App Academy

Binary Search

Binary Search

- Binary search is an extremely efficient searching algorithm.
- It is one of the few algorithms that can reach O(log(n)) runtime.
- The prerequisite for binary search is usually to have sorted array.
    - However there are some exceptions where you can binary search on things that aren’t simply sorted arrays.

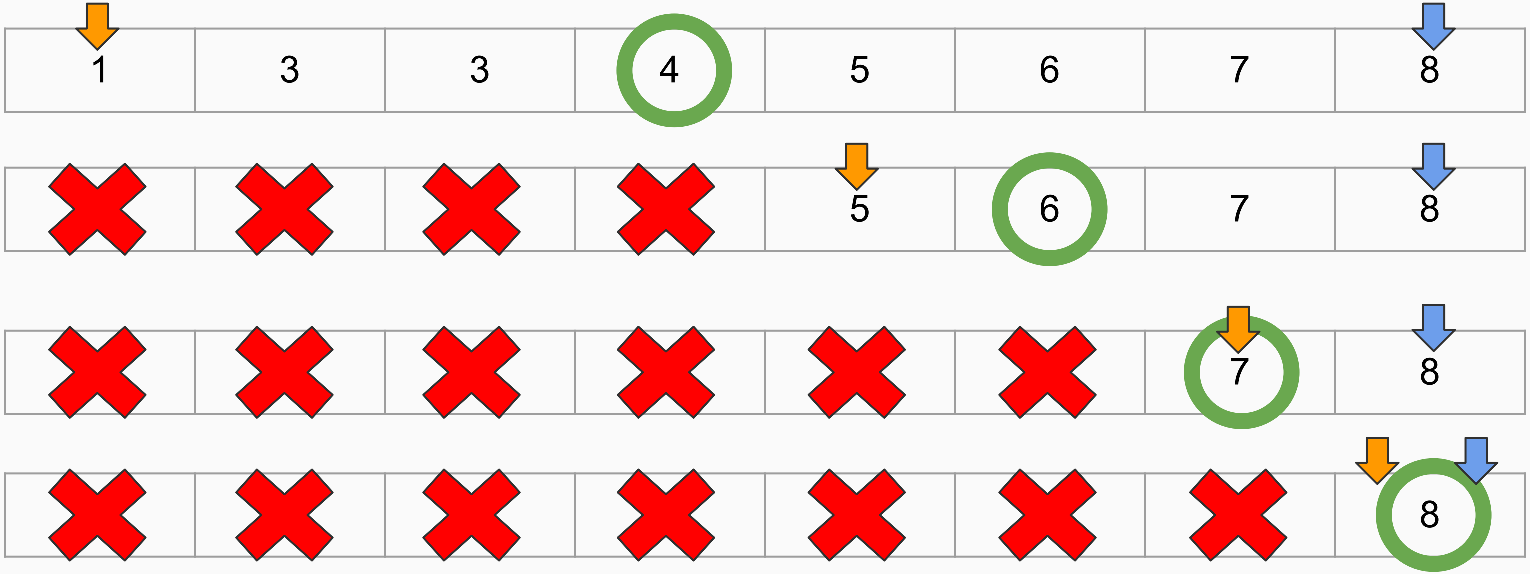
Binary Search

TARGET = 8

1	3	3	4	5	6	7	8

Binary Search example 1

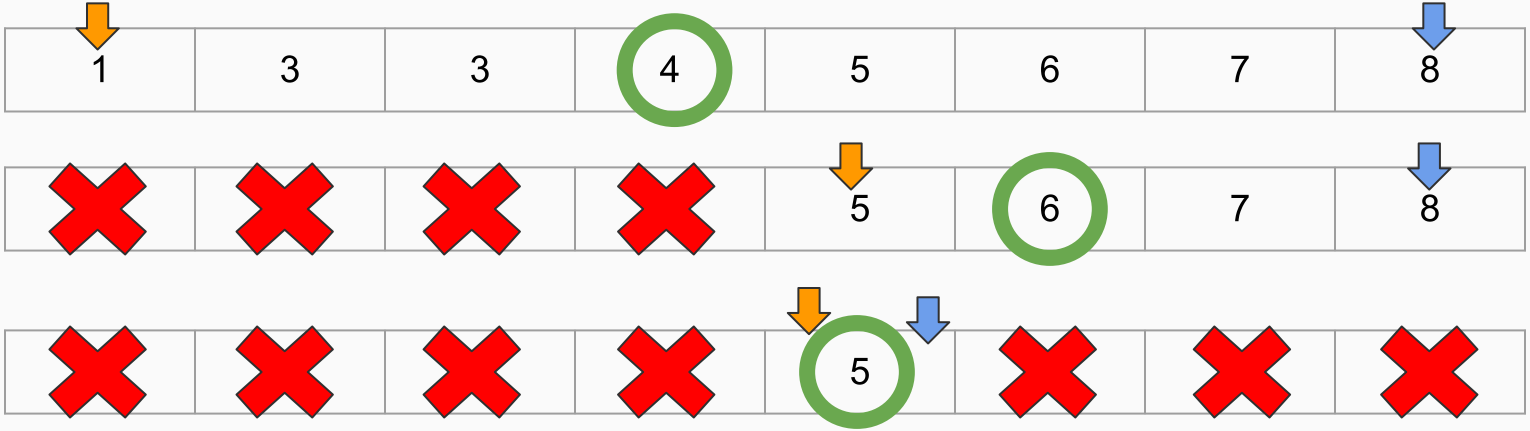
TARGET = 8



1	3	3	4	5	6	7	8
✗	✗	✗	✗	5	6	7	8
✗	✗	✗	✗	✗	✗	7	8
✗	✗	✗	✗	✗	✗	✗	8

Binary Search example 2

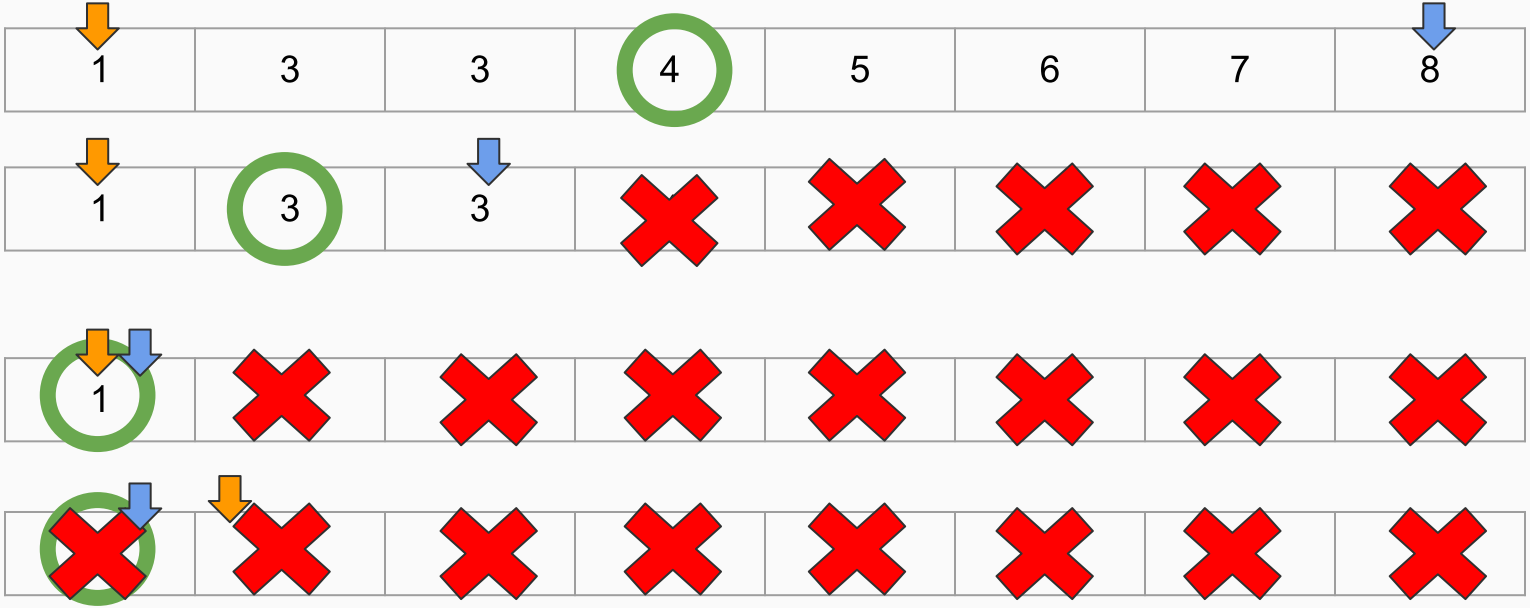
TARGET = 5



1	3	3	4	5	6	7	8
✗	✗	✗	✗	5	6	7	8
✗	✗	✗	✗	5	✗	✗	✗

Binary Search example 3

TARGET = 2



Demo

Binary Search

Recursive Binary Search

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        def recurse(left, right):
            if left >= right:
                return right if nums[right] == target else -1
            pivot = left + (right - left)//2

            if nums[pivot] == target:
                return pivot
            elif target < nums[pivot]:
                return recurse(left, pivot - 1)
            else:
                return recurse(pivot + 1, right)

        return recurse(0, len(nums)-1)
```

Iterative Binary Search

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

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

Searching Ranges

Demo: Guess Number Higher or Lower

Questions?

Let’s practice!

- Review
  - Search a 2D matrix
  - Find Peak Element
- Bonus
  - Koko Eating Bananas
  - Search in Rotated Sorted Array