CS566Lab5-2024

October 2, 2024

0.1 Lab 5. Introduction to algorithms

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

0.2 Task 1. Solve the problem "Min Stack" from https://leetcode.com/problems/min-stack/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]:
```
class MinStack(object):

    def __init__(self):
        self.stack = []
        self.min_stack = []

    def push(self, val):
        """
        :type val: int
        :rtype: None
        """
        if not self.min_stack:
            self.min_stack.append(val)
            self.stack.append(val)
            return

        self.min_stack.append(min(val, self.min_stack[-1]))
        self.stack.append(val)


    def pop(self):
        """
        :rtype: None
        """
        top = self.stack.pop()
        self.min_stack.pop()
    def top(self):
        """
        :rtype: int
        """
        return self.stack[-1]


    def getMin(self):
        """
        :rtype: int
        """
        return self.min_stack[-1]



# Your MinStack object will be instantiated and called as such:
# obj = MinStack()
# obj.push(val)
# obj.pop()
# param_3 = obj.top()
# param_4 = obj.getMin()
```

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]:
```
#test_case_1
minStack = MinStack()
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
actual = minStack.getMin();
expected = -3
assert expected==actual, "Mistake in test case 1"
minStack.pop();
actual = minStack.top();
expected = 0
assert expected==actual, "Mistake in test case 2"
actual = minStack.getMin();
expected = -2
assert expected==actual, "Mistake in test case 2"
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(1) - Amortized

0.3 Task 2. Solve the problem "Implement Queue Using Stacks" from https://leetcode.com/problems/implement-queue-using-stacks/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]:
```
class MyQueue(object):

    def __init__(self):
        self.s1 = []
        self.s2 = []

    def push(self, x):
        """
        :type x: int
        :rtype: None
        """
        self.s1.append(x)


    def pop(self):
        """
        :rtype: int
        """
        if not self.s2:
            while self.s1:
                self.s2.append(self.s1.pop())
        return self.s2.pop()

    def peek(self):
        """
        :rtype: int
        """
        if not self.s2:
            while self.s1:
                self.s2.append(self.s1.pop())
        return self.s2[len(self.s2)-1]

    def empty(self):
        """
        :rtype: bool
        """
        return 0 == max(len(self.s1),len(self.s2))


# Your MyQueue object will be instantiated and called as such:
# obj = MyQueue()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.peek()
# param_4 = obj.empty()
```

[5]:
```
#test_case_1
myQueue = MyQueue();
myQueue.push(1);
myQueue.push(2);
actual = myQueue.peek();
expected = 1
assert expected==actual, "Mistake in test case 1"
actual = myQueue.pop();
assert expected==actual, "Mistake in test case 2"
actual = myQueue.empty();
expected = False
assert expected==actual, "Mistake in test case 3"

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(1) - Amortized

0.4 Task 3. Solve the problem "Reverse Linked List" from https://leetcode.com/problems/reverse-linked-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)

[6]:
```
# Definition for singly-linked list.
from typing import Optional
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

```
class Solution:
    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        prev = None
        current = head

        while current:
            next_node = current.next
            current.next = prev
            prev = current
            current = next_node

        return prev
```

[7]:
```
ls = ListNode(0, ListNode(1, ListNode(2, ListNode(3))))
actual = Solution().reverseList(ls)
expected = 3
assert actual.val==expected, "Mistake in test case 1"
print("OK")
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)