

EXPERIMENT LIST FOR PROGRAMMING ABILITY AND LOGIC

BUILDING – 21

NAME : IKSHU GOEL

BATCH : 2CSE5

ROLL NUMBER : 2410030774

WEEK : 19/1/26 - 25/1/26

LECTURE : 2

EXPERIMENT 1

Given an array **arr**, rotate the array by one position in clockwise direction.

Examples:

Input: arr[] = [1, 2, 3, 4, 5]

Output: [5, 1, 2, 3, 4]

Explanation: If we rotate arr by one position in clockwise 5 come to the front and

remaining those are shifted to the end.

The screenshot shows a web-based IDE interface for solving a problem. The browser address bar shows the URL: [geeksforgeeks.org/problems/cyclically-rotate-an-array-by-one2614/1](https://www.geeksforgeeks.org/problems/cyclically-rotate-an-array-by-one2614/1). The IDE has a top navigation bar with links for Courses, Tutorials, Practice, and Jobs. The main editor area displays a Java solution for rotating an array by one position clockwise. The code is as follows:

```
1- class Solution {
2-     public void rotate(int[] arr) {
3-         int last = arr[arr.length - 1];
4-
5-         for (int i = arr.length - 1; i > 0; i--) {
6-             arr[i] = arr[i - 1];
7-         }
8-
9-         arr[0] = last;
10-    }
11- }
12-
```

On the left side, there is an "Output Window" with a "Compilation Results" tab. It shows "Compilation Completed" for "Case 1". The input is "1 2 3 4 5", the "Your Output" is "5 1 2 3 4", and the "Expected Output" is "5 1 2 3 4". At the bottom right, there are buttons for "Custom Input", "Compile & Run", and "Submit".

EXPERIMENT 2

You are given an integer array **arr[]**. You need to find the **maximum** sum of a

subarray (containing at least one element) in the array **arr[]**.

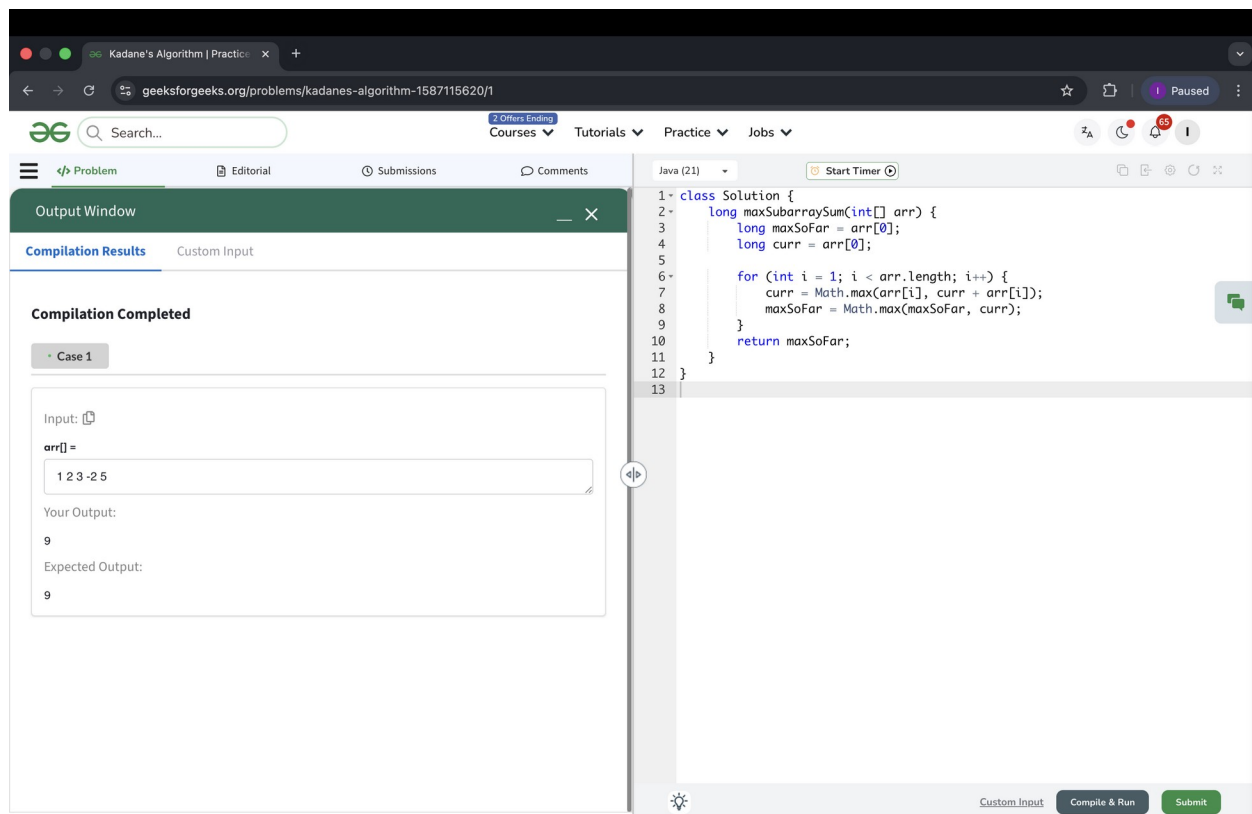
Note : A **subarray** is a continuous part of an array.

Examples:

Input: arr[] = [2, 3, -8, 7, -1, 2, 3]

Output: 11

Explanation: The subarray [7, -1, 2, 3] has the largest sum 11.



EXPERIMENT 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

Search Insert Position - Leet...

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

Array<>Submit

DescriptionEditorialSolutionsSubmissions

35. Search Insert Position

EasyTopicsCompanies

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.
- $-10^4 \leq \text{target} \leq 10^4$

18.5K413335 Online

Code

JavaAuto

```
1 class Solution {
2     public int searchInsert(int[] nums, int target) {
3         int l = 0, r = nums.length - 1;
4
5         while (l <= r) {
6             int mid = (l + r) / 2;
7             if (nums[mid] == target)
8                 return mid;
9             else if (nums[mid] < target)
10                l = mid + 1;
11            else
12                r = mid - 1;
13        }
14        return l;
15    }
16 }
```

SavedLn 15, Col 6

TestcaseTest Result

AcceptedRuntime: 0 ms

Case 1Case 2Case 3

Input

nums =
[1,3,5,6]

target =
5

Output

Search Insert Position - Leet...

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

Array<>Submit

DescriptionEditorialSolutionsSubmissions

35. Search Insert Position

EasyTopicsCompanies

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.
- $-10^4 \leq \text{target} \leq 10^4$

18.5K413336 Online

Code

JavaAuto

```
1 class Solution {
2     public int searchInsert(int[] nums, int target) {
3         int l = 0, r = nums.length - 1;
4
5         while (l <= r) {
6             int mid = (l + r) / 2;
7             if (nums[mid] == target)
8                 return mid;
9             else if (nums[mid] < target)
10                l = mid + 1;
11            else
12                r = mid - 1;
13        }
14        return l;
15    }
16 }
```

SavedLn 15, Col 6

TestcaseTest Result

target =
5

Output

2

Expected

2

Contribute a testcase

EXPERIMENT 4

Given an array of integers `nums` and an integer `target`, return *indices of the two*

numbers such that they add up to target.

You may assume that each input would have **exactly one solution**, and you may

not use the *same* element twice.

You can return the answer in any order.

Example 1:

Input: `nums = [2,7,11,15]`, `target = 9`

Output: [0,1]

Explanation: Because $\text{nums}[0] + \text{nums}[1] == 9$, we return [0, 1].

The screenshot shows the LeetCode interface for the 'Two Sum' problem. The left panel contains the problem description, which asks for two indices in an array that sum to a target. It includes three examples and constraints. The right panel shows a Java solution using a HashMap to store the complement of each number. Below the code, the 'Test Result' section shows 'Accepted' status with a runtime of 0 ms. The input for the test case is `nums = [2,7,11,15]` and `target = 9`.

1. Two Sum

Easy Topics Companies Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have *exactly one* solution, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: `nums = [2,7,11,15]`, `target = 9`
Output: `[0,1]`
Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

Example 2:

Input: `nums = [3,2,4]`, `target = 6`
Output: `[1,2]`

Example 3:

Input: `nums = [3,3]`, `target = 6`
Output: `[0,1]`

Constraints:

- `2 <= nums.length <= 104`

```
1 import java.util.*;
2
3 class Solution {
4     public int[] twoSum(int[] nums, int target) {
5         HashMap<Integer, Integer> map = new HashMap<>();
6
7         for (int i = 0; i < nums.length; i++) {
8             int rem = target - nums[i];
9             if (map.containsKey(rem)) {
10                 return new int[]{map.get(rem), i};
11             }
12             map.put(nums[i], i);
13         }
14         return new int[]{};
15     }
16 }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

`nums =`
`[2,7,11,15]`

`target =`
`9`

Two Sum - LeetCode

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

Array

Submit

0

Premium

Description

Editorial

Solutions

Submissions

1. Two Sum

EasyTopicsCompaniesHint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: `nums = [2,7,11,15]`, `target = 9`
Output: `[0,1]`
Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

Example 2:

Input: `nums = [3,2,4]`, `target = 6`
Output: `[1,2]`

Example 3:

Input: `nums = [3,3]`, `target = 6`
Output: `[0,1]`

Constraints:

- `2 <= nums.length <= 104`

67K1.8K2990 Online

Code

Java

```
1 import java.util.*;
2
3 class Solution {
4     public int[] twoSum(int[] nums, int target) {
5         HashMap<Integer, Integer> map = new HashMap<>();
6
7         for (int i = 0; i < nums.length; i++) {
8             int rem = target - nums[i];
9             if (map.containsKey(rem)) {
10                 return new int[]{map.get(rem), i};
11             }
12             map.put(nums[i], i);
13         }
14         return new int[]{};
15     }
16 }
```

SavedLn 17, Col 1

Testcase

Test Result

target = 9

Output [0,1]

Expected [0,1]

Contribute a testcase

EXPERIMENT 5

You are given an array **arr[]** of non-negative numbers. Each number tells you the **maximum number of steps** you can jump forward from that position.

For example:

- If **arr[i] = 3**, you can jump to index **i + 1**, **i + 2**, or **i + 3** from position **i**.
- If **arr[i] = 0**, you **cannot jump forward** from that position.

Your task is to find the **minimum number of jumps** needed to move from the **first** position in the array to the **last** position.

Note: Return **-1** if you can't reach the end of the array.

Examples :

Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]

Output: 3

Explanation: First jump from 1st element to 2nd element with value 3. From here

we jump to 5th element with value 9, and from here we will jump to the last.

Minimum Jumps | Practice | X +

geeksforgeeks.org/problems/minimum-number-of-jumps-1587115620/1

2 Offers Ending

Courses Tutorials Practice Jobs

Paused

Search...

Problem Editorial Submissions Comments

Output Window

Compilation Results Custom Input

Compilation Completed

Case 1

Input:

arr[] =
1 3 5 8 9 2 6 7 6 8 9

Your Output:
3

Expected Output:
3

Java (21)

Start Timer

```
1 class Solution {
2     public int minJumps(int[] arr) {
3         if (arr.length <= 1)
4             return 0;
5         if (arr[0] == 0)
6             return -1;
7
8         int maxReach = arr[0];
9         int steps = arr[0];
10        int jumps = 1;
11
12        for (int i = 1; i < arr.length; i++) {
13            if (i == arr.length - 1)
14                return jumps;
15
16            maxReach = Math.max(maxReach, i + arr[i]);
17            steps--;
18
19            if (steps == 0) {
20                jumps++;
21                if (i >= maxReach)
22                    return -1;
23                steps = maxReach - i;
24            }
25        }
26        return -1;
27    }
28 }
29
```

Custom Input Compile & Run Submit