

## **Experiment List for Programming Ability and Logic Building-2**

Name : Ikshu Goel

Batch : 2CSE5

Roll Number : 2410030774

Week : 09/02/26 to 14/02/26

Lecture : 1 & 2

# EXPERIMENT 1

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:**

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

### Example 2:

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

### Example 3:

nums = [1,3,5,6], target = 7  
4

### Constraints:

- $1 \leq \text{nums.length} \leq 10^4$
- $-10^4 \leq \text{nums}[i] \leq 10^4$
- nums contains **distinct** values sorted in **ascending** order.
- $-10^4 \leq \text{target} \leq 10^4$

**35. Search Insert Position** Solved

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

**Constraints:**

- $1 \leq \text{nums.length} \leq 10^4$
- $-10^4 \leq \text{nums}[i] \leq 10^4$
- nums contains **distinct** values sorted in **ascending** order.
- $-10^4 \leq \text{target} \leq 10^4$

```
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) / 2;
7
8             if (nums[mid] == target)
9                 return mid;
10            else if (nums[mid] < target)
11                low = mid + 1;
12            else
13                high = mid - 1;
14        }
15        return low;
16    }
17 }
18
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums = [1,3,5,6]

target =

## EXPERIMENT 2

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:

```
candidates = [2,3,6,7], target = 7  
[[2,2,3],[7]]
```

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:

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

### Example 3:

candidates = [2], target = 1  
[]

### Constraints:

- $1 \leq \text{candidates.length} \leq 30$
- $2 \leq \text{candidates}[i] \leq 40$
- All elements of candidates are **distinct**.
- $1 \leq \text{target} \leq 40$

The screenshot shows the LeetCode interface for the problem '39. Combination Sum'. The problem description states: '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  
Output: []

The code editor shows a Java solution using a recursive backtracking approach. The solution class is named 'Solution' and has a method 'combinationSum' that returns a list of lists. It uses a helper method 'solve' to recursively find combinations. The test result shows 'Accepted' with a runtime of 0 ms.

```
1 import java.util.*;
2
3 class Solution {
4     List<List<Integer>> ans = new ArrayList<>();
5
6     public List<List<Integer>> combinationSum(int[] c, int target) {
7         solve(c, target, 0, new ArrayList<>());
8         return ans;
9     }
10
11     void solve(int[] c, int target, int i, List<Integer> list) {
12         if (target == 0) {
13             ans.add(new ArrayList<>(list));
14             return;
15         }
16         if (i == c.length || target < 0) return;
17
18         list.add(c[i]);
19         solve(c, target - c[i], i, list);
20         list.remove(list.size() - 1);
21
22         solve(c, target, i + 1, list);
23     }
24 }
25
```

## EXPERIMENT 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:

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

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

### Example 2:

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

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

## Constraints:

- $1 \leq \text{candidates.length} \leq 100$
- $1 \leq \text{candidates}[i] \leq 50$
- $1 \leq \text{target} \leq 30$

**40. Combination Sum II**

Medium

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]
]
```

```
1 import java.util.*;
2
3 class Solution {
4     List<List<Integer>> ans = new ArrayList<>();
5
6     public List<List<Integer>> combinationSum2(int[] candidates, int target) {
7         Arrays.sort(candidates);
8         solve(candidates, target, 0, new ArrayList<>());
9         return ans;
10    }
11
12    void solve(int[] arr, int target, int idx, List<Integer> list) {
13        if (target == 0) {
14            ans.add(new ArrayList<>(list));
15            return;
16        }
17
18        for (int i = idx; i < arr.length; i++) {
19            if (i > idx && arr[i] == arr[i - 1]) continue;
20            if (arr[i] > target) break;
21
22            list.add(arr[i]);
23            solve(arr, target - arr[i], i + 1, list);
24            list.remove(list.size() - 1);
25        }
26    }
27 }
```

Accepted Runtime: 1 ms

## EXPERIMENT 4

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:

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

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:

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

### Constraints:

- $1 \leq \text{nums.length} \leq 10^4$

- $0 \leq \text{nums}[i] \leq 100$
- It's guaranteed that you can reach  $\text{nums}[n - 1]$ .

The screenshot shows a web browser with several tabs open, including 'Search Insert Position - Leet...', 'Combination Sum - LeetCode', 'Combination Sum II - LeetCode', and 'Jump Game II - LeetCode'. The active tab is 'Jump Game II - LeetCode', displaying the problem description for '45. Jump Game II'. The problem states: '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. Constraints: 

- $1 \leq \text{nums.length} \leq 10^4$

 The code editor on the right shows a Java solution: 

```
1 class Solution {
2     public int jump(int[] nums) {
3         int jumps = 0, end = 0, far = 0;
4
5         for (int i = 0; i < nums.length - 1; i++) {
6             far = Math.max(far, i + nums[i]);
7
8             if (i == end) {
9                 jumps++;
10                end = far;
11            }
12        }
13        return jumps;
14    }
15 }
```

 The test result shows 'Accepted' with a runtime of 0 ms. The input for the test case is `nums = [2,3,1,1,4]`.



# EXPERIMENT 5

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"]]`

## Constraints:

- $1 \leq \text{strs.length} \leq 10^4$
- $0 \leq \text{strs}[i].\text{length} \leq 100$
- `strs[i]` consists of lowercase English letters.

Search Insert Position - Leet... x Combination Sum - LeetCode x Combination Sum II - LeetCo... x Jump Game II - LeetCode x Group Anagrams - LeetCode x

leetcode.com/problems/group-anagrams/?envType=problem

To exit full screen, press and hold **esc**

Array < > ↺

Submit

0 Premium

### 49. Group Anagrams

Medium Topics Companies

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"]]`

21.8K 382 482 Online

#### Code

Java Auto

```
1 import java.util.*;
2
3 class Solution {
4     public List<List<String>> groupAnagrams(String[] strs) {
5         Map<String, List<String>> map = new HashMap<>();
6
7         for (String s : strs) {
8             char[] ch = s.toCharArray();
9             Arrays.sort(ch);
10            String key = new String(ch);
11
12            map.putIfAbsent(key, new ArrayList<>());
13            map.get(key).add(s);
14        }
15        return new ArrayList<>(map.values());
16    }
17 }
18
```

Saved Ln 13, Col 3

#### Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

`strs =`

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

Output

## EXPERIMENT 6

You are given a **large integer** represented as an integer array `digits`, where each `digits[i]` is the  $i^{\text{th}}$  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:

`digits = [1,2,3]`

`[1,2,4]`

The array represents the integer 123.

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

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

#### Example 2:

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

`[4,3,2,2]`

The array represents the integer 4321.

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

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

#### Example 3:

`digits = [9]`

`[1,0]`

The array represents the integer 9.

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

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

#### Constraints:

- $1 \leq \text{digits.length} \leq 100$
- $0 \leq \text{digits}[i] \leq 9$
- `digits` does not contain any leading 0's.

66. Plus One

Easy Topics Companies

You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the  $i^{\text{th}}$  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]`.

```
public int[] plusOne(int[] d) {
    for (int i = d.length - 1; i >= 0; i--) {
        if (d[i] < 9) {
            d[i]++;
            return d;
        }
        d[i] = 0;
    }
    int[] res = new int[d.length + 1];
    res[0] = 1;
    return res;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

digits =

[1,2,3]

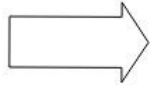
## EXPERIMENT 7

Given an  $m \times n$  integer matrix `matrix`, if an element is 0, set its entire row and column to 0's.

You must do it **in place**.

### Example 1:

1	1	1
1	0	1
1	1	1



1	0	1
0	0	0
1	0	1

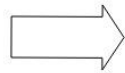
matrix =

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

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

### Example 2:

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



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

matrix =

[[0,1,2,0],[3,4,5,2],[1,3,1,5]]

[[0,0,0,0],[0,4,5,0],[0,3,1,0]]

### Constraints:

- $m == \text{matrix.length}$
- $n == \text{matrix}[0].\text{length}$
- $1 \leq m, n \leq 200$
- $-2^{31} \leq \text{matrix}[i][j] \leq 2^{31} - 1$

Search Insert Position x Combination Sum x Combination Sum x Jump Game II - Le x Group Anagrams x Plus One - LeetC x Set Matrix Zeroes x

leetcode.com/problems/set-matrix-zeroes/?envType=problem

To exit full screen, press and hold **esc**

Array < > ↺

Description Editorial Solutions Submissions

### 73. Set Matrix Zeroes

Medium Topics Companies Hint

Given an  $m \times n$  integer matrix `matrix`, if an element is 0, set its entire row and column to 0's.

You must do it in place.

**Example 1:**

1	1	1		1	0	1
1	0	1	→	0	0	0
1	1	1		1	0	1

**Input:** `matrix = [[1,1,1],[1,0,1],[1,1,1]]`  
**Output:** `[[1,0,1],[0,0,0],[1,0,1]]`

**Example 2:**

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

**Code**

```

Java Auto
4     boolean[] col = new boolean[m[0].length];
5
6     for (int i = 0; i < m.length; i++)
7         for (int j = 0; j < m[0].length; j++)
8             if (m[i][j] == 0) {
9                 row[i] = true;
10                col[j] = true;
11            }
12
13     for (int i = 0; i < m.length; i++)
14         for (int j = 0; j < m[0].length; j++)
15             if (row[i] || col[j])
16                 m[i][j] = 0;
17
18 }
19

```

Saved Ln 19, Col 1

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

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

17K 321 280 Online

## EXPERIMENT 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

matrix =

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

**Example 2:**

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

matrix =

[[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13  
false

## Constraints:

- $m == \text{matrix.length}$
- $n == \text{matrix}[i].\text{length}$
- $1 \leq m, n \leq 100$
- $-10^4 \leq \text{matrix}[i][j], \text{target} \leq 10^4$

The screenshot shows the LeetCode interface for the problem "74. Search a 2D Matrix". The problem description states that the matrix has two properties: each row is sorted in non-decreasing order, and the first integer of each row is greater than the last integer of the previous row. The goal is to return true if the target is in the matrix or false otherwise, with a time complexity of  $O(\log(m * n))$ .

**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:**

The Java code implements a binary search algorithm. It first determines the number of rows (n) and columns (m). Then, it sets a low pointer to 0 and a high pointer to n \* m - 1. A while loop continues as long as low is less than or equal to high. Inside the loop, it calculates the mid index, retrieves the value at that index in the matrix, and compares it to the target. If the value equals the target, it returns true. If the value is less than the target, it updates low to mid + 1. If the value is greater than the target, it updates high to mid - 1. If the loop ends without finding the target, it returns false.

**Test Result:** Accepted. Runtime: 0 ms. Case 1 and Case 2 are both passed.

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



# EXPERIMENT 9

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:

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

## Example 2:

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

## Constraints:

- `n == nums.length`
- `1 <= n <= 300`
- `nums[i]` is either 0, 1, or 2.

**75. Sort Colors**

Medium Topics Companies Hint

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]`

**Constraints:**

- `n == nums.length`
- `1 <= n <= 300`
- `nums[i]` is either `0`, `1`, or `2`.

```
1 class Solution {
2     public void sortColors(int[] nums) {
3         int c0 = 0, c1 = 0, c2 = 0;
4
5         for (int x : nums) {
6             if (x == 0) c0++;
7             else if (x == 1) c1++;
8             else c2++;
9         }
10
11         int i = 0;
12         while (c0-- > 0) nums[i++] = 0;
13         while (c1-- > 0) nums[i++] = 1;
14         while (c2-- > 0) nums[i++] = 2;
15     }
16 }
17
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

[2,0,2,1,1,0]

## EXPERIMENT 10

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:**

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

**Example 2:**

```
nums = [0]
[[],[0]]
```

**Constraints:**

- $1 \leq \text{nums.length} \leq 10$
- $-10 \leq \text{nums}[i] \leq 10$
- All the numbers of `nums` are **unique**.

The screenshot shows the LeetCode interface for the '78. Subsets' problem. The problem description states: '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]]`

**Constraints:**

- $1 \leq \text{nums.length} \leq 10$
- $-10 \leq \text{nums}[i] \leq 10$
- All the numbers of `nums` are **unique**.

The code editor shows a Java solution:

```
1 import java.util.*;
2
3 class Solution {
4     List<List<Integer>> ans = new ArrayList<>();
5
6     public List<List<Integer>> subsets(int[] nums) {
7         solve(nums, 0, new ArrayList<>());
8         return ans;
9     }
10
11     void solve(int[] nums, int i, List<Integer> list) {
12         if (i == nums.length) {
13             ans.add(new ArrayList<>(list));
14             return;
15         }
16
17         list.add(nums[i]);
18         solve(nums, i + 1, list);
19
20         list.remove(list.size() - 1);
21         solve(nums, i + 1, list);
22     }
23 }
```

The bottom of the screenshot shows the 'Testcase' tab with 'Accepted' status, runtime of 0 ms, and two cases passed.

## EXPERIMENT 11

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

board =  
 [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"  
 true

#### Example 2:

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

board =  
 [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"  
 true

#### Example 3:

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

board =

[[ "A", "B", "C", "E"], [ "S", "F", "C", "S"], [ "A", "D", "E", "E"]], word = "ABCB"  
false

#### Constraints:

- $m == \text{board.length}$
- $n = \text{board}[i].\text{length}$
- $1 \leq m, n \leq 6$
- $1 \leq \text{word.length} \leq 15$
- board and word consists of only lowercase and uppercase English letters.

Search x Combin x Combin x Jump C x Group x Plus O x Set Ma x Search x Sort C x Subse x Word S x +

leetcode.com/problems/word-search/?envType=problem-list-v2&envId=array

Array < > ↕

Description Editorial Solutions Submissions

## 79. Word Search

Medium Topics Companies

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:**

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

17.5K 332 292 Online

Code

```

Java Auto
10 boolean dfs(char[][] b, String w, int i, int j, int k) {
11     if (k == w.length()) return true;
12     if (i < 0 || j < 0 || i >= b.length || j >= b[0].length || b[i][j] != w.charAt(k))
13         return false;
14     char temp = b[i][j];
15     b[i][j] = '#';
16
17     boolean found = dfs(b, w, i + 1, j, k + 1) ||
18                   dfs(b, w, i - 1, j, k + 1) ||
19                   dfs(b, w, i, j + 1, k + 1) ||
20                   dfs(b, w, i, j - 1, k + 1);
21
22     b[i][j] = temp;
23     return found;
24 }
25

```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

board =

```

[["A","B","C","E"], ["S","F","C","S"], ["A","D","E","E"]]

```

word =

```

"ABCCED"

```

## EXPERIMENT 12

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**.
- $\text{nums}[a] + \text{nums}[b] + \text{nums}[c] + \text{nums}[d] == \text{target}$

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

#### Example 1:

```
nums = [1,0,-1,0,-2,2], target = 0  
[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

#### Example 2:

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

#### Constraints:

- $1 \leq \text{nums.length} \leq 200$
- $-10^9 \leq \text{nums}[i] \leq 10^9$
- $-10^9 \leq \text{target} \leq 10^9$



leetcode.com/problems/4sum?envType=problem-list-v2&envId=array

### 18. 4Sum

Medium

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 <= 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:**

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]]`

**Constraints:**

- `1 <= nums.length <= 200`
- `-109 <= nums[i] <= 109`
- `-108 <= target <= 108`

```
1 import java.util.*;
2 class Solution {
3     public List<List<Integer>> fourSum(int[] nums, int target) {
4         List<List<Integer>> result = new ArrayList<>();
5         int n = nums.length;
6         Arrays.sort(nums);
7         for (int i = 0; i < n - 3; i++) {
8             if (i > 0 && nums[i] == nums[i - 1]) continue;
9             for (int j = i + 1; j < n - 2; j++) {
10                 if (j > i + 1 && nums[j] == nums[j - 1]) continue;
11                 int left = j + 1;
12                 int right = n - 1;
13                 while (left < right) {
14                     long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];
15                     if (sum == target) {
16                         result.add(Arrays.asList(nums[i], nums[j], nums[left], nums[right]));
17                         while (left < right && nums[left] == nums[left + 1]) left++;
18                         while (left < right && nums[right] == nums[right - 1]) right--;
19                         left++;
20                         right--;
21                     }
22                     else if (sum < target) {
23                         left++;
24                     }
25                     else {
26                         right--;
27                     }
28                 }
29             }
30         }
31         return result;
32     }
33 }
```

Accepted Runtime: 1 ms

## EXPERIMENT 13

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:

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

#### Example 2:

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

#### Example 3:

```
nums = [1], target = 0
-1
```

#### Constraints:

- $1 \leq \text{nums.length} \leq 5000$
- $-10^4 \leq \text{nums}[i] \leq 10^4$
- All values of `nums` are **unique**.
- `nums` is an ascending array that is possibly rotated.
- $-10^4 \leq \text{target} \leq 10^4$

The screenshot shows the LeetCode interface for problem 33, "Search in Rotated Sorted Array". The problem description on the left explains that an array sorted in ascending order has been rotated at an unknown index. It provides examples: [0,1,2,4,5,6,7] rotated by 3 indices becomes [4,5,6,7,0,1,2]. The task is to find the index of a target value, or -1 if it's not present, with a required O(log n) runtime complexity.

The code editor on the right contains a Java solution using a binary search approach. It initializes left (l) and right (r) pointers and enters a while loop. In each iteration, it finds the middle element (m). If the target equals nums[m], it returns m. If the target is greater than or equal to nums[l], it updates r = m - 1. Otherwise, it updates l = m + 1. If the loop ends without finding the target, it returns -1.

Below the code editor, the "Testcase" tab shows "Test Result" as "Accepted" with a runtime of 0 ms. Three test cases (Case 1, Case 2, Case 3) are all marked as passed.

## EXPERIMENT 14

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:**

nums = [5,7,7,8,8,10], target = 8  
[3,4]

**Example 2:**

nums = [5,7,7,8,8,10], target = 6  
[-1,-1]

**Example 3:**

nums = [], target = 0  
[-1,-1]

**Constraints:**

- $0 \leq \text{nums.length} \leq 10^5$
- $-10^9 \leq \text{nums}[i] \leq 10^9$
- nums is a non-decreasing array.

•  $-10^9 \leq \text{target} \leq 10^9$

The screenshot shows the LeetCode interface for the problem "34. Find First and Last Position of Element in Sorted Array". The problem description states: "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]`

Constraints:  
•  $0 \leq \text{nums.length} \leq 10^5$

The code editor shows a Java solution:

```
1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int[] ans = {-1, -1};
4
5         for (int i = 0; i < nums.length; i++) {
6             if (nums[i] == target) {
7                 ans[0] = i;
8                 break;
9             }
10        }
11
12        for (int i = nums.length - 1; i >= 0; i--) {
13            if (nums[i] == target) {
14                ans[1] = i;
15                break;
16            }
17        }
18
19        return ans;
20    }
21 }
22
```

The test results show "Accepted" with a runtime of 0 ms. The input for Case 1 is `nums = [5,7,7,8,8,10]`.