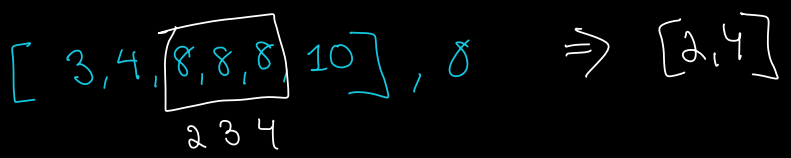
Problem: Given sorted (asc.) int array, return start + end index of given target [x, y]. Solution should be O(logn) time.

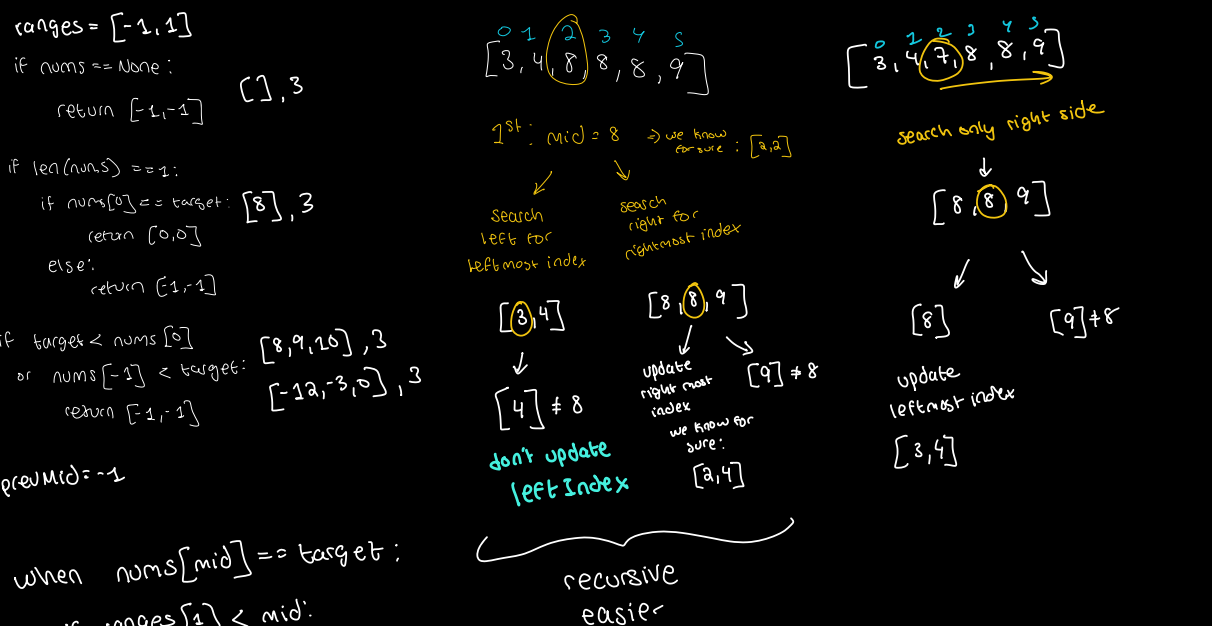


Intro:

- Verify Constraints
  -
- Create Testcases
  - [3, 4, 8, 8, 8, 9], 8 -> [2, 4]
  - [3, 4, 8, 8, 8], 6 -> [-1, -1]
  - [1] 1 -> [0,0]
  - [] 8 -> [-1, -1]

Optimal 1:

- Brainstorming & Pattern Observations



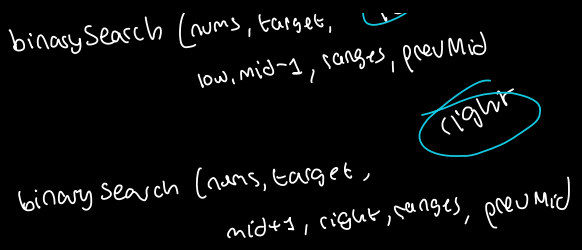

```
when nums[mid] == target:
    if ranges[1] < mid:
        ranges[1] = mid
    if mid < ranges[0]:
        ranges[0] = mid
    if prevMid != -1:
        if prevMid < mid:
            bs(nums, trgt, mid+1, right, ranges, mid)   # right
        else
            bs(nums, trgt, left, mid-1, ranges, mid)    # left
else:
```

left

⇒ regular binarySearch until find target

⇒ when find target

- left search for leftmost index
- right search for rightmost index



- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity
  - Time: O(logn)
  - Space: (recursive): O(logn)

Iterative Optimal:
- Time: O(logn)
- Space: O(1)