Lab 10. Introduction to algorithms

This is Tenth Lab for CS 566. This problem was given in lecture.

Task 1. Solve the problem "Container with Most Water" from https://leetcode.com/problems/container-with-most-water/description/ using Python3.

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

```
from typing import Optional, List

class Solution:
    def maxArea(self, height: List[int]) -> int:
        # using two pointer
        left, right = 0, len(height)-1
        area = 0
        while left<=right:
            # calculate area
            width = right - left
            curr_height = min(height[left], height[right])
            curr_area = width * curr_height
            area = max(area, curr_area)

            # move left pointer if its height is smaller, move right otherwise
            if height[left] < height[right]:
                left+=1
            else:
                right-=1
        return area
```

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

```
#test_case_1
height = [1,8,6,2,5,4,8,3,7]
expected = 49
actual = Solution().maxArea(height)
assert actual==expected, "Mistake in test case 1"

height = [1,1]
```

```
expected = 1
actual = Solution().maxArea(height)
assert actual==expected, "Mistake in test case 1"

print("OK")
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)

Task 2. Solve the problem "Jump Game" from https://leetcode.com/problems/jump-game/ using Python3.

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

```
import collections

from typing import Optional, List
class Solution:
    def canJump(self, nums: List[int]) -> bool:
        # greedy
        last_jump = len(nums) - 1
        for i in range(len(nums)-2, -1, -1):
            if i+nums[i]>=last_jump:
                last_jump = i

        return last_jump<=0
```

```
#test_case_1
nums = [2,3,1,1,4]
expected = True
actual = Solution().canJump(nums)
assert actual==expected, "Mistake in test case 1"

#test_case_2
nums = [3,2,1,0,4]
expected = False
actual = Solution().canJump(nums)
assert actual==expected, "Mistake in test case 2"

print('OK')
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)

Task 3. Solve the problem "Best Time To Buy and Sell With Transaction Fee" from https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/description/ using Python3.

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

```
class Solution:
    def maxProfit(self, prices: List[int], fee: int) -> int:
        dp1, dp2 = -prices[0],0
        for price in prices:
            dp1, dp2 = max(dp1, dp2-price), max(dp2, dp1 + price - fee)
        return dp2
```

```
#test_case_1
prices = [1,3,2,8,4,9]
fee = 2
expected = 8
actual = Solution().maxProfit(prices, fee)
assert actual==expected, "Mistake in test case 1"

prices = [1,3,7,5,10,3]
fee = 3
expected = 6
actual = Solution().maxProfit(prices, fee)
assert actual==expected, "Mistake in test case 2"

print('OK')
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)