You can view this report online at : https://www.hackerrank.com/x/tests/1121388/candidates/26930304/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Review Week 3 Assessment 2021**
Taken On:	21 Jun 2021 21:51:42 PDT
Time Taken:	85 min 7 sec/ 90 min
Work Experience:	3 years
Contact Number:	+14084096862
Resume:	https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCAThWIiNY1UGAfhwRPsrmVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF
Linkedin:	https://www.linkedin.com/in/my-nguyen-87849
Invited by:	Curriculum
Skills Score:	
Tags Score:	

100%

475/475

scored in **Review Week 3 Assessment 2021** in 85 min 7 sec on 21 Jun 2021 21:51:42 PDT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	Runtime Analysis 1 > **Multiple Choice**	57 sec	5/ 5	✓
Q2	Runtime Analysis 2 > **Multiple Choice**	41 sec	5/ 5	✓
Q3	Runtime Analysis 3 > **Multiple Choice**	59 sec	5/ 5	✓
Q4	Runtime Analysis 4 > **Multiple Choice**	33 sec	5/ 5	✓
Q5	Runtime Analysis 5 > **Multiple Choice**	1 min 1 sec	5/ 5	✓
Q6	Insertion Sort List > **Coding**	35 min 50 sec	150/ 150	!
Q7	132 Pattern > **Coding**	8 min 16 sec	150/ 150	!

QUESTION 1

Correct Answer

Score 5

Runtime Analysis 1 > Multiple Choice

QUESTION DESCRIPTION

What is the runtime of this code snippet? `a` is the size of list a, while `b` is the size of list b.

Java:

```
void print_pairs(ArrayList<Integer> a, ArrayList<Integer> b) {
   for (int i = 0; i < a.size(); i++) {
      for (int j = 0; j < b.size(); j++) {
         for (int k = 0; k < 10000; k++) {
            System.out.print(i + " " + j);
         }
      }
   }
}
```

Python:

```
def print_pairs(list_a, list_b):
   for i in list_a:
       for j in list_b:
           for k in range(10000):
               print i, j
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- O(a)
- O(b)
- O(a + b)
- ✓ O(a * b)

No Comments

QUESTION 2

Correct Answer

Score 5

Runtime Analysis 2 › Multiple Choice

QUESTION DESCRIPTION

The following code computes the integer square root of a number. If the number is not a perfect square, then it returns `a - 1`.

What is its run time? n represents the size of the input, `n`.

Java:

```
int get_square_root(int n) {
    return square_root_helper(n, 1, n);
}

int square_root_helper(int n, int min, int max) {
    if (max < min) {
        return -1;
    }

    int guess = (min + max) / 2;
    if (guess * guess == n) {
        return guess;
    } else if (guess * guess < n) {
        return square_root_helper(n, guess + 1, max);
    } else {
        return square_root_helper(n, min, guess - 1);
    }
}
```

Python:

```
def get_square_root(n):
   return square_root_helper(n, 1, n)

def square_root_helper(n, min, max):
   if max < min:
       return -1
   guess = (min + max) // 2
   if guess * guess == n:
       return guess
   elif guess * guess < n:
       return square_root_helper(n, guess + 1, max)
   else:
       return square_root_helper(n, min, guess - 1)
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- ✓ O(log n)
- O(n)
- O(n^2)
- O(2^n)

No Comments

QUESTION 3

Correct Answer

Score 5

Runtime Analysis 3 › Multiple Choice

QUESTION DESCRIPTION

The following code reverses a list.

What is its runtime? n represents the size of the `input_list`.

Java:

```
void reverseList(ArrayList<Integer> input_list) {
        int input_list_len = input_list.size();
        for (int i = 0; i < input_list_len / 2; i++) {
            int other_index = input_list_len - i - 1;
            int curr = input_list.get(i);
            input_list.set(i, input_list.get(other_index));
            input_list.set(other_index, curr);
        }
    }
```

Python:

```
def reverse_list(input_list):
   input_list_len = len(input_list)
   for i in range(input_list_len // 2):
       other_index = input_list_len - i - 1
       curr = input_list[i]
       input_list[i] = input_list[other_index]
       input_list[other_index] = curr
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- O(log n)
- ✓ O(n)
- O(n^2)
- O(2^n)

No Comments

QUESTION 4

Correct Answer

Score 5

Runtime Analysis 4 › Multiple Choice

QUESTION DESCRIPTION

The following code all the fibonacci numbers up to n. What is the runtime? n represents the size of the input to `all_fib`.

Java:

```
void all_fib(int n) {
    for (int i = 0; i < n; i++) {
        System.out.print(fib(i));
    }
}

int fib(int n) {
    if (n < 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }

    return fib(n-1) + fib(n -2);
}
```

Python:

```
def all_fib(n):
   for i in range(n):
       print fib(i)

def fib(n):
   if n < 0:
       return 0
   elif n == 1:
       return 1
   return fib(n - 1) + fib(n - 2)
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(log n)
- O(n)
- O(n^2)
- ✓ O(2^n)
- O(n * 2^n)

No Comments

QUESTION 5

Correct Answer

Score 5

Runtime Analysis 5 › Multiple Choice

QUESTION DESCRIPTION

The following code inefficiently finds the running min at each index of the array and returns a mapping of array index to the minimum element up to that index. What is the runtime of this code snippet? n represents the size of the `input_list`.

Java:

```
ArrayList<Integer> get_running_min(ArrayList<Integer> input_list) {
    ArrayList<Integer> result = new ArrayList<>();
    for (int i = 0; i < input_list.size(); i++) {
        result.set(i, get_min_helper(i, input_list));
    }
    return result;
}

int get_min_helper(int index, ArrayList<Integer> list) {
    Integer curr_min = null;
    for (int i = 0; i < index + 1; i++) {
        if (curr_min == null || list.get(i) < curr_min) {
            curr_min = list.get(i);
        }
    }
    return curr_min;
}
```

```
def get_running_min(input_list):
   result = {}
   for i in range(len(input_list)):
       result[i] = get_min_helper(i, input_list)
   return result

def get_min_helper(index, list):
   curr_min = None
   for i in range(index + 1):
       if not curr_min or list[i] < curr_min:
           curr_min = list[i]
   return curr_min
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- O(log n)
- O(n)
- ✓ O(n^2)
- O(2^n)

No Comments

QUESTION 6

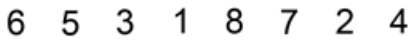
Insertion Sort List › Coding

Needs Review

Score 150

QUESTION DESCRIPTION

Sort a linked list using insertion sort.



A graphical example of insertion sort. The partial sorted list (black) initially contains only the first element in the list.
With each iteration one element (red) is removed from the input data and inserted in-place into the sorted list

Algorithm of Insertion Sort:

1. Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list.
2. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there.
3. It repeats until no input elements remain.

Example 1:

```
Input: 4->2->1->3
Output: 1->2->3->4
```

Example 2:

```
Input: -1->5->3->4->0
Output: -1->0->3->4->5
```

CANDIDATE ANSWER

Language used: **Java 7**

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public static ListNode insertionSortList(ListNode head) {
    ListNode dummy = new ListNode(-1);
    ListNode tmp = head;
    while (tmp != null) {
        ListNode current = dummy;
        while (current.next != null && current.next.val < tmp.val) {
            current = current.next;
        }
        ListNode tmp2 = tmp.next;
        tmp.next = current.next;
        current.next = tmp;
        // System.out.println("current: " + current.val + ", head: " +
head.val + ", next: " + current.next.val);
```

```
head.val + ", next: " + current.next.val);
                tmp = tmp2;
            }
            return dummy.next;
        }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0754 sec	23.8 KB
Testcase 1	Easy	Sample case	Success	10	0.0704 sec	23.8 KB
Testcase 2	Easy	Hidden case	Success	10	0.0801 sec	23.8 KB
Testcase 3	Easy	Hidden case	Success	10	0.0683 sec	23.8 KB
Testcase 4	Easy	Hidden case	Success	10	0.0745 sec	23.7 KB
Testcase 5	Easy	Hidden case	Success	10	0.0806 sec	23.7 KB
Testcase 6	Easy	Hidden case	Success	10	0.0732 sec	23.8 KB
Testcase 7	Easy	Hidden case	Success	10	0.0837 sec	23.8 KB
Testcase 8	Easy	Hidden case	Success	10	0.073 sec	23.8 KB
Testcase 9	Easy	Hidden case	Success	10	0.0695 sec	23.7 KB
Testcase 10	Easy	Hidden case	Success	10	0.0662 sec	23.8 KB
Testcase 11	Easy	Hidden case	Success	10	0.0741 sec	23.8 KB
Testcase 12	Easy	Hidden case	Success	10	0.0697 sec	23.7 KB
Testcase 13	Easy	Hidden case	Success	10	0.0694 sec	23.8 KB
Testcase 14	Easy	Hidden case	Success	10	0.0751 sec	23.6 KB

No Comments

QUESTION 7

Needs Review

Score 150

132 Pattern › Coding

QUESTION DESCRIPTION

Given a sequence of n integers a_1, a_2, ..., a_n, a 132 pattern is a subsequence a_i, a_j, a_k such that $i < j < k$ and $a_i < a_k < a_j$. Design an algorithm that takes a list of n numbers as input and checks whether there is a 132 pattern in the list.
Note: n will be less than 15,000.

Example 1:

```
Input: [1, 2, 3, 4]

Output: False

Explanation: There is no 132 pattern in the sequence.
```

Example 2:

```
Input: [3, 1, 4, 2]

Output: True

Explanation: There is a 132 pattern in the sequence: [1, 4, 2].
```

Example 3:

```
Input: [-1, 3, 2, 0]

Output: True

Explanation: There are three 132 patterns in the sequence: [-1, 3, 2],
[-1, 3, 0] and [-1, 2, 0].
```

CANDIDATE ANSWER

Language used: **Java 7**

```
1
2    public static boolean find132pattern(int[] nums) {
3        if (nums.length < 3)
4            return false;
5
6        for (int i = 0; i < nums.length - 2; i++) {
7            for (int j = i + 1; j < nums.length - 1; j++) {
8                for (int k = j + 1; k < nums.length; k++) {
9                    if (nums[i] < nums[k] && nums[k] < nums[j])
10                       return true;
11               }
12           }
13       }
14       return false;
15   }
16
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.1284 sec	23.7 KB
Testcase 1	Easy	Sample case	Success	10	0.0777 sec	23.6 KB
Testcase 2	Easy	Sample case	Success	10	0.0667 sec	23.4 KB
Testcase 3	Easy	Hidden case	Success	10	0.0722 sec	23.7 KB
Testcase 4	Easy	Hidden case	Success	10	0.0758 sec	23.5 KB
Testcase 5	Easy	Hidden case	Success	10	0.0753 sec	23.5 KB
Testcase 6	Easy	Hidden case	Success	10	0.1015 sec	24.3 KB
Testcase 7	Easy	Hidden case	Success	10	0.0766 sec	23.4 KB
Testcase 8	Easy	Hidden case	Success	10	0.0869 sec	23.8 KB
Testcase 9	Easy	Hidden case	Success	10	0.0679 sec	23.6 KB
Testcase 10	Easy	Hidden case	Success	10	0.0673 sec	23.7 KB
Testcase 11	Easy	Hidden case	Success	10	0.0721 sec	23.7 KB
Testcase 12	Easy	Hidden case	Success	10	0.0721 sec	23.7 KB
Testcase 13	Easy	Hidden case	Success	10	0.0847 sec	23.7 KB
Testcase 14	Easy	Hidden case	Success	10	0.0695 sec	23.8 KB

No Comments

QUESTION 8

Correct Answer

Score 150

Sliding Window Maximum › Coding

QUESTION DESCRIPTION

Given an array *nums*, there is a sliding window of size *k* which is moving from the very left of the array to the very right. You can only see the *k* numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window.

Example:

```
Input: nums = [1,3,-1,-3,5,3,6,7], and k = 3
Output: [3,3,5,5,6,7]
Explanation:

Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

Note:

You may assume *k* is always valid, 1 ≤ k ≤ input array's size for non-empty array.

CANDIDATE ANSWER

Language used: **Java 7**

```
1     public static int[] maxSlidingWindow(int[] nums, int k) {
2         Queue<Integer> maxHeap = new PriorityQueue(k,
3 Collections.reverseOrder());
4         if (nums.length < k) {
5             for (int num : nums) {
6                 maxHeap.offer(num);
7             }
8             return new int[] { maxHeap.peek() };
9         }
10
11        for (int i = 0; i < k; i++) {
12            maxHeap.offer(nums[i]);
13        }
14
15        int[] max = new int[nums.length - k + 1];
16        max[0] = maxHeap.peek();
17        for (int i = k; i < nums.length; i++) {
18            maxHeap.remove(nums[i-k]);
19            maxHeap.offer(nums[i]);
20            max[i-k+1] = maxHeap.peek();
21        }
22        return max;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0732 sec	23.8 KB
Testcase 1	Easy	Hidden case	Success	10	0.0727 sec	23.8 KB
Testcase 2	Easy	Hidden case	Success	10	0.0729 sec	23.8 KB
Testcase 3	Easy	Hidden case	Success	10	0.0779 sec	23.8 KB

Testcase 4	Easy	Hidden case	Success	10	0.0684 sec	23.8 KB
Testcase 5	Easy	Hidden case	Success	10	0.0728 sec	23.9 KB
Testcase 6	Easy	Hidden case	Success	10	0.0852 sec	23.8 KB
Testcase 7	Easy	Hidden case	Success	10	0.0712 sec	23.6 KB
Testcase 8	Easy	Hidden case	Success	10	0.0772 sec	23.5 KB
Testcase 9	Easy	Hidden case	Success	10	0.0746 sec	23.7 KB
Testcase 10	Easy	Hidden case	Success	10	0.0681 sec	23.7 KB
Testcase 11	Easy	Hidden case	Success	10	0.0711 sec	23.8 KB
Testcase 12	Easy	Hidden case	Success	10	0.0773 sec	23.8 KB
Testcase 13	Easy	Hidden case	Success	10	0.0681 sec	23.8 KB
Testcase 14	Easy	Hidden case	Success	10	0.0851 sec	23.8 KB

No Comments

PDF generated at: 22 Jun 2021 06:22:28 UTC