Polytechnic Tutoring Center

Exam 2 Review Answer Key-CS 1134,Fall 2021

Disclaimer: This mock exam is only for practice. It was made by tutors in the Polytechnic Tutoring Center and is not representative of the actual exam given by the CS Department.

1. Given an unmodified basic implementation of a stack, singly/doubly linked list, queue, dequeue what are the runtimes for the following:

- Find the minimum value in a stack of ints: O(n)
- Insert at front of a singly linked list: O(1)
- Push onto a stack: O(1)
- Remove the last element from a singly linked list: O(n)
- Insert into middle of a doubly linked list: O(n)

2. Assume that the function takes in the front node of the list, a sample node class is defined below. (assume that the first node has valid data and is not an empty header node)

Write a function that takes a singly linked list and recursively prints it out in reverse.

```
class Node(object):
    def __init__(self, data=None, next_node=None):

        self.data = data

        self.next_node = next_node
```

Code:

```
def printReverse(Node current):

    if current == None:

        return

   printReverse(current.next_node)

   print(current.data)
```

3 Circle the faster runtime:

O(log(log(n)) or O(nlog(n))

O(nlogn) or O(n^1.25)

O(2^n) or O(n!)

4 Given a string with an undefined number of open or closed parentheses and braces: ([and]),

determine if the parentheses are balanced “()[([])]” is balanced.

“[()(])]” is NOT balanced.

You may assume you have predefined implementations of an array, stack, queue, and dequeue. You may assume the string passed as a parameter will only consist of ‘(‘, ’)’, ’[‘, ’]’ characters.

Code:

```
def balance(stringy):

    stk = Stack()

    for elem in stringy:
        if elem == ‘(‘ or elem == ‘[‘:

            stk.push(elem)

        else:

            if elem == ‘)’ or elem == ‘]’:

                 if len(stk) == 0:

                    return False

                if elem == ‘)’ and stk.top() == ‘(‘:

                    stk.pop()
                elif elem == ‘]’ and stk.top() == ‘[‘:

                    stk.pop()

                else:

                    return False

    if len(stk) > 0:

        return False

    return True
```

5. Given a dequeue of characters, write a function that determines if the dequeue currently holds a palindrome. (you may modify the contents of the deque). Assume that a predefined implementation of a dequeue has been provided with the following methods:

- deq.push_front(element):
- deq.push_back(element):
- deq.pop_front()
- deq.pop_back()
- deq.front()
- deq.back()
- deq.__len__()

inserts an element to the front of the dequeue inserts an element to the back of the dequeue removes the front of the dequeue
removes the back of the dequeue

returns the front element from the dequeue returns the back element from the dequeue returns the number of elements in the dequeue

A palindrome is a string that is identical if read from front to back or from back to front. For example: “racecar” is a palindrome

Code:

```
def palindrome(deq):
    while len(deq) > 1:
        if deq.front() != deq.back():
            return False
            deq.pop_back()
    deq.pop_back() return True
```

6. Given a node in a singly linked list, write a function to remove all subsequent instances of a single number passed as a parameter. Assume the list has at least one element

```
class Node(object):
def __init__(self, data=None, next_node=None):

    self.data = data

    self.next_node = next_node
```

Code:

```
def removeNum(listNode, removal):

    current = listNode

    while current != None and current.next_node != None:

        if current.next_node.data == removal:

            current.next_node = current.next_node.next_node

        else:

            current = current.next_node

    if listNode.data == removal: # checked the first node now

        listNode = listNode.next_node

    return listNode
```

7. a/ Write a function to check if two binary trees are identical (same structure and content). Return True if identical and False otherwise

```
def isIdentical(node1, node2):

    # base case
    if node1 is None and node2 is None:
        return True

    return (node1 and node2) and (node1.key == node2.key) and
        isIdentical(node1.left, node2.left) and isIdentical(node1.right, node2.right)
```

b/ Write a function to invert a binary tree

```
def swap(root):
    # base case
    if root is None:
        return

    temp = root.left
    root.left = root.right
    root.right = temp


def invertBinaryTree(root):

    # base case
    if root is None:
        return

    # swap left subtree with right subtree
    swap(root)

    # invert left subtree
    invertBinaryTree(root.left)

    # invert right subtree
    invertBinaryTree(root.right)
```