CS566HW5-2024

October 2, 2024

0.1 HW 5. Introduction to algorithms

This is fifth HW for CS 566.

0.2 Task 1. Solve the problem "Linked List Cycle II" from https://leetcode.com/problems/linked-list-cycle-ii/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)

[2]:
```python
# Definition for singly-linked list.
from typing import Optional

class ListNode:
    def __init__(self, x, next=None):
        self.val = x
        self.next = next

class Solution:
    def detectCycle(self, head: Optional[ListNode]) -> Optional[ListNode]:
        # add nodes to hashmap, if node exist in hashmap return it
        hash_map = {}
        result = None
        curr = head
        while curr:
            if curr in hash_map:
                return curr
            else:
                hash_map[curr] = 1
            curr = curr.next
        return 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.

[3]:
```
end_node = ListNode(-2)
head = ListNode(3, ListNode(2, ListNode(0, end_node)))
actual = Solution().detectCycle(head)
expected = None
assert actual==expected, "Mistake in test case 1"

end_node.next = head
actual = Solution().detectCycle(head)
expected = head
assert actual==expected, "Mistake in test case 2"
print('OK')
```

```
OK
```

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

Memory Analysis: O(n)

Time Analysis: O(n)

0.3 Task 2. Solve the problem "Remove N-th Node from Linked List" from https://leetcode.com/problems/remove-nth-node-from-end-of-list/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)

[4]:
```
from typing import Optional

class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def removeNthFromEnd(self, head: Optional[ListNode], n: int) ->␣
 ↪Optional[ListNode]:
        # get length of linked list
        length = 0
        curr = head
        while curr:
            curr = curr.next
            length+=1
        # if we're deleting the head, then we need to update head node
        if length == n:
```

```
            head = head.next
            return head

        # iterate until right before the nth node
        i = length
        curr2 = head
        while curr2:
            # point node right before nth node to the one after nth node
            if i == n+1:
                if not curr2.next.next:
                    curr2.next = None
                else:
                    curr2.next = curr2.next.next
            i -= 1
            curr2 = curr2.next

        return head
```

[5]:
```
#test_case_1
head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))
new_head = Solution().removeNthFromEnd(head, 2)
actual = new_head.next.next.next.val
expected = 5
assert actual==expected, "Mistake in test case 1"
print("OK")
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)

0.4 Task 3. Solve the problem "Number Of Islands" from https://leetcode.com/problems/number-of-islands/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)

[6]:
```
from typing import List

class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        """ helper function that performs dfs, marks adjacent islands """
        def dfs(grid,i,j):
```

```
            if i<0 or j<0:
                return
            if i>=len(grid) or j>=len(grid[0]):
                return
            if grid[i][j]!='1':
                return

            grid[i][j] = 0
            dfs(grid,i+1,j)
            dfs(grid,i-1,j)
            dfs(grid,i,j+1)
            dfs(grid,i,j-1)

        if not grid:
            return 0
        count = 0
        rows, cols = len(grid), len(grid[0])
        for i in range(rows):
            for j in range(cols):
                if grid[i][j] == '1':
                    dfs(grid,i,j)
                    count+=1
        return count
```

[7]:
```
grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
expected = 1
actual = Solution().numIslands(grid)
assert expected == actual, "Mistake in test case 1"


grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
expected = 3
actual = Solution().numIslands(grid)
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(n*m) - where n is number of rows in grid and m is number of columns in grid

Time Analysis: O(n*m) - where n is number of rows in grid and m is number of columns in grid