CS566HW4-2024

September 25, 2024

0.1 HW 4. Introduction to algorithms

This is fourth Homework for CS 566.

0.2 Task 1. Solve the problem "3 Sum" from https://leetcode.com/problems/3sum/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

[7]:
```
from typing import List

class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
         # sort array, fix one element and find 2-sum, add triplet to result
        nums.sort()
        result = set()
        for i in range(len(nums)-2):
            # skip dupes
            if i>0 and nums[i] == nums[i - 1]:
                continue
            # two sum method: hashmap stores seen values, find difference␣
 ↪between fixed element and other element, if difference in hashmap then it is␣
 ↪the third element that makes the triplet for 3 sum
            target = -nums[i]
            hash_map = {}

            for j in range(i + 1, len(nums)):
                difference = target - nums[j]
                if difference in hash_map:
                    result.add((nums[i], difference, nums[j]))
                hash_map[nums[j]] = j

        return list(result)
```

0.2.1 Do not modify the testing code below. If you get message “Mistake in test case #”, it means that you algorithm is incorrect.

[8]:
```python
#test_case_1
expected, nums = set([(-1,0,1),(-1,-1,2)]), [-1,0,1,2,-1,-4]
actual = Solution().threeSum(nums)
actual = set([tuple(x) for x in actual])
assert expected==actual, "Mistake in test case 1"

#test_case_2
expected, nums = set([(-1,-1,2),(-1,0,1)]), [-1,0,1,2,-1,-4]
actual = Solution().threeSum(nums)
actual = set([tuple(x) for x in actual])
assert expected==actual, "Mistake in test case 2"

#test_case_3
expected, nums = set([(0,0,0)]), [0,0,0]
actual = Solution().threeSum(nums)
actual = set([tuple(x) for x in actual])
assert expected==actual, "Mistake in test case 3"
print('OK')
```

```
OK
```

0.2.2 Write analysis of the Memory Complexity and Time Complexity using Aymptotic Notation O. (1 point)

Memory Analysis: O(n)

Time Analysis: O(n^2) - Running two for loops

0.3 Task 2. Solve the problem “Merge Intervals” from https://leetcode.com/problems/merge-intervals/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

[9]:
```python
from typing import List
from collections import defaultdict

class Solution:
    def merge(self, intervals: List[List[int]]) -> List[List[int]]:
        # sort by start
        intervals.sort(key=lambda x:x[0])
        stack = [intervals[0]]
        for i in range(1,len(intervals)):
            top_stack = stack[-1] # top of stack is basically the previous␣
 ↪interval
            current = intervals[i]
            # if current start interval is less than top of stack (previous␣
↪interval's end interval), it means it overlaps and needs to be merged
            if current[0] <= top_stack[1]:
                # update top of stack's end interval
                top_stack[1] = max(top_stack[1], current[1])
            else:
                # add current to stack
                stack.append(current)
        return stack
```

[10]:
```
#test_case_1
expected, nums = [[1,6],[8,10],[15,18]], [[1,3],[2,6],[8,10],[15,18]]
actual = Solution().merge(nums)
assert expected==actual, "Mistake in test case 1"

#test_case_2
expected, nums =  [[1,5]], [[1,4],[4,5]]
actual = Solution().merge(nums)
assert expected==actual, "Mistake in test case 2"
print('OK')
```

```
OK
```

0.3.1 Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(n)

Time Analysis: O(nlogn)

0.4 Task 3. Solve the problem "Kth Largest Element In array" from https://leetcode.com/problems/kth-largest-element-in-an-array/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

[11]:
```
from typing import List
from heapq import heappush, heappop

class Solution:
    def findKthLargest(self, nums: List[int], k: int) -> int:
        # implement working algorithm
        # create heap
        max_heap = []

        for num in nums:
            heappush(max_heap,num)
            if len(max_heap)>k:
                heappop(max_heap)

        return max_heap[0]
```

[12]:
```
#test_case_1
expected, nums, k = 5, [3,2,1,5,6,4], 2
actual = Solution().findKthLargest(nums, k)
assert expected==actual, "Mistake in test case 1"

#test_case_2
expected, nums, k = 4, [3,2,3,1,2,4,5,5,6], 4
actual = Solution().findKthLargest(nums, k)
assert expected==actual, "Mistake in test case 2"
print('OK')
```

```
OK
```

0.4.1 Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(k)

Time Analysis: O(nlogk)