

JAVA PABL PRACTICE SHEET – 3

Given a sorted array of distinct integers 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 must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: nums = [1,3,5,6], target = 5

Output: 2

Example 2:

Input: nums = [1,3,5,6], target = 2

Output: 1

Example 3:

Input: nums = [1,3,5,6], target = 7

Output: 4

1.

SOLUTION-

The screenshot displays a LeetCode submission for the "Search Insert Position" problem. The submission is accepted, with a runtime of 0ms and memory usage of 44.66 MB. The code is written in Java and implements a binary search algorithm. The test result shows three cases passed.

Code:

```
1 class Solution {
2     public int searchInsert(int[] nums, int target) {
3         int low = 0, high = nums.length - 1;
4
5         while (low <= high) {
6             int mid = low + (high - low) / 2;
7
8             if (nums[mid] == target) {
```

Testcase:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums = [1,3,5,6]

target = 5

Output

2

Expected

Given an array of **distinct** integers candidates and a target integer target, return a *list of all **unique combinations** of candidates where the chosen numbers sum to target*. You may return the combinations in **any order**.

The **same** number may be chosen from candidates an **unlimited number of times**.

Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

Example 1:

Input: candidates = [2,3,6,7], target = 7

Output: [[2,2,3],[7]]

Explanation:

2 and 3 are candidates, and $2 + 2 + 3 = 7$. Note that 2 can be used multiple times.

7 is a candidate, and $7 = 7$.

These are the only two combinations.

Example 2:

Input: candidates = [2,3,5], target = 8

Output: [[2,2,2,2],[2,3,3],[3,5]]

Example 3:

Input: candidates = [2], target = 1

2. **Output:** []

Solution -

The screenshot displays the LeetCode interface for the 'Combination Sum' problem. The top section shows the problem description and the solution's performance metrics: 3 ms runtime (beating 36.96%) and 45.84 MB memory (beating 51.48%). A bar chart illustrates the runtime distribution across different time intervals. The code editor shows a Java solution using a recursive approach with a List to store the current combination. The test results section shows that the solution passed all 160 test cases, with a runtime of 0 ms for the specific test case shown.

```
class Solution {
    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> ans = new ArrayList<>();
        solve(0, candidates, target, new ArrayList<>(), ans);
        return ans;
    }

    void solve(int i, int[] candidates, int target, List<Integer> temp, List<List<Integer>> ans) {
        if (target < 0) return;
        if (target == 0) {
            ans.add(new ArrayList<>(temp));
            return;
        }
        for (int j = i; j < candidates.length; j++) {
            temp.add(candidates[j]);
            solve(j, candidates, target - candidates[j], temp, ans);
            temp.remove(temp.size() - 1);
        }
    }
}
```

Testcase 1: Runtime: 0 ms

Case 1 Case 2 Case 3

Input

candidates = [2,3,6,7]

target = 7

Output

[[2,2,3],[7]]

Expected

[[2,2,3],[7]]

3.

Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target.

Each number in candidates may only be used **once** in the combination.

Note: The solution set must not contain duplicate combinations.

Example 1:

Input: candidates = [10,1,2,7,6,1,5], target = 8

Output:

```
[
  [1,1,6],
  [1,2,5],
  [1,7],
  [2,6]
]
```

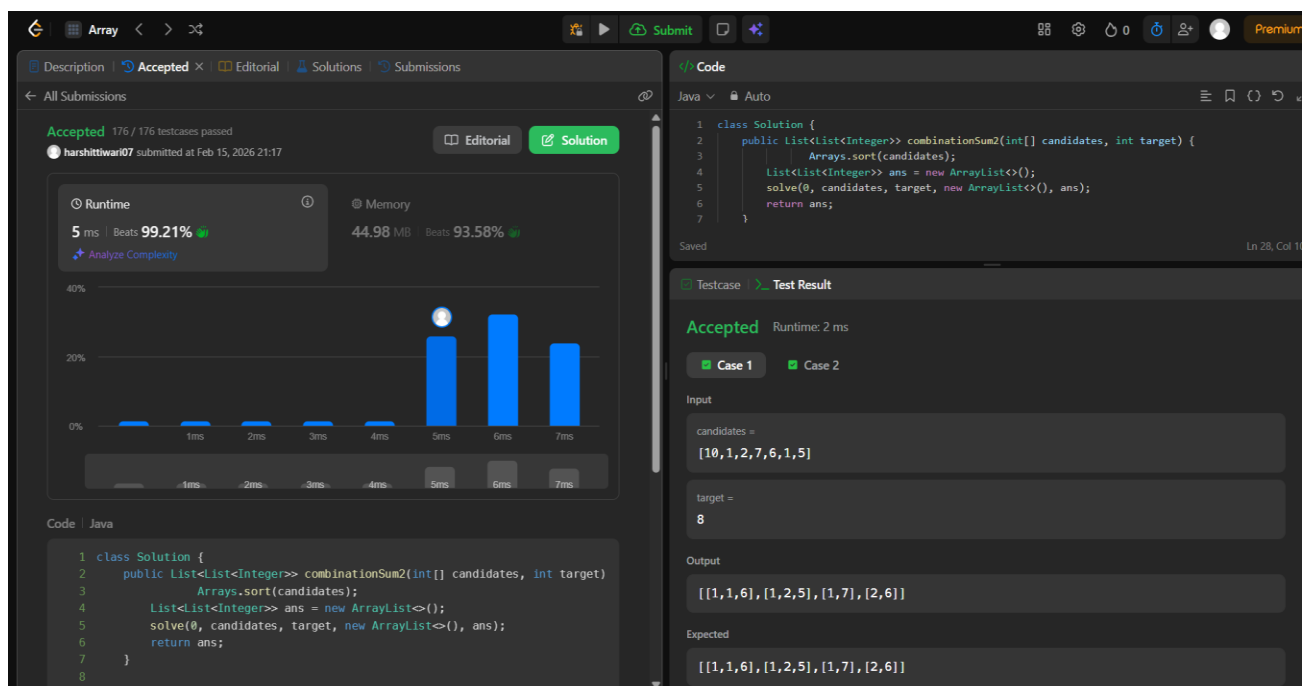
Example 2:

Input: candidates = [2,5,2,1,2], target = 5

Output:

```
[
  [1,2,2],
  [5]
]
```

Solution –



You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at index 0.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at index `i`, you can jump to any index `(i + j)` where:

- $0 \leq j \leq \text{nums}[i]$ and
- $i + j < n$

Return *the minimum number of jumps to reach index `n - 1`*. The test cases are generated such that you can reach index `n - 1`.

Example 1:

Input: `nums = [2,3,1,1,4]`

Output: 2

Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2:

Input: `nums = [2,3,0,1,4]`

Output: 2

4.

Solution-

The screenshot displays the LeetCode interface for the 'Jump Game' problem. The top section shows the problem description and constraints. Below this, the 'Accepted' status is confirmed with 110/110 test cases passed. The submission details indicate it was submitted by 'harshittiwari07' on Feb 15, 2026, at 21:22. The runtime performance is shown as 1 ms, beating 99.54% of solutions, with a memory usage of 47.39 MB, beating 55.60%.

The code editor shows the following Java solution:

```
1 class Solution {
2     public int jump(int[] nums) {
3         int jumps = 0;
4         int far = 0;
5         int end = 0;
6
7         for (int i = 0; i < nums.length - 1; i++) {
8             far = Math.max(far, i + nums[i]);
```

The test results section shows the solution is 'Accepted' with a runtime of 0 ms. The input is `nums = [2,3,1,1,4]` and the output is `2`, which matches the expected result.

Given an array of strings `strs`, group the anagrams together. You can return the answer in **any order**.

Example 1:

Input: `strs = ["eat","tea","tan","ate","nat","bat"]`

Output: `[["bat"],["nat","tan"],["ate","eat","tea"]]`

Explanation:

- There is no string in `strs` that can be rearranged to form "bat".
- The strings "nat" and "tan" are anagrams as they can be rearranged to form each other.
- The strings "ate", "eat", and "tea" are anagrams as they can be rearranged to form each other.

Example 2:

Input: `strs = [""]`

Output: `[[""]]`

Example 3:

Input: `strs = ["a"]`

Output: `[["a"]]`

5. Link: <https://leetcode.com/problems/group-anagrams/description/?envType=problem-list-v2&envId=array>

Solution-

The screenshot displays the LeetCode submission interface for the "Group Anagrams" problem. The submission is marked as "Accepted" with 128/128 test cases passed. The user's performance is 6 ms (99.35% beats) and 49.62 MB (74.84% beats). The code is written in Java, using a `HashMap` to group anagrams by their sorted character arrays. The test case shows the input `strs = ["eat","tea","tan","ate","nat","bat"]` and the expected output `[["bat"],["nat","tan"],["ate","eat","tea"]]`.

```
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        HashMap<String, List<String>> map = new HashMap<>();

        for (String word : strs) {
            char[] ch = word.toCharArray();
            Arrays.sort(ch);
```

You are given a **large integer** represented as an integer array `digits`, where each `digits[i]` is the i^{th} digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return *the resulting array of digits*.

Example 1:

Input: `digits = [1,2,3]`

Output: `[1,2,4]`

Explanation: The array represents the integer 123.

Incrementing by one gives $123 + 1 = 124$.

Thus, the result should be `[1,2,4]`.

Example 2:

Input: `digits = [4,3,2,1]`

Output: `[4,3,2,2]`

Explanation: The array represents the integer 4321.

Incrementing by one gives $4321 + 1 = 4322$.

Thus, the result should be `[4,3,2,2]`.

Example 3:

Input: `digits = [9]`

Output: `[1,0]`

Explanation: The array represents the integer 9.

Incrementing by one gives $9 + 1 = 10$.

Thus, the result should be `[1,0]`.

6.

Solution-

The screenshot displays a LeetCode submission for the 'Plus One' problem. The problem description is at the top, followed by the Java solution code. The solution is accepted, with a runtime of 0 ms and memory usage of 43.39 MB. The test results show three cases, all passed.

Code:

```
class Solution {
    public int[] plusOne(int[] digits) {
        for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] < 9) {
                digits[i]++;
                return digits;
            }
        }
        // If all digits are 9, we need to add a new digit at the beginning
        int[] newDigits = new int[digits.length + 1];
        newDigits[0] = 1;
        for (int i = 1; i <= digits.length; i++) {
            newDigits[i] = 0;
        }
        return newDigits;
    }
}
```

Test Results:

- Case 1: `digits = [1,2,3]` → `[1,2,4]` (Accepted)
- Case 2: `digits = [4,3,2,1]` → `[4,3,2,2]` (Accepted)
- Case 3: `digits = [9]` → `[1,0]` (Accepted)

You must do it in place.

1	1	1
1	0	1
1	1	1

Output: $[[1,0,1],[0,0,0],[1,0,1]]$

0	1	2	0
3	4	5	2
1	3	1	5

Output: `[[0,0,0,0],[0,4,5,0],[0,3,1,0]]`

Take
Hom
e

7.

Solution-

Array < > ↻
Submit
Premium

Description | Accepted x | Editorial | Solutions | Submissions

← All Submissions

Accepted 202 / 202 testcases passed

harshittiwari07 submitted at Feb 15, 2026 21:34

Runtime
1 ms Beats **60.26%**

Memory
47.35 MB Beats **70.32%**

Time Interval	Percentage
1ms	~45%
2ms	~10%
3ms	~2%
4ms	~2%

Code

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
matrix =
[[1,1,1],[1,0,1],[1,1,1]]
```

Output

```
[[1,0,1],[0,0,0],[1,0,1]]
```

Expected

```
[[1,0,1],[0,0,0],[1,0,1]]
```

Contribute a testcase

Code | Java

```

1 class Solution {
2     public void setZeros(int[][] matrix) {
3         int m = matrix.length;
4         int n = matrix[0].length;
5
6         boolean firstRowZero = false;
7         boolean firstColZero = false;
8     }
        
```

8.

You are given an $m \times n$ integer matrix `matrix` with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer `target`, return `true` if `target` is in `matrix` or `false` otherwise.

You must write a solution in $O(\log(m * n))$ time complexity.

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`

Accepted 133 / 133 testcases passed

harshittiwari07 submitted at Feb 15, 2026 21:39

Runtime

0 ms | Beats 100.00%

Memory

43.94 MB | Beats 61.79%

150%

100%

50%

0%

1ms

2ms

3ms

4ms

Code | Java

```

1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int m = matrix.length;
4         int n = matrix[0].length;
5
6         int low = 0;
7         int high = (m*n) - 1;
8

```

Code

```

1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int m = matrix.length;
4         int n = matrix[0].length;
5
6         int low = 0;
7         int high = (m*n) - 1;
8
9         while (low <= high){
10             int mid = low + (high - low)/2;
11
12             int row = mid / n;
13             int col = mid % n;
14

```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

matrix =

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

target =

3

Output

true

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

Input: `nums = [2,0,2,1,1,0]`

Output: `[0,0,1,1,2,2]`

Example 2:

Input: `nums = [2,0,1]`

Output: `[0,1,2]`

9.

The screenshot shows a LeetCode submission interface for the 'Sort Colors' problem. The submission is for the 'Array' category and is marked as 'Accepted' with 89/89 testcases passed. The user 'harshittiwari07' submitted it on Feb 15, 2026 at 21:42. The runtime is 0 ms, beating 100.00% of solutions, and the memory usage is 43.70 MB, beating 41.66%. A bar chart shows the runtime distribution, with the majority of submissions falling between 0 and 1 ms. The code is written in Java and implements a sorting algorithm. The test result shows that the input `nums = [2,0,2,1,1,0]` is correctly sorted to `[0,0,1,1,2,2]`.

Runtime: 0 ms | Beats 100.00% | Memory: 43.70 MB | Beats 41.66%

Code:

```
1 class Solution {
2     public void sortColors(int[] nums) {
3         int low = 0, mid = 0, high = nums.length - 1;
4
5         while (mid <= high) {
6             if (nums[mid] == 0) {
7                 int temp = nums[low];
8                 nums[low] = nums[mid];
9                 nums[mid] = temp;
10                low++;
11                mid++;
12            }
13        }
14    }
15 }
```

Test Result: Accepted Runtime: 0 ms

Case 1: Input: `nums = [2,0,2,1,1,0]` Output: `[0,0,1,1,2,2]` Expected: `[0,0,1,1,2,2]`

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

Input: `nums = [1,2,3]`

Output: `[[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]`

Example 2:

Input: `nums = [0]`

Output: `[[],[0]]`

10.

Solution-

The screenshot displays a LeetCode submission for the 'Subsets' problem. The submission is 'Accepted' and was made by user 'harshitiwari07' on Feb 15, 2026 at 21:45. The performance metrics show a runtime of 1 ms (beating 87.84% of solutions) and a memory usage of 44.16 MB (beating 75.65% of solutions). A bar chart visualizes the runtime performance. The Java code is as follows:

```
1 class Solution {
2     public List<List<Integer>> subsets(int[] nums) {
3         List<List<Integer>> ans = new ArrayList<>();
4         solve(0, nums, new ArrayList<>(), ans);
5         return ans;
6     }
7
8     void solve(int index, int[] nums, List<Integer> temp, List<List<Integer>> ans) {
9
10        ans.add(new ArrayList<>(temp));
11
12        for (int i = index; i < nums.length; i++) {
13            temp.add(nums[i]);
14            solve(i + 1, nums, temp, ans);
15            temp.remove(temp.size() - 1);
16        }
17    }
18 }
```

The test result section shows the input `nums = [1, 2, 3]` and the output `[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]`. The expected output is also shown as `[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]`.

Given an $m \times n$ grid of characters `board` and a string `word`, return `true` if `word` exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

Example 1:

A	B	C	E
S	F	C	S
A	D	E	E

Input: `board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, `word = "ABCCED"`

Output: `true`

Example 2:

11.

A	B	C	E
S	F	C	S
A	D	E	E

Input: `board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, `word = "SEE"`

Output: `true`

Example 3:

A	B	C	E
S	F	C	S
A	D	E	E

Input: `board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, `word = "ABCB"`

Output: `false`

The screenshot displays a LeetCode submission for the 'Word Search' problem. The problem description is at the top, followed by the Java code solution. The code uses a recursive DFS function to explore the grid from each cell, checking for the word. The submission is marked as 'Accepted' with a runtime of 129 ms and memory usage of 43.22 MB. The test results section shows three test cases, all of which are passed.

```

class Solution {
    public boolean exist(char[][] board, String word) {
        int m = board.length;
        int n = board[0].length;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (dfs(board, word, i, j, 0)) {
                    return true;
                }
            }
        }
        return false;
    }
    private boolean dfs(char[][] board, String word, int i, int j, int index) {
        if (index == word.length()) return true;
        if (i < 0 || i > board.length - 1 || j < 0 || j > board[0].length - 1 || board[i][j] != word.charAt(index)) return false;
        board[i][j] = '\0';
        boolean found = dfs(board, word, i + 1, j, index + 1) ||
            dfs(board, word, i - 1, j, index + 1) ||
            dfs(board, word, i, j + 1, index + 1) ||
            dfs(board, word, i, j - 1, index + 1);
        board[i][j] = word.charAt(index);
        return found;
    }
}

```

Given an array `nums` of n integers, return *an array of all the unique quadruplets* `[nums[a], nums[b], nums[c], nums[d]]` such that:

- $0 \leq a, b, c, d < n$
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

12. **Input:** `nums = [1,0,-1,0,-2,2]`, `target = 0`

Output: `[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]`

Example 2:

Input: `nums = [2,2,2,2,2]`, `target = 8`

Output: `[[2,2,2,2]]`

Solution-

The screenshot displays a LeetCode submission for the "4Sum" problem. The submission is accepted, with a runtime of 129ms and memory usage of 43.22 MB. The code is written in Java and uses a DFS approach to find unique quadruplets. The test result shows three cases passed.

Runtime: 129 ms, Beats 81.51%
Memory: 43.22 MB, Beats 30.01%

Code (Java):

```
1 class Solution {
2     public boolean exist(char[][] board, String word) {
3         int m = board.length;
4         int n = board[0].length;
5
6         for (int i = 0; i < m; i++) {
7             for (int j = 0; j < n; j++) {
8                 if (dfs(board, word, i, j, 0)) {
```

Testcase: Accepted, Runtime: 0 ms

Case 1: board = `[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]`, word = `"ABCCED"`, Output: `true`

There is an integer array `nums` sorted in ascending order (with **distinct** values). Prior to being passed to your function, `nums` is **possibly left rotated** at an unknown index `k` ($1 \leq k < \text{nums.length}$) 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 left rotated by 3 indices 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

13.

The screenshot displays the LeetCode submission interface for the problem "Search in a Sorted Array". The top section shows the problem description and the submission status: "Accepted" with 196/196 testcases passed. The submission was made by "harshittiwari07" on Feb 15, 2026, at 22:03. The performance metrics are 0 ms runtime (beats 100.00%) and 43.91 MB memory (beats 44.98%). A bar chart shows the runtime performance relative to other submissions. The code is written in Java and implements a binary search algorithm. The test case input is `nums = [4,5,6,7,0,1,2]` and `target = 0`, with the output being 4.

```
class Solution {
    public int search(int[] nums, int target) {
        int low = 0, high = nums.length - 1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (nums[mid] == target) return mid;

            if (nums[low] <= nums[mid]) {
                if (target >= nums[low] && target < nums[mid]) {
                    high = mid - 1;
                } else {
                    low = mid + 1;
                }
            } else {
                low = mid + 1;
            }
        }

        return -1;
    }
}
```

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

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

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

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]`

Example 3:

Input: `nums = []`, `target = 0`

Output: `[-1,-1]`

14.

The screenshot shows a LeetCode submission for the problem "Find First and Last Position of Element in Sorted Array". The submission is accepted, with a runtime of 0ms and memory usage of 48.58 MB. The code is in Java and implements a binary search for the first and last positions of the target.

Runtime: 0 ms, Beats 100.00%
Memory: 48.58 MB, Beats 11.72%

Code:

```
1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int first = findFirst(nums, target);
4         int last = findLast(nums, target);
5         return new int[]{first, last};
6     }
7
8     int findFirst(int[] nums, int target) {
9         int low = 0, high = nums.length - 1;
10        int ans = -1;
11        while (low <= high) {
12            int mid = (low + high) / 2;
13            if (nums[mid] < target) {
14                low = mid + 1;
15            } else if (nums[mid] > target) {
16                high = mid - 1;
17            } else {
18                ans = mid;
19                low = mid + 1;
20            }
21        }
22        return ans;
23    }
24
25    int findLast(int[] nums, int target) {
26        int low = 0, high = nums.length - 1;
27        int ans = -1;
28        while (low <= high) {
29            int mid = (low + high) / 2;
30            if (nums[mid] < target) {
31                low = mid + 1;
32            } else if (nums[mid] > target) {
33                high = mid - 1;
34            } else {
35                ans = mid;
36                high = mid - 1;
37            }
38        }
39        return ans;
40    }
41}
```

Testcase: Accepted
Runtime: 0 ms
Case 1: [5,7,7,8,8,10], target = 8, Output: [3,4]