



Experiment 5

Student Name: Niraj Kumar

Branch: CSE

Semester: 6th

Subject: AP

UID: 22BCS16736

Section: 638/A

DOP: 18-02-2014

Subject Code:22CSP-351

Aim:

Problem-1: Beautiful Array

Algorithm:

- **Start with Base Case**
- Begin with a list containing only [1] as the base case.
- **Build Odd and Even Sequences**
- Use a divide-and-conquer approach:
 - Generate odd numbers: $2 * \text{num} - 1$ (as long as they are $\leq n$).
 - Generate even numbers: $2 * \text{num}$ (as long as they are $\leq n$).
- Append these numbers in order to ensure no three numbers satisfy $2 * \text{nums}[k] == \text{nums}[i] + \text{nums}[j]$.
- **Convert List to Array and Return**
- Store the result in an integer array and return it

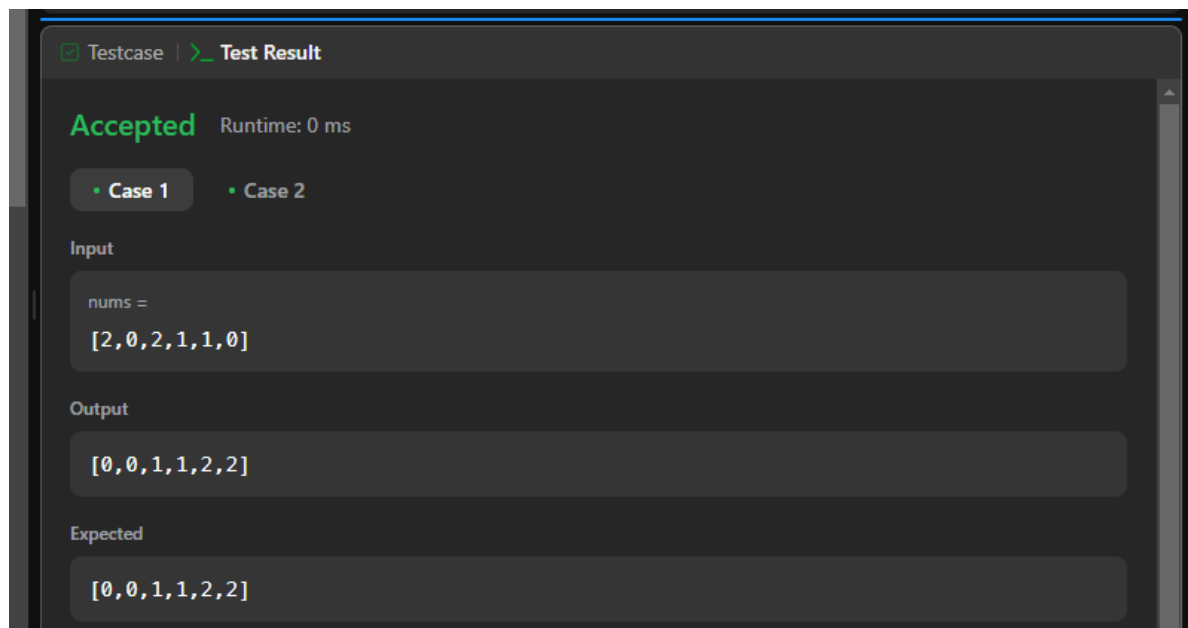
Code:

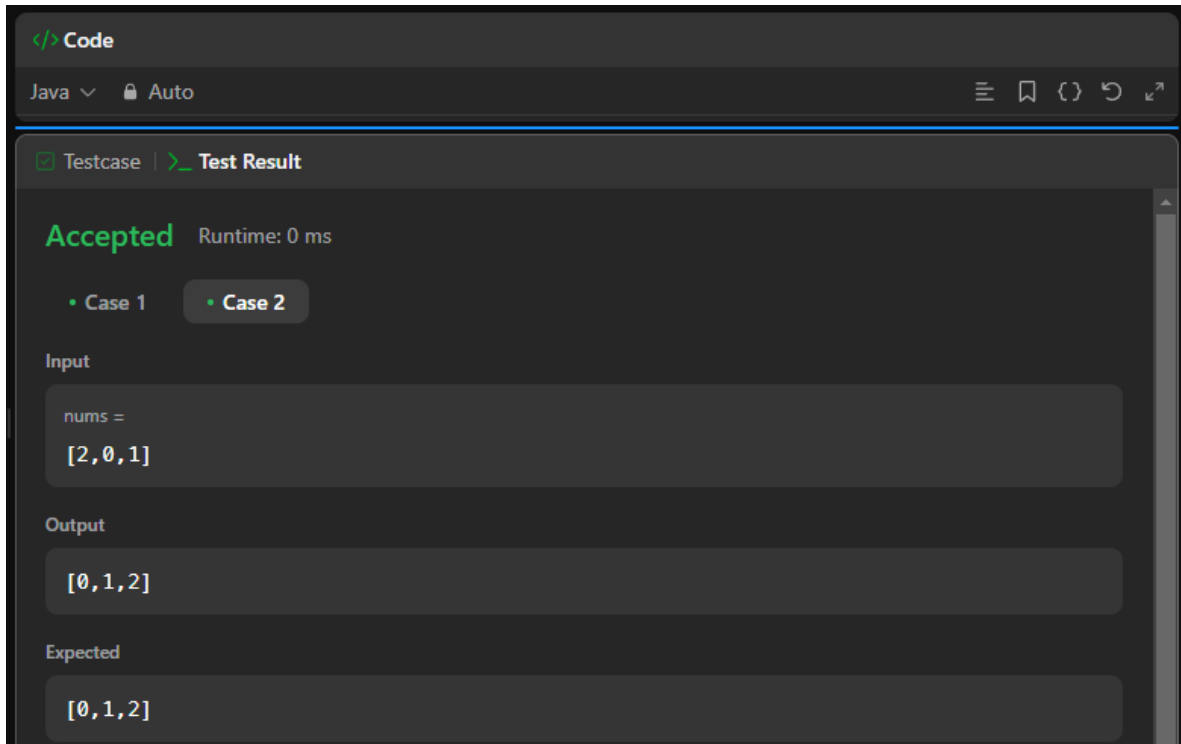
```
import java.util.Arrays;

class Solution {
    public void sortColors(int[] nums) {
        int low = 0, mid = 0, high = nums.length - 1;
        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums, low, mid);
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                swap(nums, mid, high);
                high--;
            }
        }
    }
}
```

```
private void swap(int[] nums, int i, int j) {  
    int temp = nums[i];  
    nums[i] = nums[j];  
    nums[j] = temp;  
}  
}
```

Output:





Code

Java ▾ Auto

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

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

Expected

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

Aim:

Problem-2: The Skyline Problem

Algorithm :

Use a Min-Heap (Priority Queue)

- Create a min-heap (PriorityQueue) to store the top k largest elements.

Iterate Over the Array

- Insert each number into the min-heap.
- If the heap size exceeds k, remove the smallest element to keep only the top k largest elements.

Return the Kth Largest Element

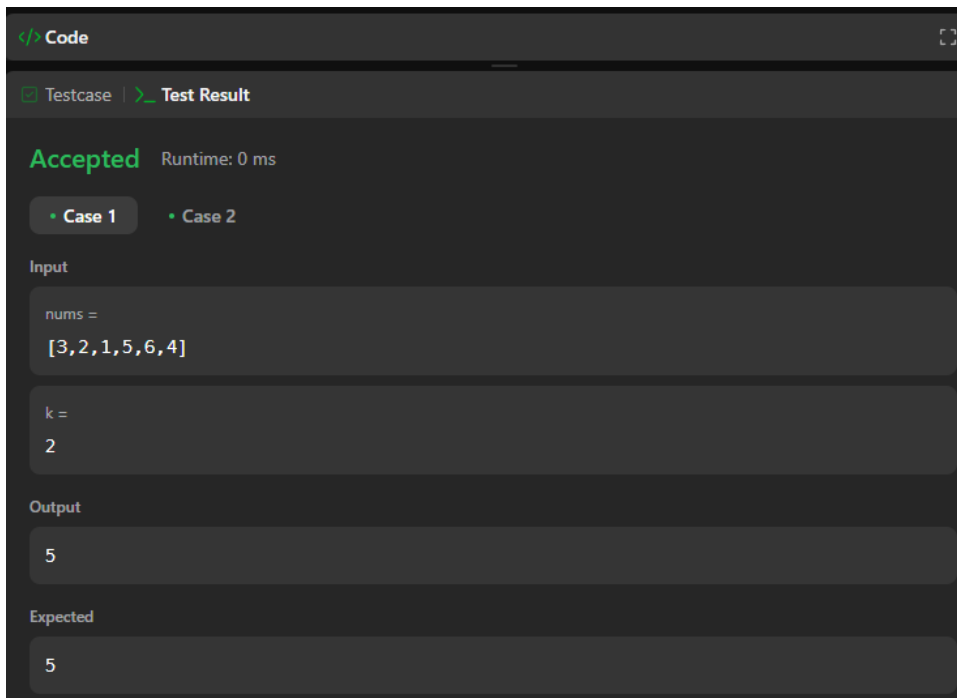
- The root of the min-heap (peek()) gives the kth largest element in the array.

Code :

```
import java.util.PriorityQueue;
import java.util.*;

class Solution {
    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int num : nums) {
            minHeap.offer(num);
            if (minHeap.size() > k) {
                minHeap.poll();
            }
        }
        return minHeap.peek();
    }
}
```

Output:



The screenshot shows a code editor with the following code:

```
</> Code
```

Below the code editor, there are tabs for "Testcase" and "Test Result". The "Test Result" tab is active, showing the following information:

- Accepted** Runtime: 0 ms
- Case 1** (selected) • Case 2
- Input**
 - nums = [3,2,1,5,6,4]
 - k = 2
- Output**
 - 5
- Expected**
 - 5