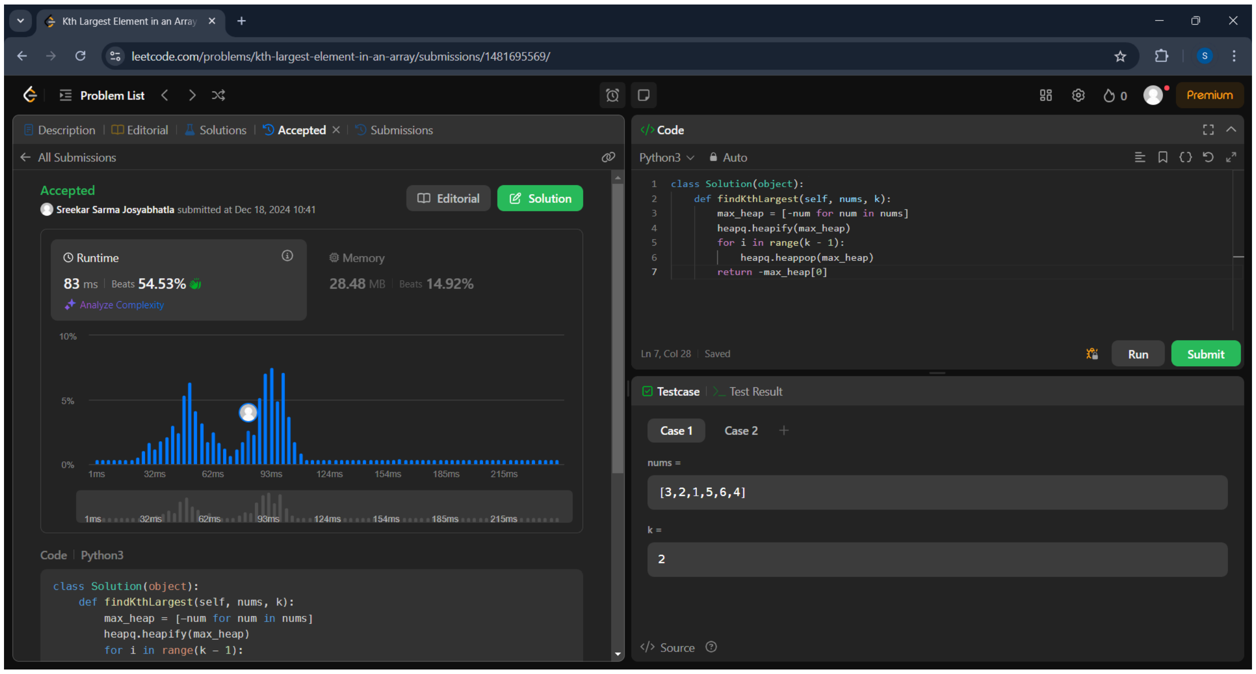
N.VISHNU PAVAN KUMAR
VU21CSEN0300271

Advanced Coding -3

Kth Largest Element in an Array

```
class Solution(object):
    def findKthLargest(self, nums, k):
        max_heap = [-num for num in nums]
        heapq.heapify(max_heap)
        for i in range(k - 1):
            heapq.heappop(max_heap)
        return -max_heap[0]
```

OUTPUT



leetcode.com/problems/kth-largest-element-in-an-array/submissions/1481695569/

Accepted
Sreekar Sarma Josyabhatla submitted at Dec 18, 2024 10:41

Runtime
83 ms | Beats 54.53%

Memory
28.48 MB | Beats 14.92%

Code | Python3

```
class Solution(object):
    def findKthLargest(self, nums, k):
        max_heap = [-num for num in nums]
        heapq.heapify(max_heap)
        for i in range(k - 1):
```

```
class Solution(object):
    def findKthLargest(self, nums, k):
        max_heap = [-num for num in nums]
        heapq.heapify(max_heap)
        for i in range(k - 1):
            heapq.heappop(max_heap)
        return -max_heap[0]
```

Ln 7, Col 28 | Saved

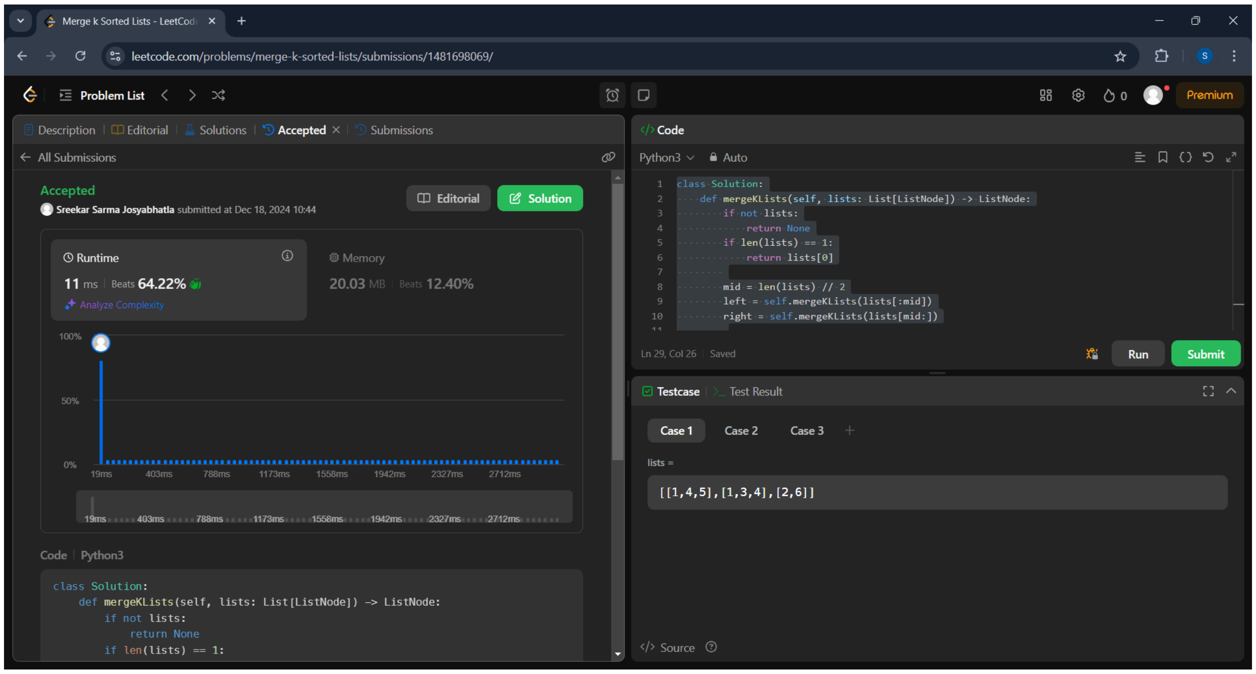
nums =
[3,2,1,5,6,4]

k =
2

1. **Merge k Sorted Lists**

```
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        if not lists:
            return None
        if len(lists) == 1:
            return lists[0]

        mid = len(lists) // 2
        left = self.mergeKLists(lists[:mid])
        right = self.mergeKLists(lists[mid:])
        return self.merge(left, right)

    def merge(self, l1, l2):
        dummy = ListNode(0)
        curr = dummy

        while l1 and l2:
            if l1.val < l2.val:
                curr.next = l1
                l1 = l1.next
            else:
                curr.next = l2
                l2 = l2.next
            curr = curr.next

        curr.next = l1 or l2

        return dummy.next
```

OUTPUT



Merge k Sorted Lists - LeetCode

leetcode.com/problems/merge-k-sorted-lists/submissions/1481698069/

Accepted

Sreekar Sarma Josyabhatla submitted at Dec 18, 2024 10:44

Runtime

11 ms | Beats 64.22%

Memory

20.03 MB | Beats 12.40%

Code | Python3

```
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        if not lists:
            return None
        if len(lists) == 1:
```

Code

Python3 Auto

```
class Solution:
    def mergeKLists(self, lists: List[ListNode]) -> ListNode:
        if not lists:
            return None
        if len(lists) == 1:
            return lists[0]
        
        mid = len(lists) // 2
        left = self.mergeKLists(lists[:mid])
        right = self.mergeKLists(lists[mid:])
```

Ln 29, Col 26 | Saved

Testcase | Test Result

Case 1 Case 2 Case 3

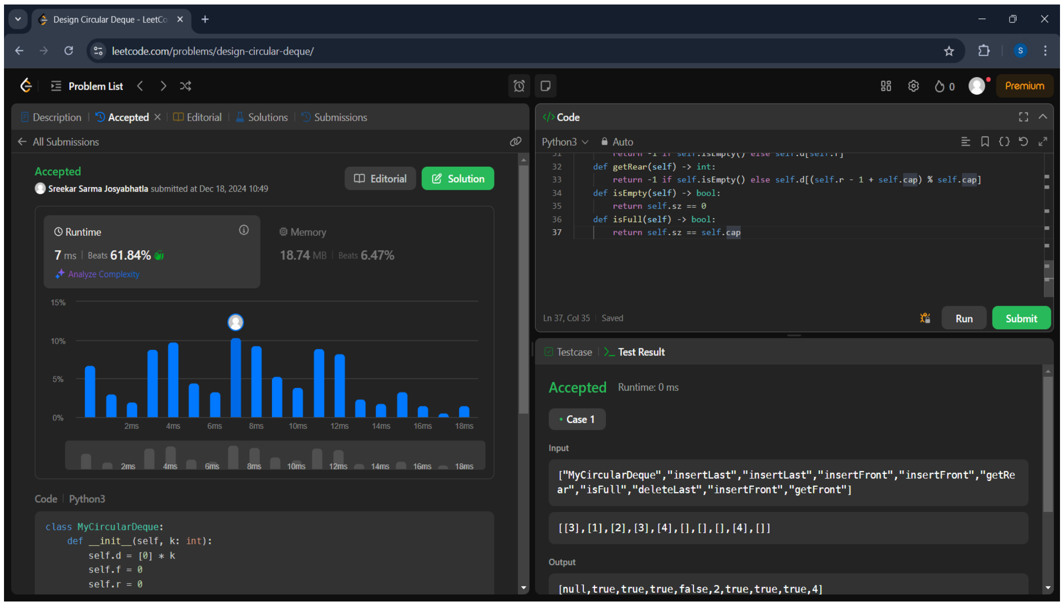
lists =

```
[[1,4,5],[1,3,4],[2,6]]
```

2. **Design Circular Deque**

```
class MyCircularDeque:
    def __init__(self, k: int):
        self.d = [0] * k
        self.f = 0
        self.r = 0
        self.sz = 0
        self.cap = k
    def insertFront(self, v: int) -> bool:
        if self.isFull(): return False
        self.f = (self.f - 1 + self.cap) % self.cap
        self.d[self.f] = v
        self.sz += 1
        return True
    def insertLast(self, v: int) -> bool:
        if self.isFull(): return False
        self.d[self.r] = v
        self.r = (self.r + 1) % self.cap
        self.sz += 1
        return True
    def deleteFront(self) -> bool:
        if self.isEmpty(): return False
        self.f = (self.f + 1) % self.cap
        self.sz -= 1
        return True
    def deleteLast(self) -> bool:
        if self.isEmpty(): return False
        self.r = (self.r - 1 + self.cap) % self.cap
        self.sz -= 1
        return True
    def getFront(self) -> int:
        return -1 if self.isEmpty() else self.d[self.f]
    def getRear(self) -> int:
        return -1 if self.isEmpty() else self.d[(self.r - 1 + self.cap) % self.cap]
    def isEmpty(self) -> bool:
        return self.sz == 0
    def isFull(self) -> bool:
        return self.sz == self.cap
```

<u>OUTPUT:</u>



Accepted

Sreekar Sarma Josyabhatla submitted at Dec 18, 2024 10:49

Runtime: 7 ms | Beats 61.84%

Analyze Complexity

Memory: 18.74 MB | Beats 6.47%

Code | Python3

```python
class MyCircularDeque:
    def __init__(self, k: int):
        self.d = [0] * k
        self.f = 0
        self.r = 0
```

```python
    def getRear(self) -> int:
        return -1 if self.isEmpty() else self.d[(self.r - 1 + self.cap) % self.cap]
    def isEmpty(self) -> bool:
        return self.sz == 0
    def isFull(self) -> bool:
        return self.sz == self.cap
```

Ln 37, Col 35 | Saved

Accepted Runtime: 0 ms

Case 1

Input

```
["MyCircularDeque","insertLast","insertLast","insertFront","insertFront","getRe
ar","isFull","deleteLast","insertFront","getFront"]
```

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

Output

```
[null,true,true,true,false,2,true,true,true,4]
```