5. Binary search

704. Binary Search

Created	@October 14, 2022 9:54 PM
Difficulty	Easy
LC Url	https://leetcode.com/problems/binary-search/
Importance	
Tag	Binary search NEET
Video	

Given an array of integers `nums` which is sorted in ascending order, and an integer `target`, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return `-1`.

You must write an algorithm with `O(log n)` runtime complexity.

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
```

Constraints:

- `1 <= nums.length <= 10 4`
- `10 4` `< nums[i], target < 10 4`
- All the integers in `nums` are **unique**.
- `nums` is sorted in ascending order.

Solution

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

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

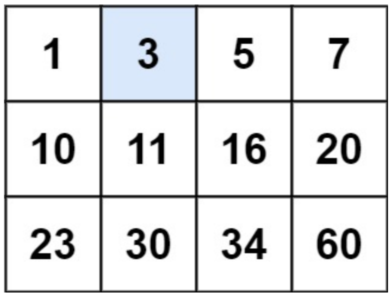
74. Search a 2D Matrix

Created	@July 9, 2021 12:03 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/search-a-2d-matrix/
Importance	
Tag	Binary search NEET
Video	

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.

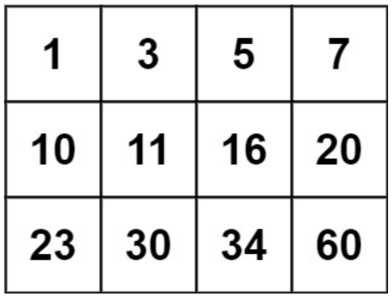
Example 1:



1	3	5	7
10	11	16	20
23	30	34	60

```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true
```

Example 2:



1	3	5	7
10	11	16	20
23	30	34	60

```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13
Output: false
```

Constraints:

- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 100`
- `10 4  <= matrix[i][j], target <= 10 4`

Solution

https://leetcode.cn/problems/search-a-2d-matrix-ii/solution/sou-suo-er-wei-ju-zhen-ii-by-leetcode-so-9hcx/

https://leetcode.cn/problems/search-a-2d-matrix-ii/solution/jian-dan-yi-dong-javac-pythonjsgo-sou-su-3mh6/

```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        if not matrix or target is None:
            return False

        rows, cols = len(matrix), len(matrix[0])
        low, high = 0, rows * cols - 1

        while low + 1 < high:
            mid = (low + high) // 2
            mid_num = matrix[mid // cols][mid % cols]

            if mid_num < target:
                low = mid
            elif mid_num > target:
                high = mid
            else:
                return True

        if matrix[low // cols][low % cols] == target:
            return True

        if matrix[high // cols][high % cols] == target:
            return True

        return False
```

```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        if not matrix or target is None:
            return False

        rows, cols = len(matrix), len(matrix[0])
        low, high = 0, rows * cols - 1

        while low <= high:
            mid = (low + high) // 2
            num = matrix[mid // cols][int(mid % cols)]

            if num == target:
                return True
            elif num < target:
                low = mid + 1
            else:
                high = mid - 1

        return False
```

A Python binary search solution - O(logn) - LeetCode Discuss

Level up your coding skills and quickly land a job. This is the best place to expand your knowledge and get prepared for your next interview.

https://leetcode.com/problems/search-a-2d-matrix/discuss/26201/A-Python-binary-search-solution-O(logn)

LeetCode

875. Koko Eating Bananas

Created	@October 15, 2022 2:30 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/koko-eating-bananas/
Importance	
Tag	Binary search NEET
Video	

Koko loves to eat bananas. There are `n` piles of bananas, the `i` `th` pile has `piles[i]` bananas. The guards have gone and will come back in `h` hours.

Koko can decide her bananas-per-hour eating speed of `k`. Each hour, she chooses some pile of bananas and eats `k` bananas from that pile. If the pile has less than `k` bananas, she eats all of them instead and will not eat any more bananas during this hour.

Koko likes to eat slowly but still wants to finish eating all the bananas before the guards return.

Return *the minimum integer* `k` *such that she can eat all the bananas within* `h` *hours*.

Example 1:

```
Input: piles = [3,6,7,11], h = 8
Output: 4
```

Example 2:

```
Input: piles = [30,11,23,4,20], h = 5
Output: 30
```

Example 3:

```
Input: piles = [30,11,23,4,20], h = 6
Output: 23
```

Constraints:

- `1 <= piles.length <= 10 4`
- `piles.length <= h <= 10 9`
- `1 <= piles[i] <= 10 9`

Solution

二分查找定位速度（最大值最小化问题，Java） - 爱吃香蕉的珂珂 - 力扣（LeetCode）

```
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) -> int:
        max_val = max(piles)
        left, right = 1, max_val

        while left + 1 < right:
            speed = (left + right) // 2
            if self.total_time(piles, speed) > h:
                left = speed
            else:
                right = speed

        if self.total_time(piles, left) <= h:
            return left

        if self.total_time(piles, right) <= h:
            return right

        return -1

    def total_time(self, piles, speed):
        t = 0
        for pile in piles:
            t += (pile + speed - 1) // speed
        return t
```

复杂度分析：

- 时间复杂度：$O(N \log \max(piles))$，这里 N 表示数组 `piles` 的长度。我们在 $[1, \max piles]$ 里使用二分查找定位最小速度，而每一次执行判别函数的时间复杂度是 $O(N)$；
- 空间复杂度：$O(1)$，算法只使用了常数个临时变量。

33. Search in Rotated Sorted Array

Created	@July 16, 2020 2:18 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/search-in-rotated-sorted-array/
Importance	****
Tag	Binary search NEET
Video	https://www.youtube.com/watch?v=7XOQIMZVIjA

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`, *or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

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

Constraints:

- `1 <= nums.length <= 5000`
- `10 4  <= nums[i] <= 10 4`
- All values of `nums` are **unique**.
- `nums` is an ascending array that is possibly rotated.
- `10 4  <= target <= 10 4`

Solution

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums) - 1
        while left <= right:
            mid = (left + right) // 2
            if nums[mid] == target:
                return mid

            if target >= nums[0]:
                if nums[mid] >= nums[0] and target > nums[mid]:
                    left = mid + 1
                else:
                    right = mid - 1
            else:
                if nums[mid] >= nums[0] or target > nums[mid]:
                    left = mid + 1
                else:
                    right = mid - 1
        return -1
```

153. Find Minimum in Rotated Sorted Array

Created	@October 15, 2022 3:42 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/
Importance	
Tag	Binary search, NEET
Video	

Suppose an array of length `n` sorted in ascending order is **rotated** between `1` and `n` times. For example, the array `nums = [0,1,2,4,5,6,7]` might become:

- `[4,5,6,7,0,1,2]` if it was rotated `4` times.
- `[0,1,2,4,5,6,7]` if it was rotated `7` times.

Notice that **rotating** an array `[a[0], a[1], a[2], ..., a[n-1]]` 1 time results in the array `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]`.

Given the sorted rotated array `nums` of **unique** elements, return *the minimum element of this array*.

You must write an algorithm that runs in `O(log n) time.`

Example 1:

```
Input: nums = [3,4,5,1,2]
Output: 1
Explanation: The original array was [1,2,3,4,5] rotated 3 times.
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2]
Output: 0
Explanation: The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.
```

Example 3:

```
Input: nums = [11,13,15,17]
Output: 11
Explanation: The original array was [11,13,15,17] and it was rotated 4 times.
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5000`
- `5000 <= nums[i] <= 5000`
- All the integers of `nums` are **unique**.
- `nums` is sorted and rotated between `1` and `n` times.

Solution

```
class Solution:
    def findMin(self, nums: List[int]) -> int:
        left, right = 0, len(nums) - 1

        while left + 1 < right:
            if nums[left] < nums[right]:
                return nums[left]

            mid = (left + right) // 2
            if nums[mid] > nums[left]:
                left = mid
            else:
                right = mid

        if nums[left] < nums[right]:
            return nums[left]
        else:
            return nums[right]
```

一文解决 4 道「搜索旋转排序数组」题！ - 寻找旋转排序数组中的最小值 - 力扣（LeetCode）

981. Time Based Key-Value Store

Created	@October 15, 2022 4:43 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/time-based-key-value-store/
Importance	
Tag	Binary search NEET
Video	

Design a time-based key-value data structure that can store multiple values for the same key at different time stamps and retrieve the key's value at a certain timestamp.

Implement the `TimeMap` class:

- `TimeMap()` Initializes the object of the data structure.
- `void set(String key, String value, int timestamp)` Stores the key `key` with the value `value` at the given time `timestamp`.
- `String get(String key, int timestamp)` Returns a value such that `set` was called previously, with `timestamp_prev <= timestamp`. If there are multiple such values, it returns the value associated with the largest `timestamp_prev`. If there are no values, it returns `""`.

Example 1:

```
Input
["TimeMap", "set", "get", "get", "set", "get", "get"]
[[], ["foo", "bar", 1], ["foo", 1], ["foo", 3], ["foo", "bar2", 4], ["foo", 4], ["foo",
 5]]
Output
[null, null, "bar", "bar", null, "bar2", "bar2"]

Explanation
TimeMap timeMap = new TimeMap();
timeMap.set("foo", "bar", 1);  // store the key "foo" and value "bar" along with timestamp
= 1.
timeMap.get("foo", 1);         // return "bar"
timeMap.get("foo", 3);         // return "bar", since there is no value corresponding to f
```

```
oo at timestamp 3 and timestamp 2, then the only value is at timestamp 1 is "bar".
timeMap.set("foo", "bar2", 4); // store the key "foo" and value "bar2" along with timestam
p = 4.
timeMap.get("foo", 4);         // return "bar2"
timeMap.get("foo", 5);         // return "bar2"
```

Constraints:

- `1 <= key.length, value.length <= 100`
- `key` and `value` consist of lowercase English letters and digits.
- `1 <= timestamp <= 10 7`
- All the timestamps `timestamp` of `set` are strictly increasing.
- At most `2 * 105` calls will be made to `set` and `get`

Solution

```
class TimeMap:

    def __init__(self):
        self.keyStore = {} # key: list of [val, timestamp]

    def set(self, key: str, value: str, timestamp: int) -> None:
        if key not in self.keyStore:
            self.keyStore[key] = []
        self.keyStore[key].append([value, timestamp])

    def get(self, key: str, timestamp: int) -> str:
        values = self.keyStore.get(key, [])
        if not values:
            return ''

        left, right = 0, len(values) - 1

        while left + 1 < right:
            mid = (left + right) // 2
            if values[mid][1] < timestamp:
                left = mid
            elif values[mid][1] > timestamp:
                right = mid
            else:
                return values[mid][1]

        if values[right][1] <= timestamp:
            return values[right][0]
        if values[left][1] <= timestamp:
            return values[left][0]

        return ''


# Your TimeMap object will be instantiated and called as such:
# obj = TimeMap()
# obj.set(key,value,timestamp)
# param_2 = obj.get(key,timestamp)
```

```
class TimeMap:
    def __init__(self):
        """
        Initialize your data structure here.
        """
        self.keyStore = {}  # key : list of [val, timestamp]

    def set(self, key: str, value: str, timestamp: int) -> None:
        if key not in self.keyStore:
            self.keyStore[key] = []
        self.keyStore[key].append([value, timestamp])

    def get(self, key: str, timestamp: int) -> str:
        res, values = "", self.keyStore.get(key, [])
        l, r = 0, len(values) - 1
        while l <= r:
            m = (l + r) // 2
            if values[m][1] <= timestamp:
                res = values[m][0]
                l = m + 1
            else:
                r = m - 1
        return res
```