            return True
        return False
```

Example 2: Extend to Matrix

- LeetCode 74



74. Search a 2D Matrix

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

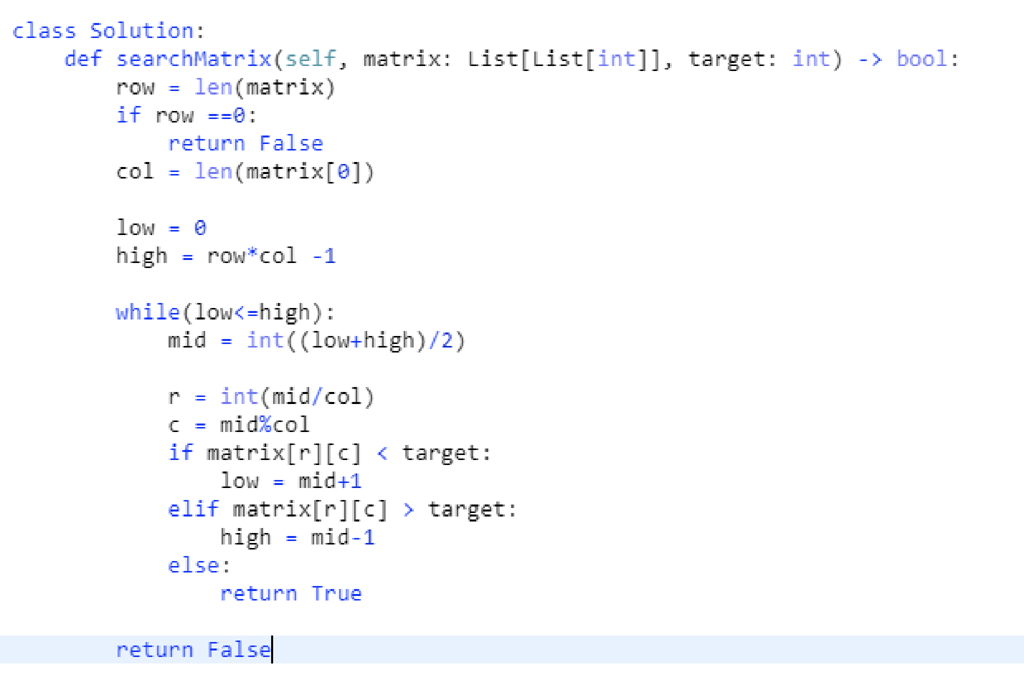
Example 2:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 13
Output: false
```

Idea:

1. For each row, find the last element smallerthan target
2. Do a binary search on that row

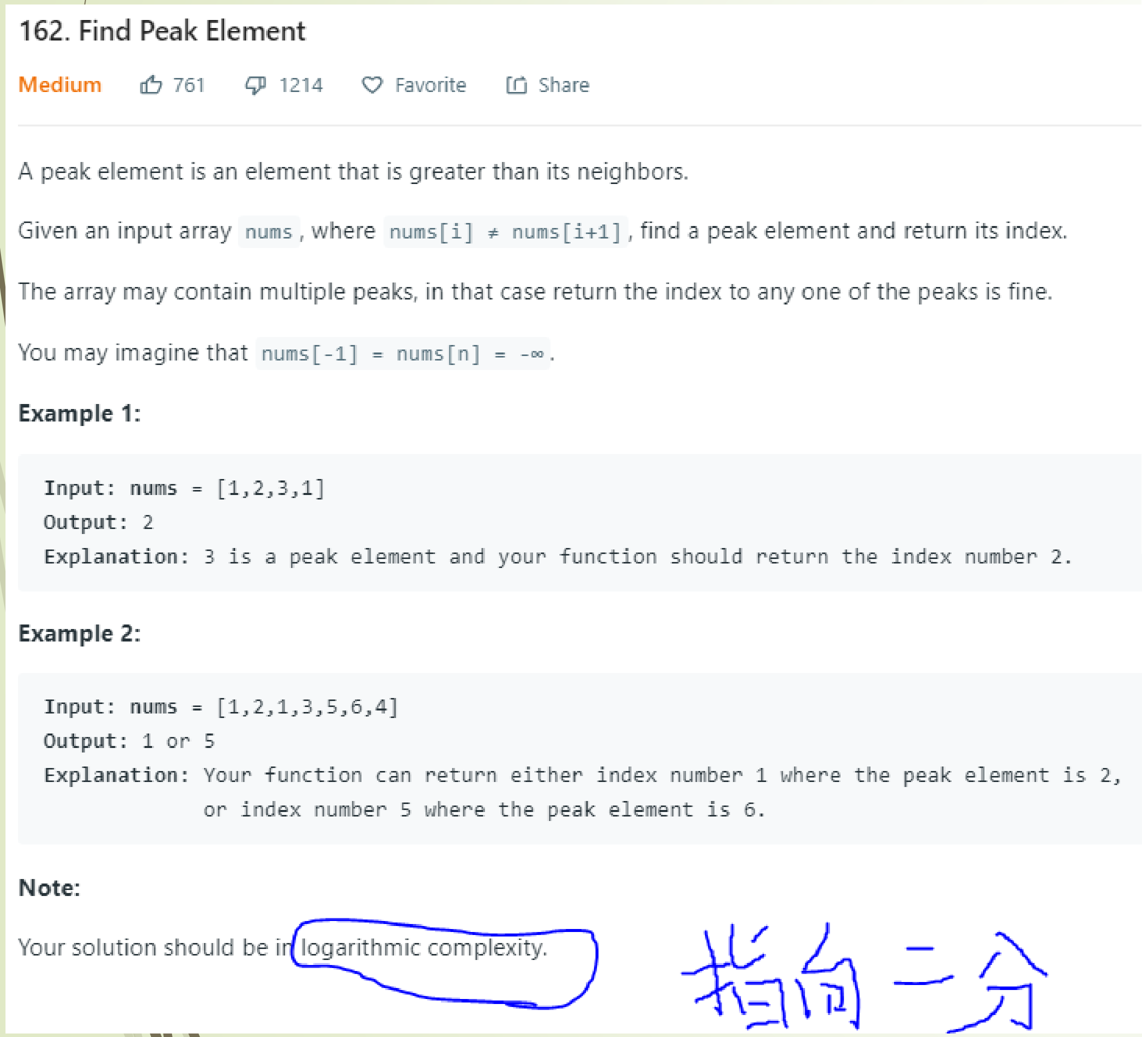
Idea 2:
Binary Search to take the matrix as a plain list



```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        row = len(matrix)
        if row ==0:
            return False
        col = len(matrix[0])

        low = 0
        high = row*col -1

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

            r = int(mid/col)
            c = mid%col
            if matrix[r][c] < target:
                low = mid+1
            elif matrix[r][c] > target:
                high = mid-1
            else:
                return True

        return False
```

Example3:Extend to more general questions

- LeetCode 162



162. Find Peak Element

Medium 👍 761 👎 1214 ♡ Favorite Share

A peak element is an element that is greater than its neighbors.

Given an input array `nums`, where `nums[i] ≠ nums[i+1]`, find a peak element and return its index.

The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.

You may imagine that `nums[-1] = nums[n] = -∞`.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

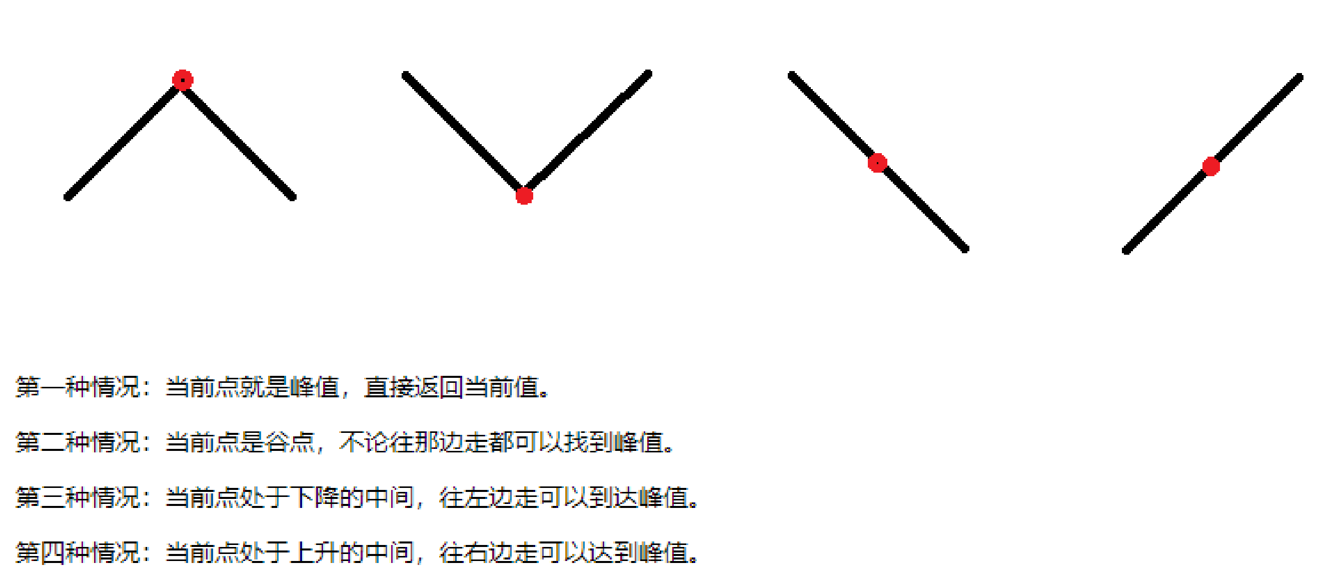
Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 1 or 5
Explanation: Your function can return either index number 1 where the peak element is 2,
             or index number 5 where the peak element is 6.
```

Note:

Your solution should be in logarithmic complexity.

指向二分



第一种情况：当前点就是峰值，直接返回当前值。

第二种情况：当前点是谷点，不论往那边走都可以找到峰值。

第三种情况：当前点处于下降的中间，往左边走可以到达峰值。

第四种情况：当前点处于上升的中间，往右边走可以达到峰值。

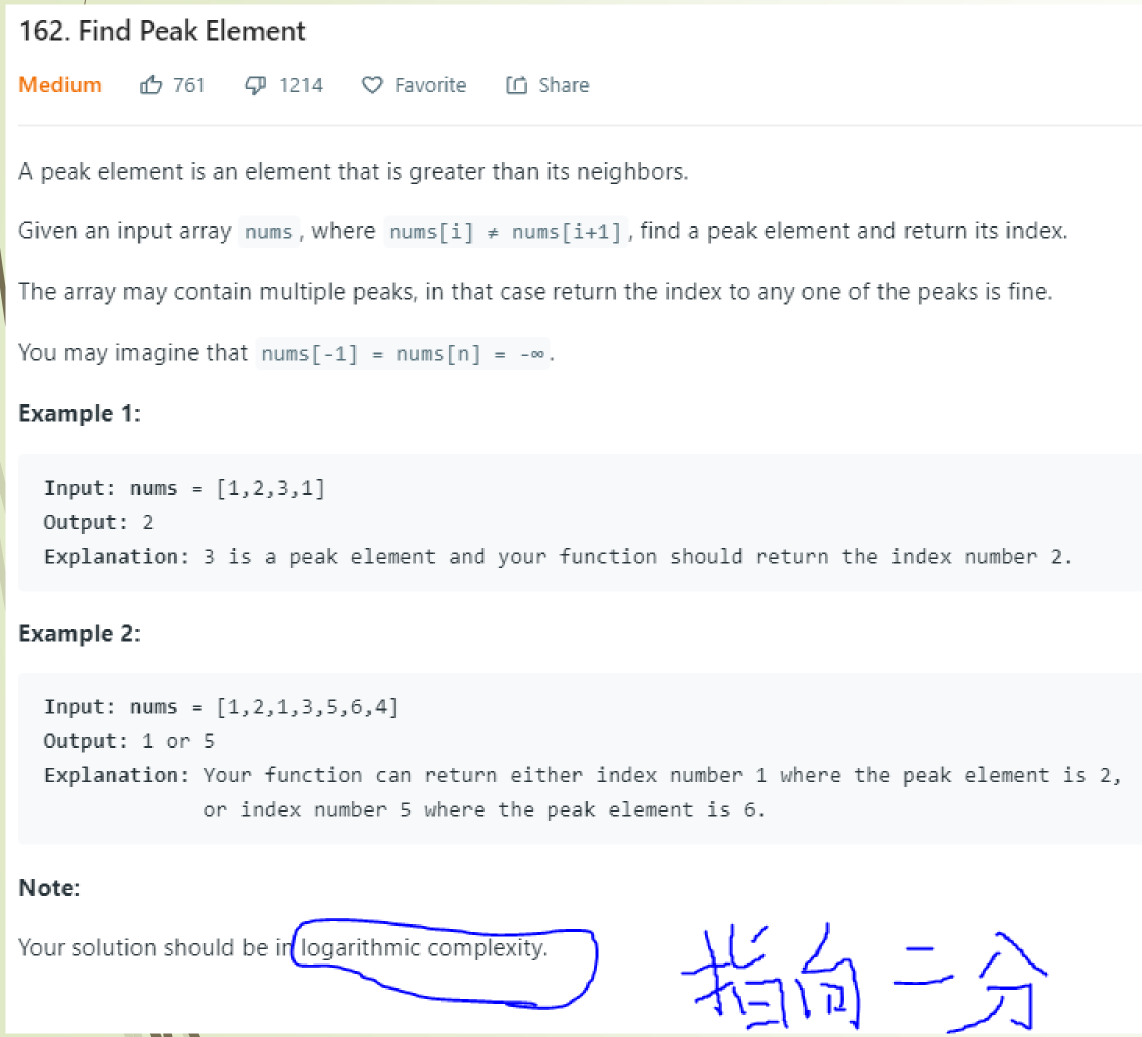
保留有solution的那一半！

Source: https://www.cnblogs.com/Raising-Sun/p/5747072.html

Fun fact: Maximum contingency array Frequently appear in buy side coding interviews (backtesting)

Example3:Extend to more general questions

- LeetCode 162



162. Find Peak Element

Medium 761 1214 Favorite Share

A peak element is an element that is greater than its neighbors.

Given an input array `nums`, where `nums[i] ≠ nums[i+1]`, find a peak element and return its index.

The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.

You may imagine that `nums[-1] = nums[n] = -∞`.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.
```

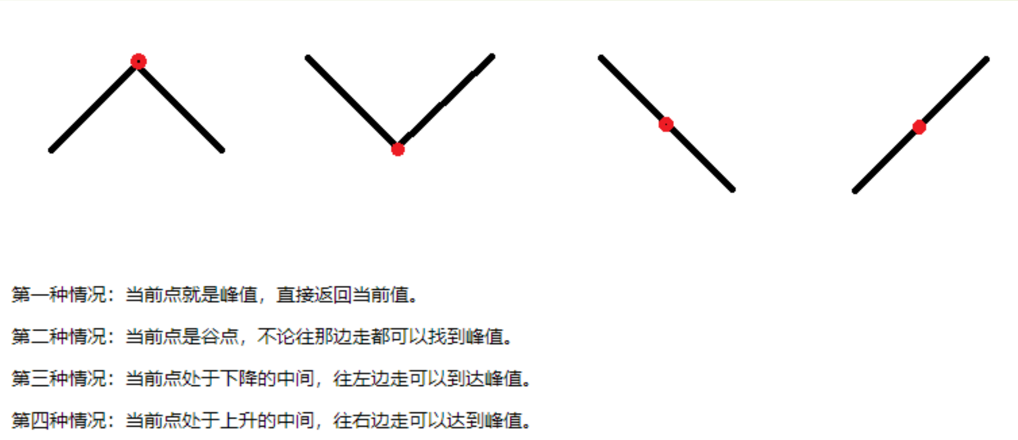
Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 1 or 5
Explanation: Your function can return either index number 1 where the peak element is 2,
             or index number 5 where the peak element is 6.
```

Note:

Your solution should be in logarithmic complexity.

指向二分

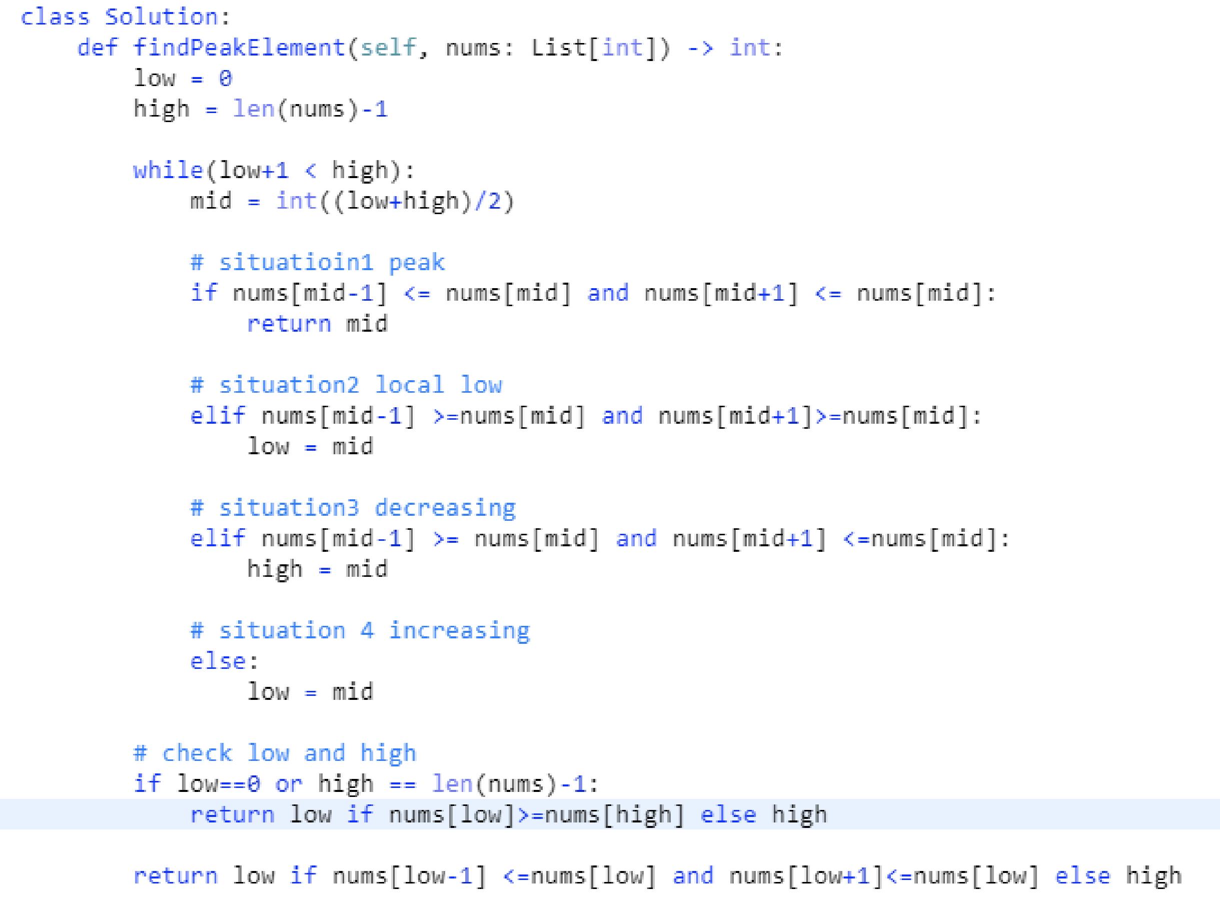


第一种情况：当前点就是峰值，直接返回当前值。

第二种情况：当前点是谷点，不论往那边走都可以找到峰值。

第三种情况：当前点处于下降的中间，往左边走可以到达峰值。

第四种情况：当前点处于上升的中间，往右边走可以达到峰值。



```python
class Solution:
    def findPeakElement(self, nums: List[int]) -> int:
        low = 0
        high = len(nums)-1

        while(low+1 < high):
            mid = int((low+high)/2)

            # situatioin1 peak
            if nums[mid-1] <= nums[mid] and nums[mid+1] <= nums[mid]:
                return mid

            # situation2 local low
            elif nums[mid-1] >=nums[mid] and nums[mid+1]>=nums[mid]:
                low = mid

            # situation3 decreasing
            elif nums[mid-1] >= nums[mid] and nums[mid+1] <=nums[mid]:
                high = mid

            # situation 4 increasing
            else:
                low = mid

        # check low and high
        if low==0 or high == len(nums)-1:
            return low if nums[low]>=nums[high] else high

        return low if nums[low-1] <=nums[low] and nums[low+1]<=nums[low] else high
```

Homework Summary (in Leetcode#)

- Required:
- 34 (First & Last Appearance)
- 74 (Matrix)
- 162 (Peak)
- 153 (minimum rotated Sorted Array)
- Suggested:
- 704 (Easy, if you do not know what is binary search, do this first)
- 81 (Extension to the matrix problem)
- 302 (hard, if you want challenge. The problem needs subscription on leetcode, you can find it somewhere else by searching in google. It is popular)

Thank you !!