Merge Two Sorted Lists (/problems/merge-two-sorted-lists/)

Submission Detail

208 / 208 test cases passed.

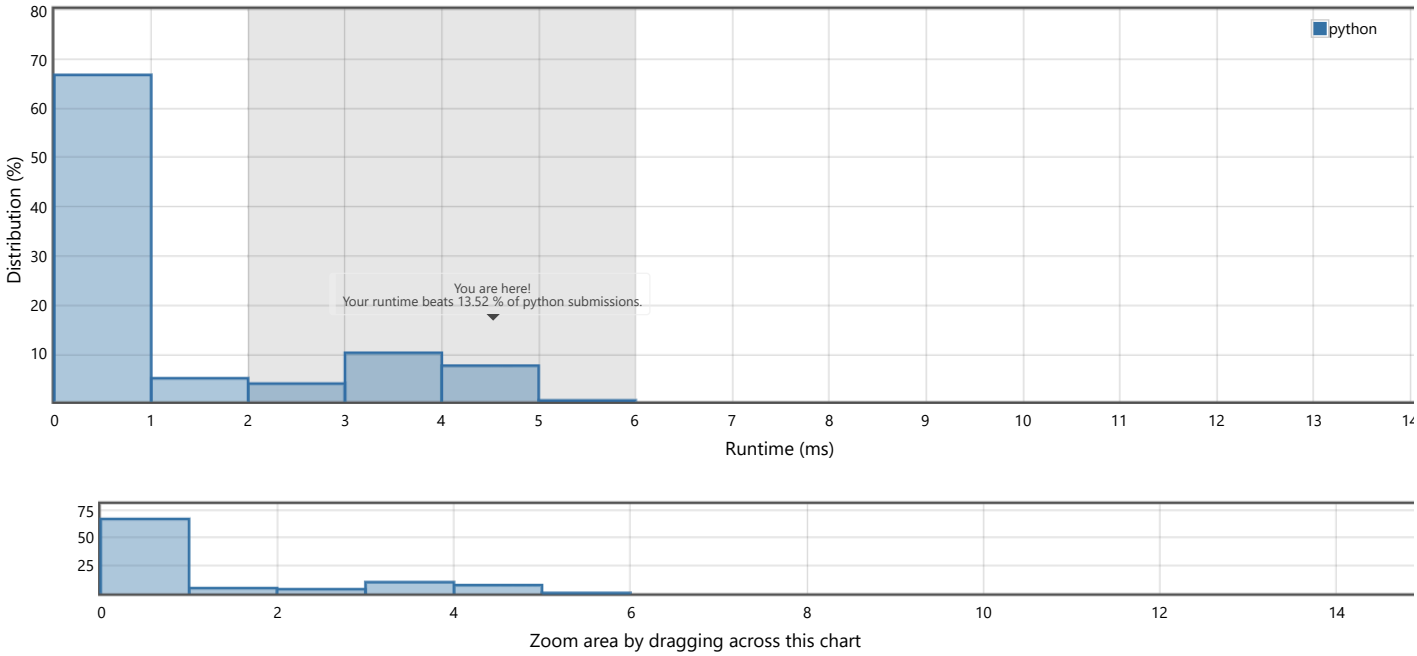
Status: Accepted

Runtime: **4 ms**

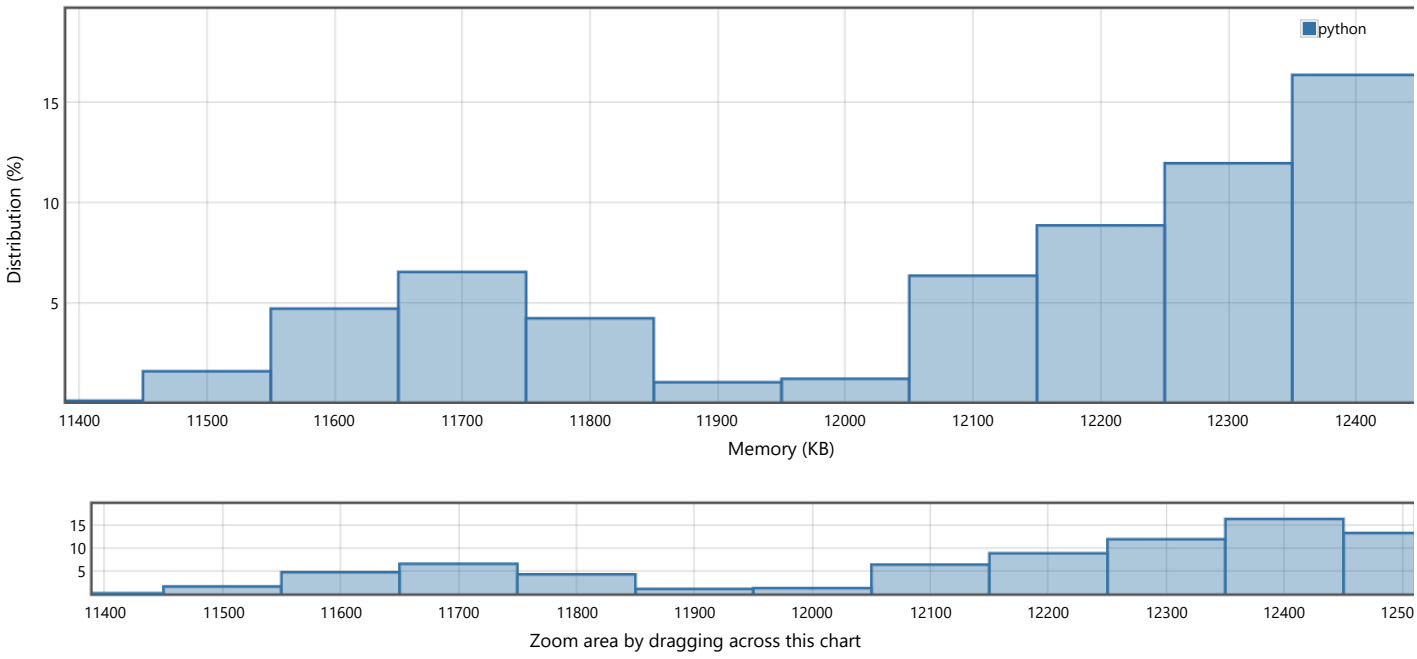
Memory Usage: **12.6 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Merge Two Sorted Lists**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  # Definition for singly-linked list.
2  # class ListNode(object):
3  #     def __init__(self, val=0, next=None):
4  #         self.val = val
```

```
#         self.next = next
class Solution(object):
    def mergeTwoLists(self, list1, list2):
        """
        :type list1: Optional[ListNode]
        :type list2: Optional[ListNode]
        :rtype: Optional[ListNode]
        """
        dummy = ListNode()
        current = dummy

        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next

        if list1:
            current.next = list1
        elif list2:
            current.next = list2

        return dummy.next

# Example usage
list1 = ListNode(1, ListNode(2, ListNode(4)))
list2 = ListNode(1, ListNode(3, ListNode(4)))
solution = Solution()
merged_list = solution.mergeTwoLists(list1, list2)

# Function to print the linked list
def printList(node):
    while node:
        print(node.val)
        node = node.next
    print("None")

printList(merged_list)  # Output: 1 -> 1 -> 2 -> 3 -> 4 -> 4 -> None
```

Back to problem (/problems/merge-two-sorted-lists/)

Valid Parentheses (/problems/valid-parentheses/)

Submission Detail

100 / 100 test cases passed.

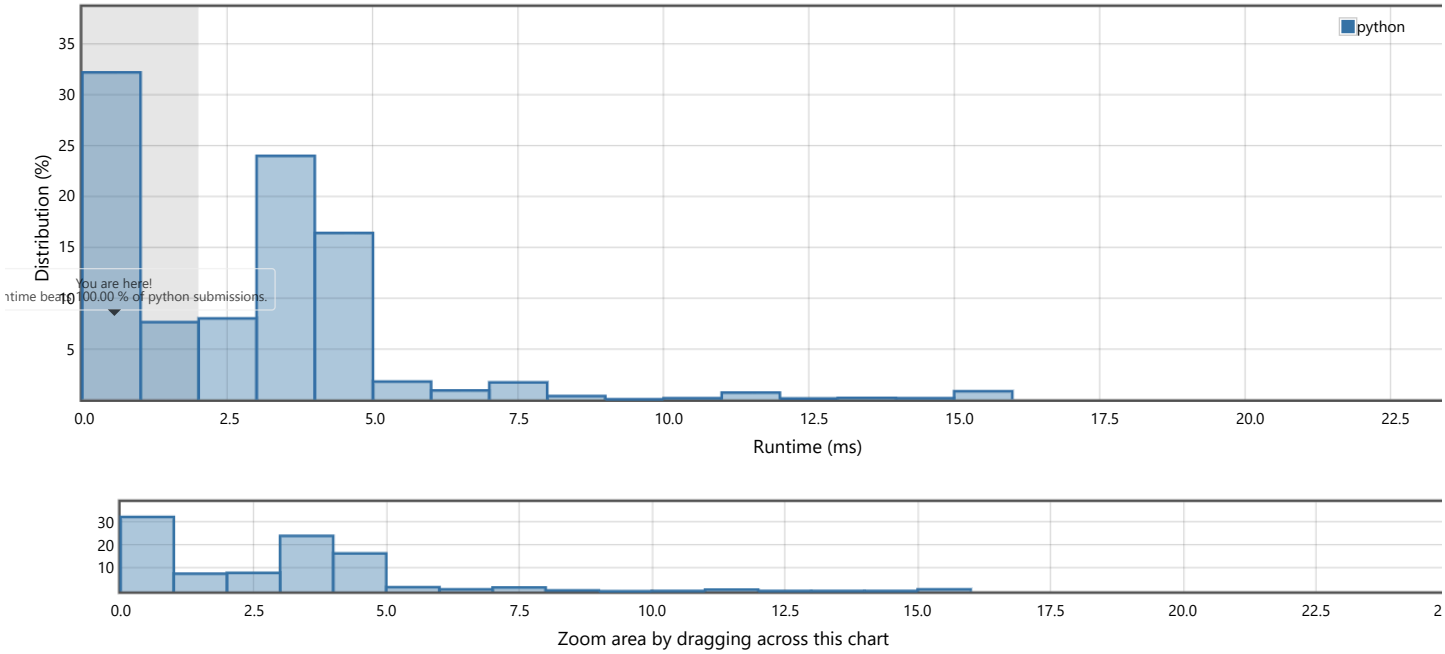
Status: Accepted

Runtime: **0 ms**

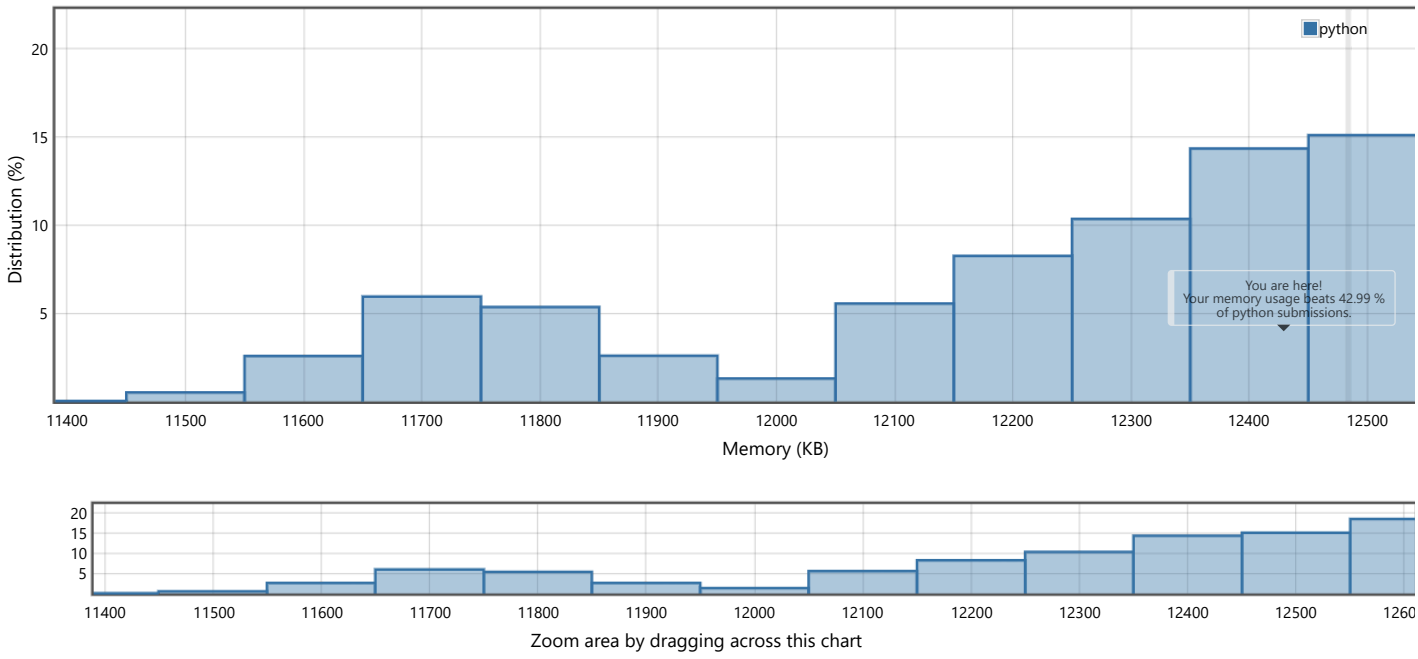
Memory Usage: **12.5 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Valid Parentheses**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  class Solution(object):
2      def isValid(self, s):
3          """
4          :type s: str
```

```
        :rtype: bool
        """
        stack = []
        mapping = {")": "(", "}": "{", "]": "["}

        for char in s:
            if char in mapping:
                top_element = stack.pop() if stack else '#'
                if mapping[char] != top_element:
                    return False
            else:
                stack.append(char)

        return not stack

# Example usage
s = "()[]{}"
solution = Solution()
print(solution.isValid(s))  # Output: True

```

Back to problem (/problems/valid-parentheses/)

Largest Number (/problems/largest-number/)

Submission Detail

235 / 235 test cases passed.

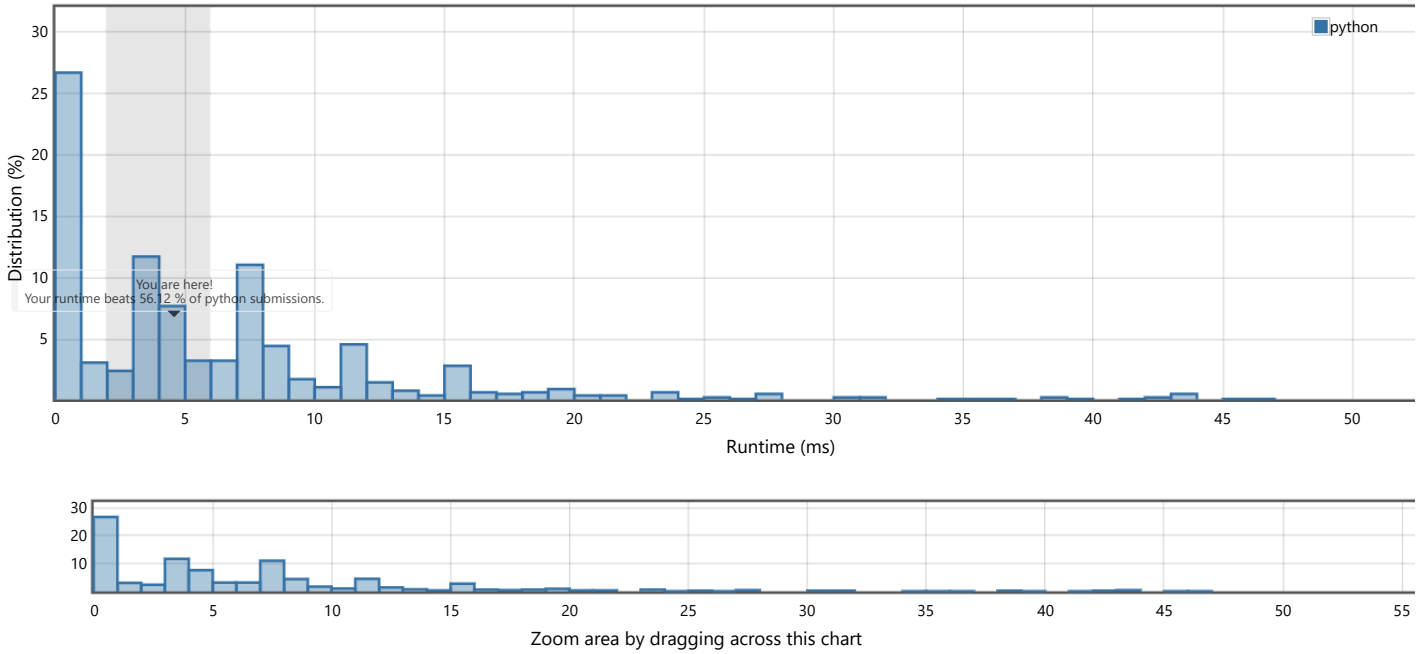
Status: **Accepted**

Runtime: **4 ms**

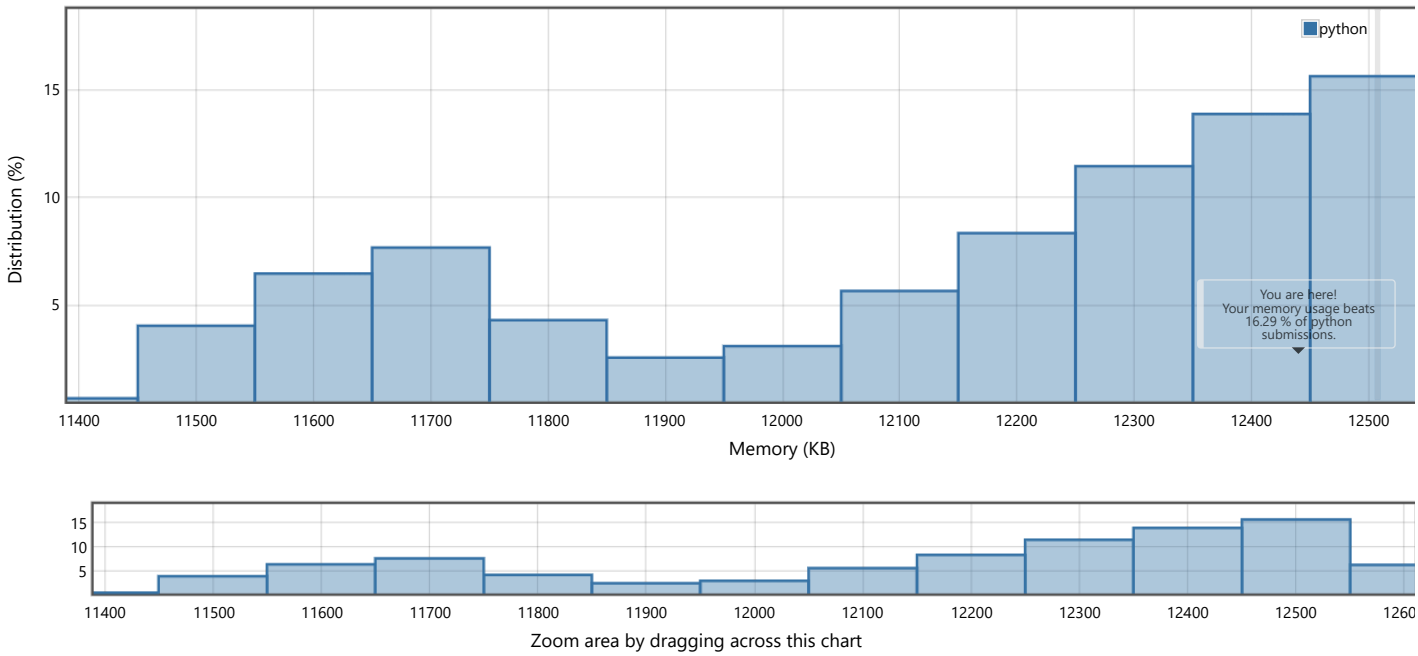
Memory Usage: **12.5 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Largest Number**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  class Solution(object):
2      def largestNumber(self, nums):
3          """
4          :type nums: List[int]
```

```
        :rtype: str
        """
        # Convert all integers to strings
        nums = list(map(str, nums))

        # Sort the numbers based on their concatenated value
        nums.sort(key=lambda x: x*10, reverse=True)

        # Join the sorted numbers to form the largest number
        largest_num = ''.join(nums)

        # Handle the case where the largest number is "0"
        if largest_num[0] == '0':
            return '0'

        return largest_num

# Example usage
nums = [10, 2]
solution = Solution()
print(solution.largestNumber(nums))  # Output: "210"
```

Back to problem (/problems/largest-number/)

Bitwise AND of Numbers Range (/problems/bitwise-and-of-numbers-range/)

Submission Detail

8271 / 8271 test cases passed.

Status: **Accepted**

Runtime: **9 ms**
Memory Usage: **12.3 MB**

Submitted: **1 day, 23 hours ago**

Submitted Code: 1 day, 23 hours ago

Language: python

```
1  class Solution(object):
2      def rangeBitwiseAnd(self, left, right):
3          """
4          :type left: int
```

```
        :type right: int
        :rtype: int
        """
        while left < right:
            right = right & (right - 1)
        return right

# Example usage
left = 5
right = 7
solution = Solution()
print(solution.rangeBitwiseAnd(left, right))  # Output: 4

```

Back to problem (/problems/bitwise-and-of-numbers-range/)

Find the Index of the First Occurrence in a String (/problems/find-the-index-of-the-first-occurrence-in-a-string/)

Submission Detail

83 / 83 test cases passed.

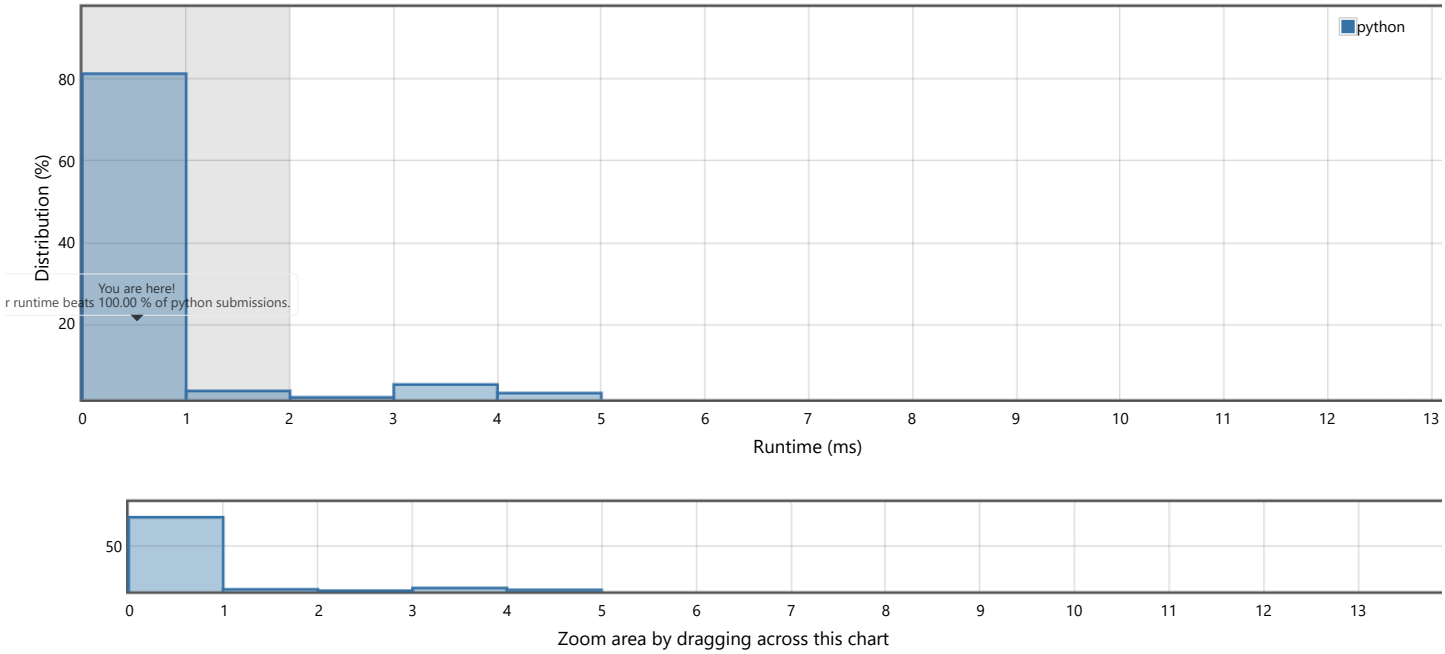
Status: **Accepted**

Runtime: **0 ms**

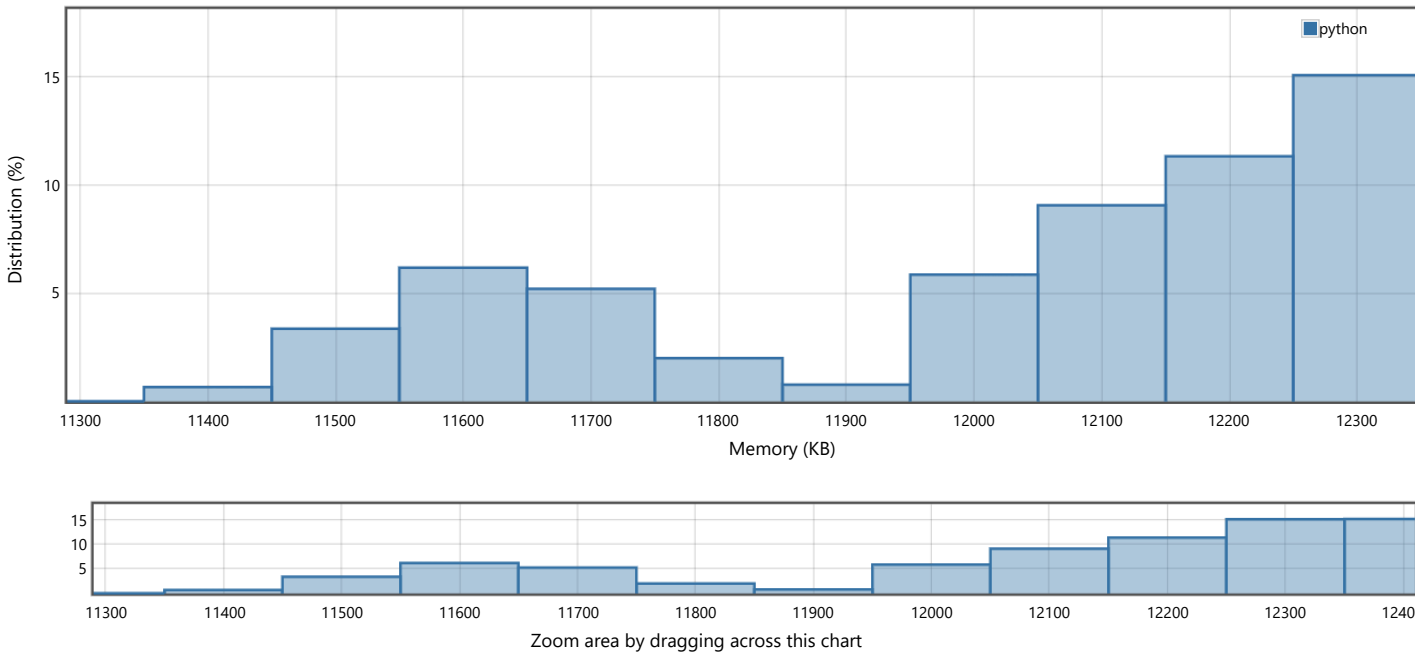
Memory Usage: **12.5 MB**

Submitted: **2 days ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Find the Index of the First Occurrence in a String**

Submitted Code: 2 days ago

Language: python

```
1  class Solution(object):
2      def strStr(self, haystack, needle):
3          """
4          :type haystack: str
```

```
        :type needle: str
        :rtype: int
        """
        # Check if needle is empty
        if not needle:
            return 0

        # Get the lengths of haystack and needle
        len_haystack = len(haystack)
        len_needle = len(needle)

        # Loop through the haystack
        for i in range(len_haystack - len_needle + 1):
            # Check if the substring matches the needle
            if haystack[i:i + len_needle] == needle:
                return i

        # If needle is not found, return -1
        return -1

# Example usage
haystack = "sadbutsad"
needle = "sad"
solution = Solution()
print(solution.strStr(haystack, needle))  # Output: 0
```

Back to problem (/problems/find-the-index-of-the-first-occurrence-in-a-string/)

Sqrt(x) (/problems/sqrtx/)

Submission Detail

1017 / 1017 test cases passed.

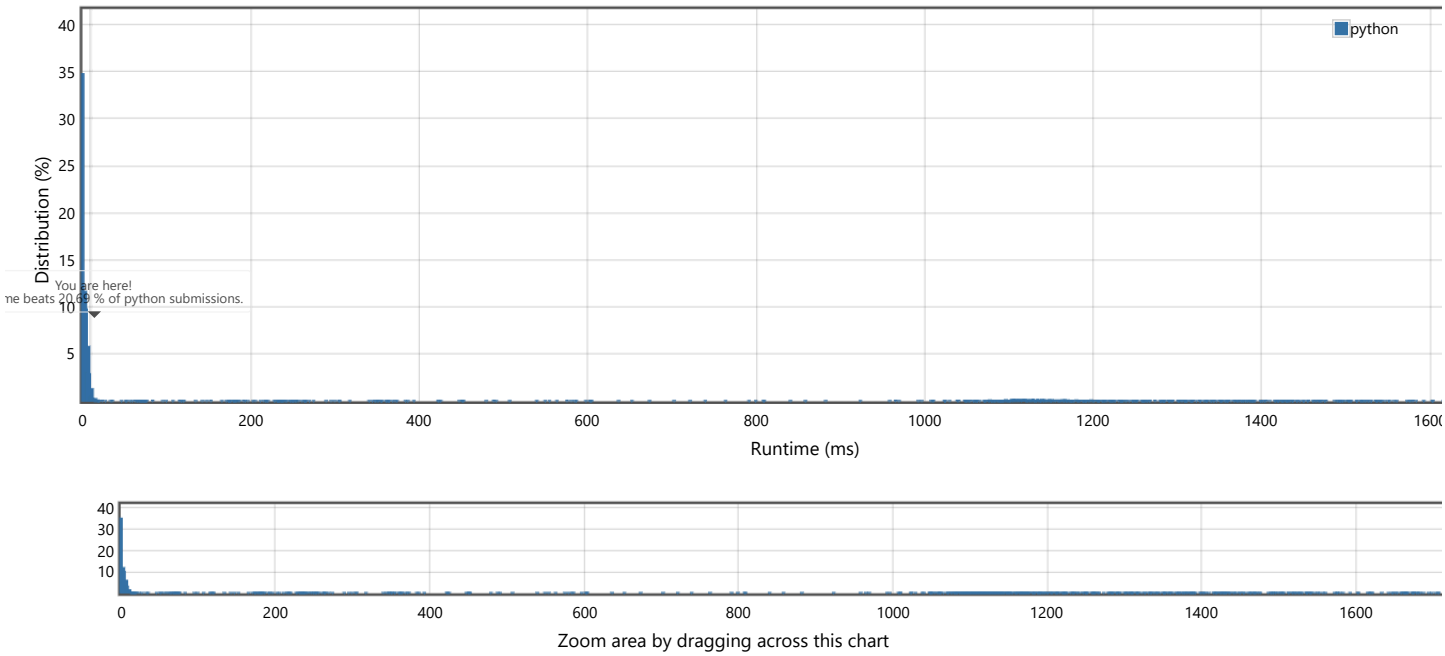
Status: Accepted

Runtime: **10 ms**

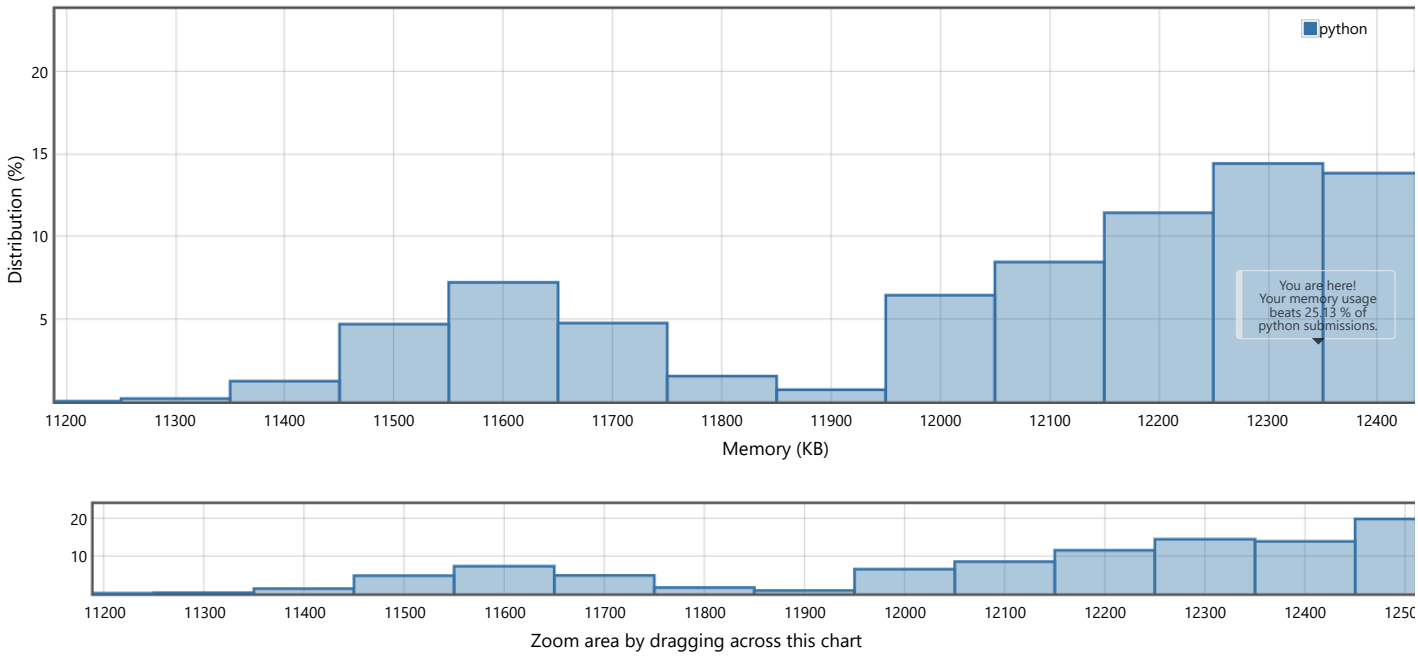
Memory Usage: **12.4 MB**

Submitted: **1 day, 13 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Sqrt(x)**

Submitted Code: 1 day, 13 hours ago

Language: python

```
1  class Solution(object):
2      def mySqrt(self, x):
3          if x < 2:
4              return x
```

```
5
6           left = 2
7           right = x // 2
8
9           while left <= right:
10              mid = (left + right) // 2
11              num = mid * mid
12
13              if num == x:
14                  return mid
15              elif num < x:
16                  left = mid + 1
17              else:
18                  right = mid - 1
19
20          return right
21
22  # Example usage
23  solution = Solution()
24  print(solution.mySqrt(4))  # Output: 2
25  print(solution.mySqrt(8))  # Output: 2
```

Back to problem (/problems/sqrtx/)

N-Queens (/problems/n-queens/)

Submission Detail

9 / 9 test cases passed.

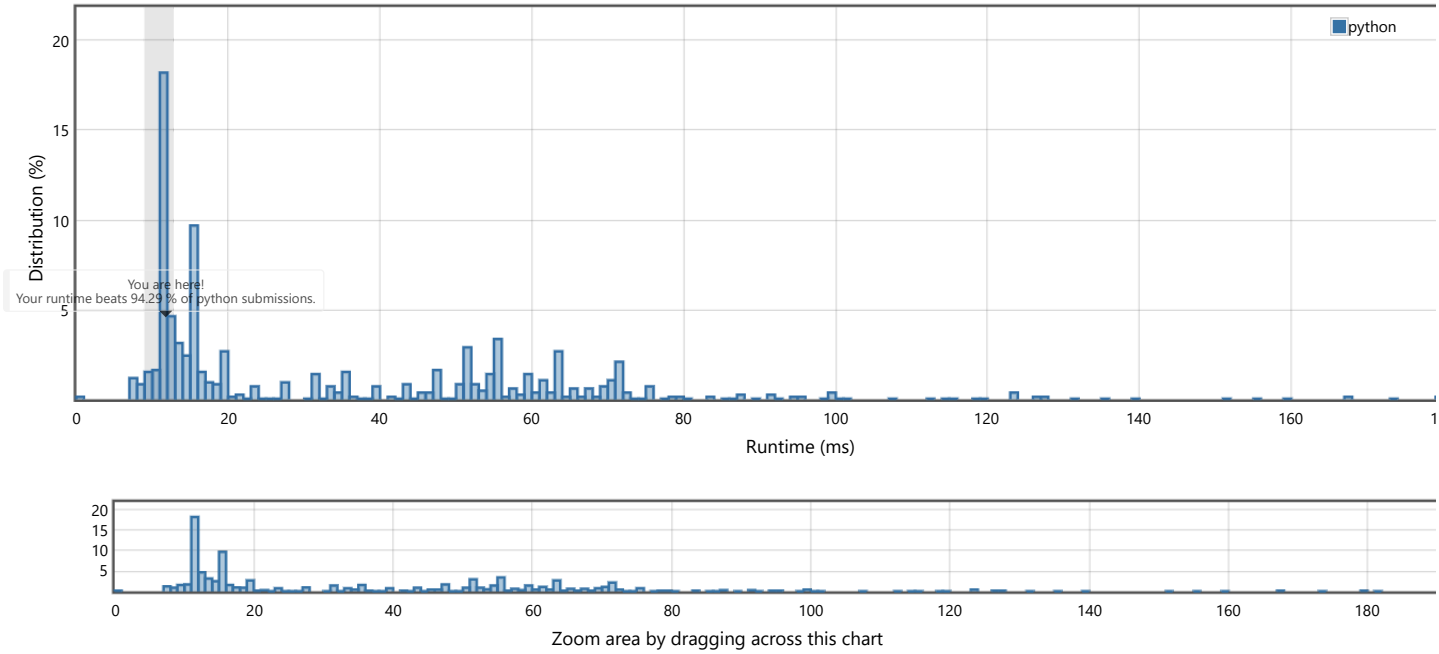
Status: Accepted

Runtime: **11 ms**

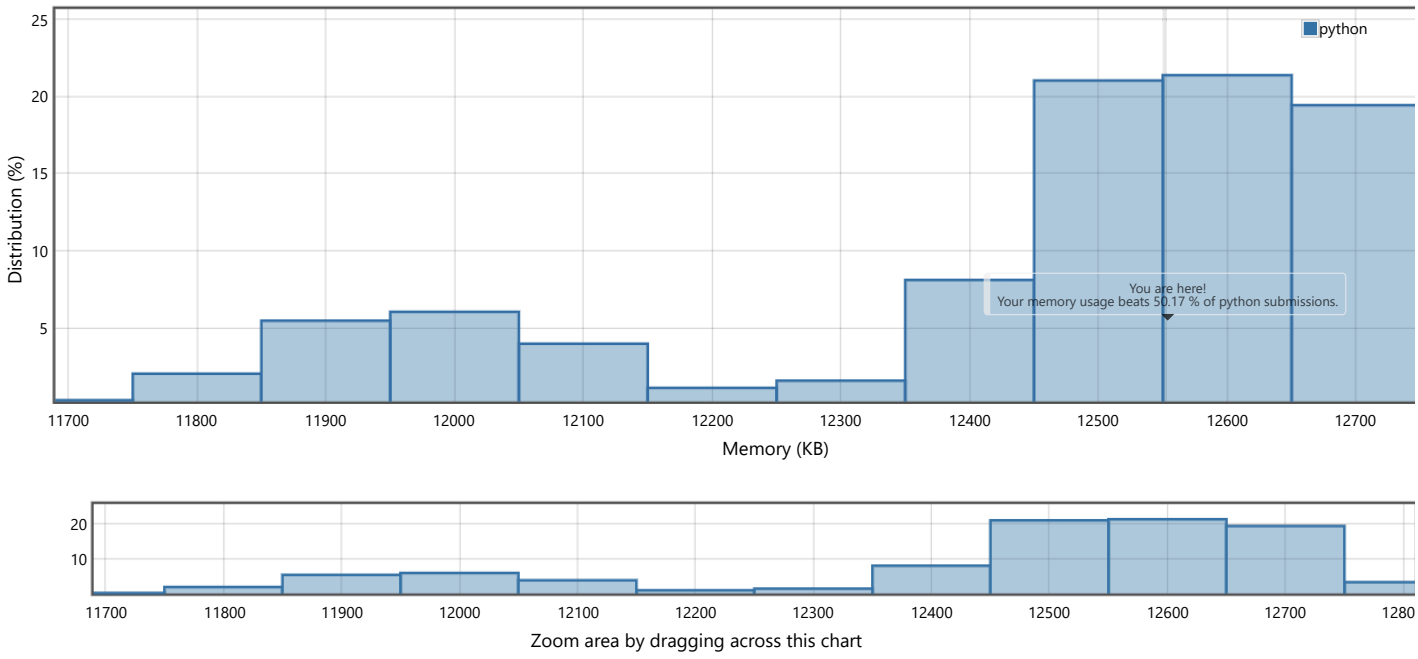
Memory Usage: **12.6 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **N-Queens**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  class Solution(object):
2      def solveNQueens(self, n):
3          """
4          :type n: int
```

```
        :rtype: List[List[str]]
        """
        def backtrack(row, diagonals, anti_diagonals, cols, state):
            if row == n:
                board = []
                for r in state:
                    board.append("".join(r))
                result.append(board)
                return

            for col in range(n):
                curr_diag = row - col
                curr_anti_diag = row + col
                if col in cols or curr_diag in diagonals or curr_anti_diag in anti_diagonals:
                    continue

                cols.add(col)
                diagonals.add(curr_diag)
                anti_diagonals.add(curr_anti_diag)
                state[row][col] = 'Q'

                backtrack(row + 1, diagonals, anti_diagonals, cols, state)

                cols.remove(col)
                diagonals.remove(curr_diag)
                anti_diagonals.remove(curr_anti_diag)
                state[row][col] = '.'

        result = []
        state = [["."] * n for _ in range(n)]
        backtrack(0, set(), set(), set(), state)
        return result

# Example usage
solution = Solution()
print(solution.solveNQueens(4))  # Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
```

Back to problem (/problems/n-queens/)

Binary Tree Inorder Traversal (/problems/binary-tree-inorder-traversal/)

Submission Detail

71 / 71 test cases passed.

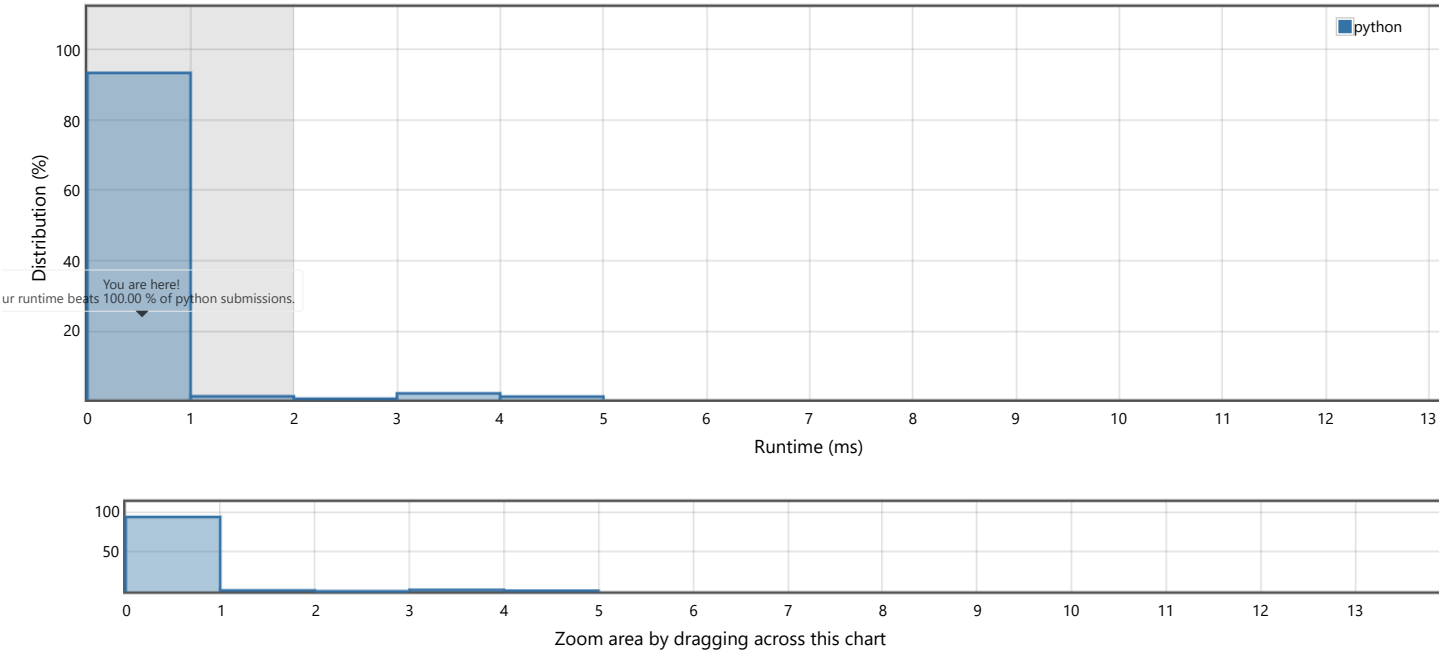
Status: **Accepted**

Runtime: **0 ms**

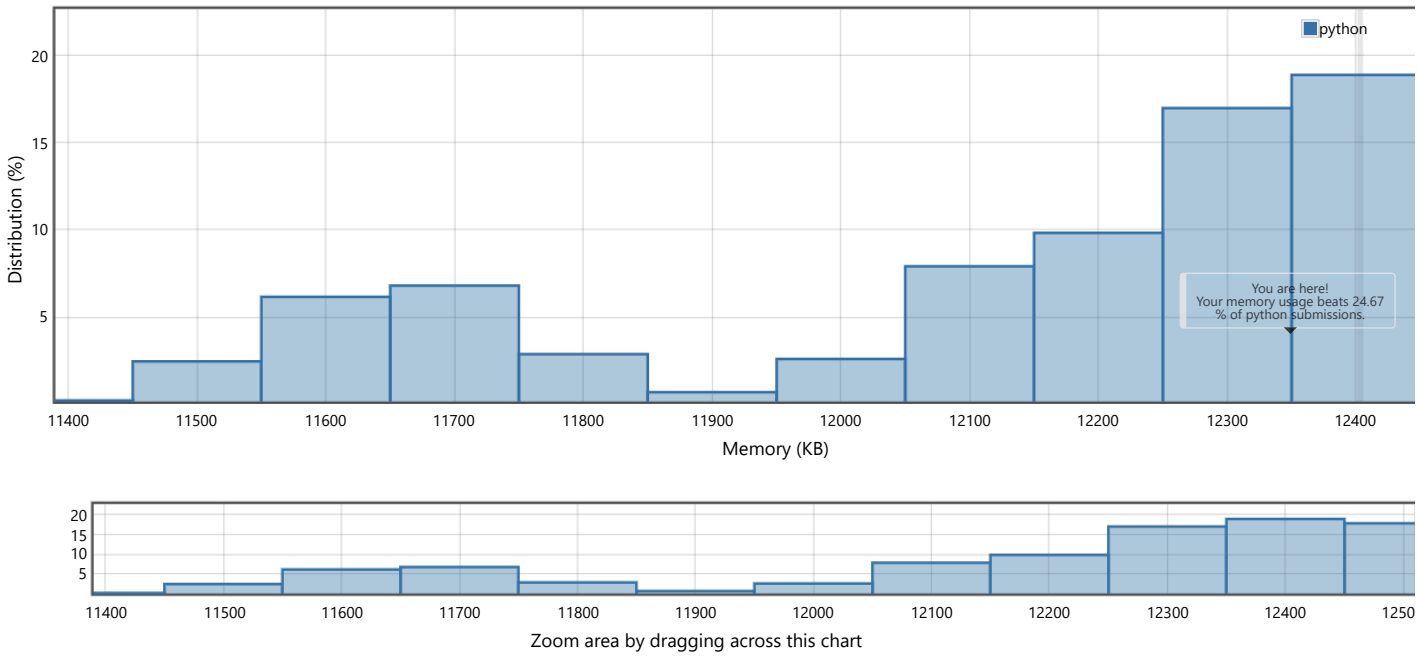
Memory Usage: **12.4 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Binary Tree Inorder Traversal**

Submitted Code: 1 day, 14 hours ago

Language: python

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
```

```python
#         self.left = left
#         self.right = right

class Solution(object):
    def inorderTraversal(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: List[int]
        """
        result = []
        stack = []
        current = root

        while current or stack:
            while current:
                stack.append(current)
                current = current.left
            current = stack.pop()
            result.append(current.val)
            current = current.right

        return result

# Example usage
# Creating a binary tree: [1, null, 2, 3]
root = TreeNode(1)
root.right = TreeNode(2)
root.right.left = TreeNode(3)

solution = Solution()
print(solution.inorderTraversal(root))  # Output: [1, 3, 2]
```

Back to problem (/problems/binary-tree-inorder-traversal/)

Find Peak Element (/problems/find-peak-element/)

Submission Detail

68 / 68 test cases passed.

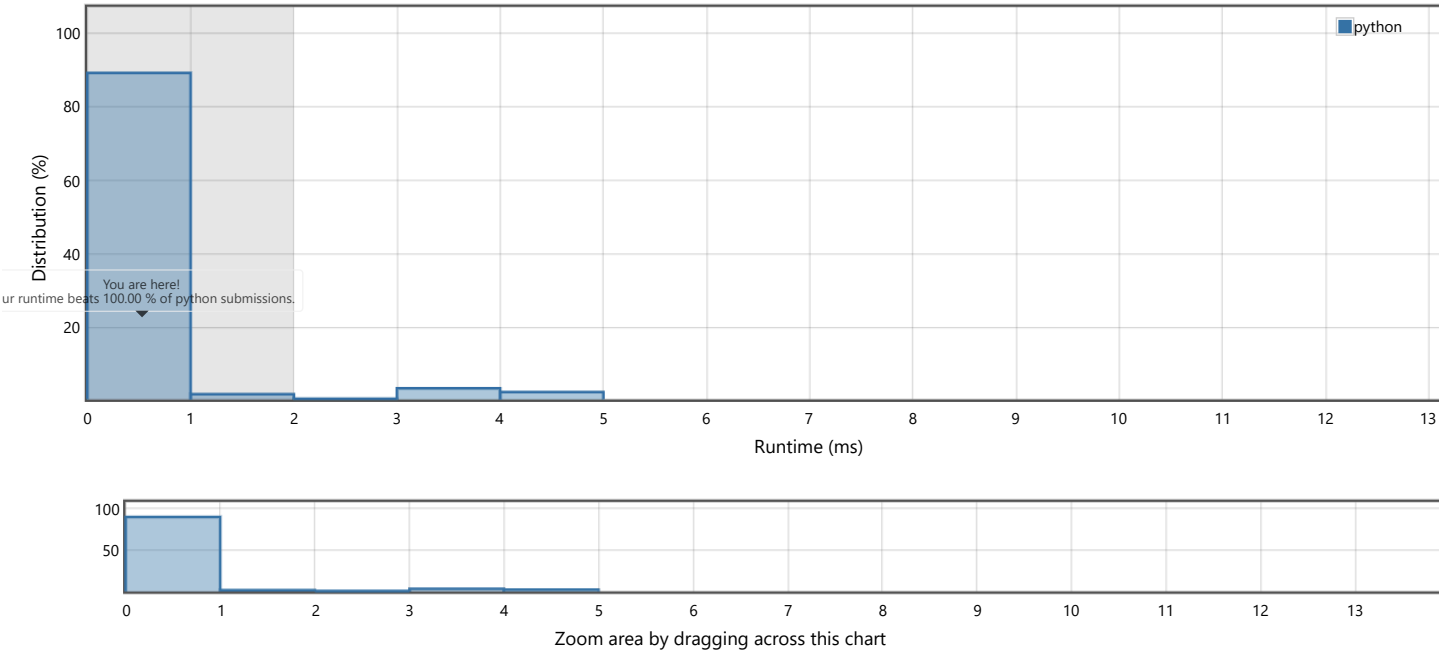
Status: **Accepted**

Runtime: **0 ms**

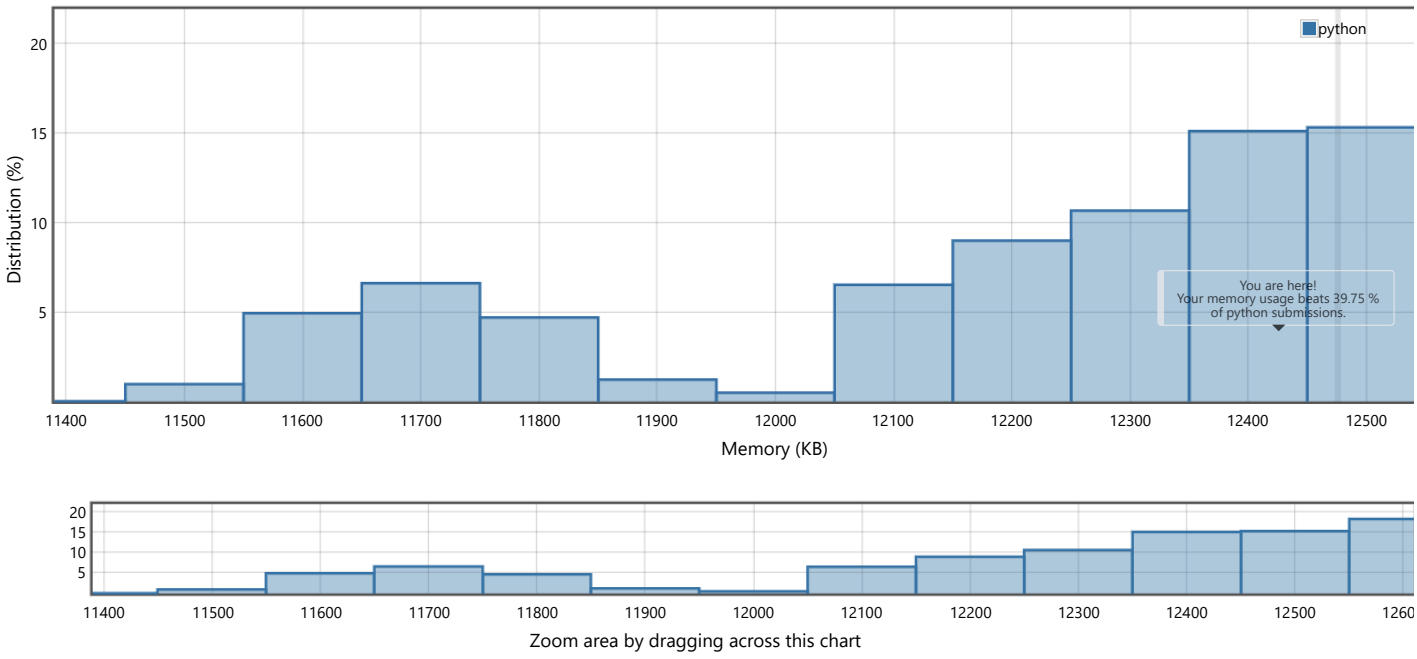
Memory Usage: **12.5 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Find Peak Element**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  class Solution(object):
2      def findPeakElement(self, nums):
3          """
4          :type nums: List[int]
```

```
        :rtype: int
        """
        left, right = 0, len(nums) - 1

        while left < right:
            mid = (left + right) // 2
            if nums[mid] > nums[mid + 1]:
                right = mid
            else:
                left = mid + 1

        return left

# Example usage
nums = [1, 2, 3, 1]
solution = Solution()
print(solution.findPeakElement(nums))  # Output: 2

```

Back to problem (/problems/find-peak-element/)

Remove Duplicates from Sorted List (/problems/remove-duplicates-from-sorted-list/)

Submission Detail

168 / 168 test cases passed.

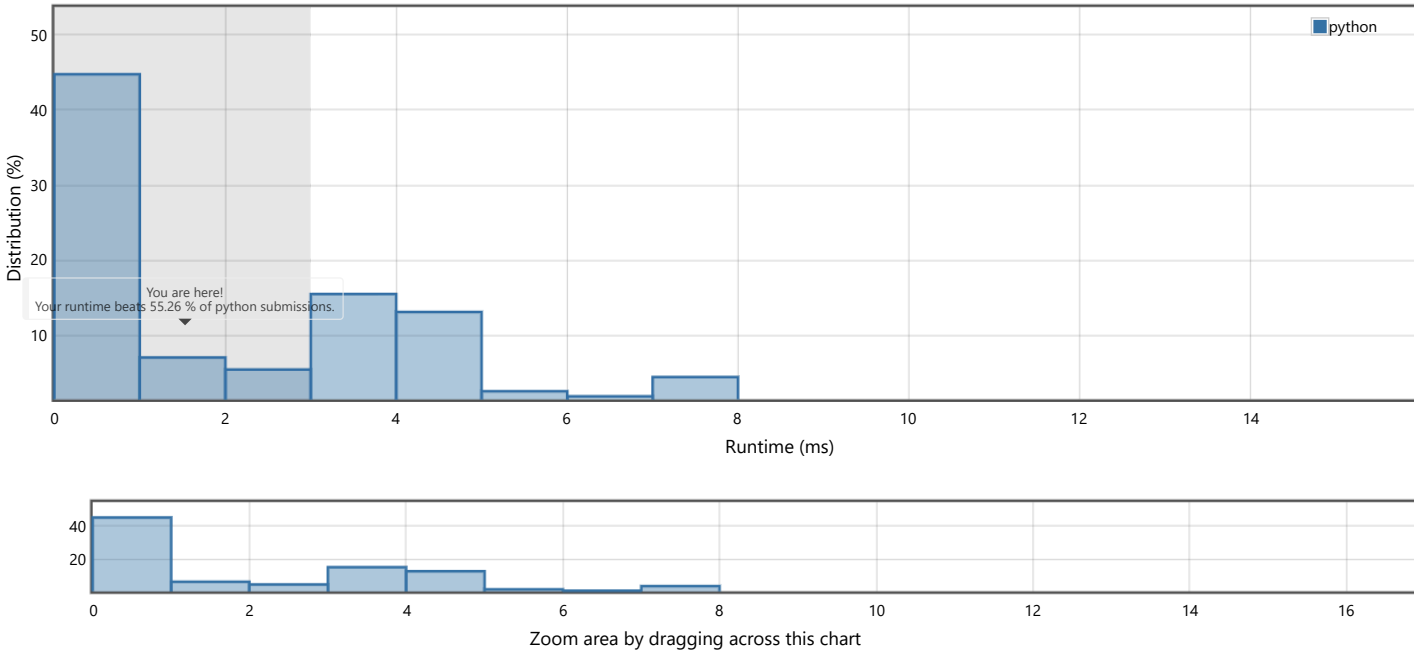
Status: Accepted

Runtime: **1 ms**

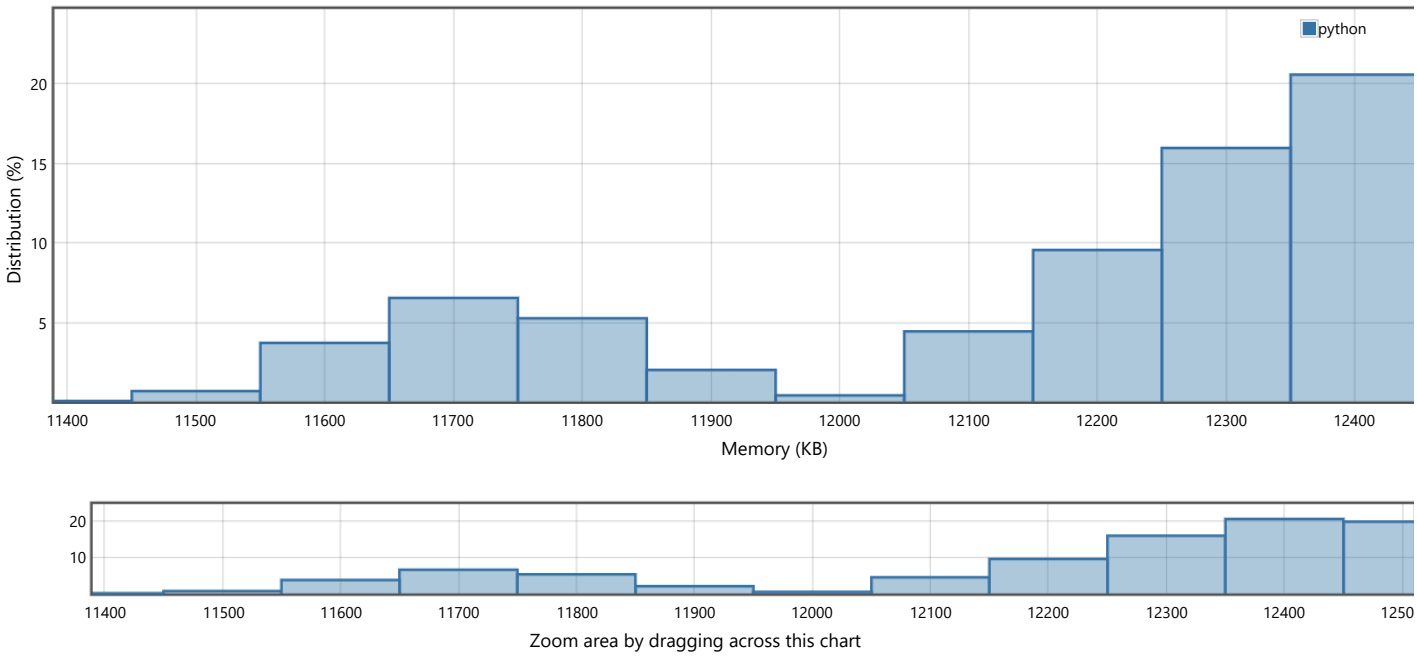
Memory Usage: **12.5 MB**

Submitted: **1 day, 14 hours ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Remove Duplicates from Sorted List**

Submitted Code: 1 day, 14 hours ago

Language: python

```
1  # Definition for singly-linked list.
2  # class ListNode(object):
3  #     def __init__(self, val=0, next=None):
4  #         self.val = val
```

```
#         self.next = next

class Solution(object):
    def deleteDuplicates(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        current = head

        while current and current.next:
            if current.val == current.next.val:
                current.next = current.next.next
            else:
                current = current.next

        return head

# Example usage
def printList(node):
    while node:
        print(node.val)
        node = node.next
    print("None")

# Creating a sorted linked list: 1 -> 1 -> 2 -> 3 -> 3
list1 = ListNode(1, ListNode(1, ListNode(2, ListNode(3, ListNode(3)))))
solution = Solution()
result = solution.deleteDuplicates(list1)
printList(result)  # Output: 1 -> 2 -> 3 -> None
```

Back to problem (/problems/remove-duplicates-from-sorted-list/)