Lab 8. Introduction to algorithms

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

Task 1. Solve the problem "House Robber" from https://leetcode.com/problems/house-robber/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 rob(self, nums: List[int]) -> int:
        h1, h2 = 0,0

        for num in nums:
            # print(h1+num,h2)
            temp_max = max(h1+num, h2)
            h1 = h2
            h2 = temp_max
        return h2
```

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
nums = [1,2,3,1]
expected = 4
actual = Solution().rob(nums)
assert actual==expected, "Mistake in test case 1"

nums = [2,7,9,3,1]
expected = 12
actual = Solution().rob(nums)
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 "Delete and Earn" from https://leetcode.com/problems/delete-and-earn/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)

```
import collections

class Solution(object):
    def deleteAndEarn(self, nums):
        # build hashmap of occurences first
        hashmap = {}
        for num in nums:
            if num in hashmap:
                hashmap[num] += 1
            else:
                hashmap[num] = 1
        # sort list by unique elements
        new_nums = sorted(list(set(nums)))
        earn1, earn2 = 0,0
        for i in range(len(new_nums)):
            curr_earn = new_nums[i] * hashmap[new_nums[i]]
            # print(curr_earn)
            if i>0 and new_nums[i] == new_nums[i-1] + 1:
                temp = earn2
                earn2 = max(curr_earn+earn1, earn2)
                earn1 = temp
            else:
                temp = earn2
                earn2 = max(curr_earn+earn1, curr_earn+earn2)
                earn1 = temp
        return earn2
```

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

```
#test_case_2
nums = [2,2,3,3,3,4]
expected = 9
actual = Solution().deleteAndEarn(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(n)

Time Analysis: O(n)

Task 3. Solve the problem "Longest Common Subsequence" from https://leetcode.com/problems/longest-common-subsequence/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 longestCommonSubsequence(self, text1: str, text2: str) -> int:
        n, m = len(text1), len(text2)
        dp = [[0]*(m+1) for i in range(n+1)]
        for i in range(n-1,-1,-1):
            for j in range(m-1,-1,-1):
                if text1[i] == text2[j]:
                    dp[i][j] = dp[i+1][j+1] + 1
                else:
                    dp[i][j] = max(dp[i][j+1],dp[i+1][j])

        # for n in dp:
            # print(n)


        return dp[0][0]
```

```
#test_case_1
text1 = "abcde"
text2 = "ace"
expected = 3
actual = Solution().longestCommonSubsequence(text1, text2)
```

```
assert actual==expected, "Mistake in test case 1"

text1 = "abc"
text2 = "def"
expected = 0
actual = Solution().longestCommonSubsequence(text1, text2)
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(n*m)

Time Analysis: O(n*m)