You can view this report online at : https://www.hackerrank.com/x/tests/1330881/candidates/42556830/report

Full Name:	Ruchit Bhardwaj
Email:	ruchitbh@usc.edu
Test Name:	**CodePath SE103: Unit 8 Assessment - Summer 2022**
Taken On:	26 Jul 2022 17:31:03 PDT
Time Taken:	79 min 19 sec/ 90 min
Invited by:	CodePath
Skills Score:	
Tags Score:	

98.3%

570/580

scored in **CodePath SE103: Unit 8 Assessment - Summer 2022** in 79 min 19 sec on 26 Jul 2022 17:31:03 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Sorting Algorithm Modification > **Multiple Choice**	1 min 51 sec	5/ 5	✓
Q2	Sorting Algorithm Efficiency > **Multiple Choice**	46 sec	5/ 5	✓
Q3	Runtime Analysis > **Multiple Choice**	53 sec	0/ 5	✗
Q4	Add Intervals Output > **Multiple Choice**	39 sec	5/ 5	✓
Q5	Add Intervals Debugging > **Coding**	35 min 33 sec	100/ 100	✓
Q6	Add Intervals Space Complexity > **Multiple Choice**	52 sec	5/ 5	✓
Q7	Add Intervals Time Complexity > **Multiple Choice**	1 min 31 sec	0/ 5	✗
Q8	3Sum Closest > **Coding**	13 min 52 sec	150/ 150	✓
Q9	Rotate Image > **Coding**	9 min 50 sec	150/ 150	✓
Q10	Minimum Window Substring > **Coding**	13 min 12 sec	150/ 150	✓

QUESTION 1

✓ Correct Answer

Score 5

Sorting Algorithm Modification › Multiple Choice

QUESTION DESCRIPTION

Given an unsorted array. The array has this property that every element in array is at most k distance from its position in sorted array where k is a positive integer smaller than size of array. Which sorting algorithm can be most easily modified for sorting this array and what is the obtainable time complexity?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ Quick sort
- ✓ ● Heap sort
- ○ Merge sort
- ○ Insertion sort

No Comments

QUESTION 2

✓ Correct Answer

Score 5

Sorting Algorithm Efficiency › Multiple Choice

QUESTION DESCRIPTION

What sorting algorithm is most efficient when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced)?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ Quick sort
- ○ Heap sort
- ○ Merge sort
- ✓ ● Insertion sort

No Comments

QUESTION 3

Wrong Answer

Score 0

Runtime Analysis › Multiple Choice

QUESTION DESCRIPTION

What is the worst possible run time of this code? N refers to the size of the array “nums” and you can assume nums will be a sorted array.

Java:

```
int num_occurences(ArrayList<Integer> nums, int x, int start, int end) {
    if (start > end) {
        return 0;
    }

    int mid = (start + end) / 2;

    if (nums.get(mid) < x) {
        return num_occurences(nums, x, mid + 1, end);
    }

    if (nums.get(mid) > x) {
        return num_occurences(nums, x, start, mid - 1);
    }

    return num_occurences(nums, x, start, mid - 1) + 1 +
num_occurences(nums, x, mid + 1, end);
}
```

Python:

```
def num_occurrences(nums, x, start, end):
   if start > end:
       return 0
   mid = (start + end) // 2
   if nums[mid] < x:
       return num_occurrences(nums, x, mid + 1, end)
   if nums[mid] > x:
       return num_occurrences(nums, x, start, mid - 1)
   return num_occurrences(nums, x, start, mid - 1) + 1 +
num_occurrences(nums, x, mid + 1, end)
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- [● O(log n)]
- ✓ O(n)
- O(n log n)
- O(n^2)

No Comments

QUESTION 4

✓ Correct Answer

Score 5

Add Intervals Output › Multiple Choice

QUESTION DESCRIPTION

Add a new interval into a set of non-overlapping intervals, merging if necessary.

The intervals given will be sorted according to their start times.

Example-
Input: intervals = [[1,5],[6,12], [14, 15]], new_interval = [3,6]
Output: [[1, 12], [14, 15]]

Given the problem statement, what is the expected output for this input?
`add_intervals([[1,2],[3,4],[6,7],[8,10],[11,17]], [4,8])`

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [[1,2], [3, 8], [8,10], [11,17]]
- [[1,2], [3,4], [4, 8], [8,10], [11,17]]
- [[1,2], [3, 4], [4,10], [11,17]]
- [✓ [[1,2] ,[3, 10], [11,17]]]

No Comments

QUESTION 5

✓ Correct Answer

Score 100

Add Intervals Debugging › Coding

QUESTION DESCRIPTION

Add a new interval into a set of non-overlapping intervals, merging if necessary.

The intervals given will be sorted according to their start times.

Example-
Input: intervals = [[1,5], [6,12], [14, 15]], new_interval = [3,6]
Output: [[1, 12], [14, 15]]

Please fix the buggy solution below.

CANDIDATE ANSWER

Language used: **Java 8**

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Interval {
    public int start;
    public int end;
    public Interval(int startI, int endI) {
        start = startI;
        end = endI;
    }

    public String toString() {
        return String.format("[%s, %s]", start, end);
    }
}

public class Solution {
    /**
     * Definition for an interval.
     * public class Interval {
     *     int start;
     *     int end;
     *     Interval(int s, int e) { start = s; end = e; }
     * }
     */
    public static ArrayList<Interval> addInterval(ArrayList<Interval>
intervals, Interval newInterval) {
        ArrayList<Interval> result = new ArrayList<Interval>();
        for(Interval interval : intervals) {
            // System.out.println(String.format("Interval [%s, %s]",
interval.start, interval.end));
            // System.out.println(String.format("New Interval [%s, %s]",
newInterval.start, newInterval.end));
            // System.out.println(String.format("Result %s", result));
            if (interval.end < newInterval.start){
                result.add(interval);
            } else if (interval.start > newInterval.end) {
                result.add(interval);
            } else if (interval.end >= newInterval.start || interval.start <=
newInterval.start) {
                newInterval = new Interval(Math.min(interval.start,
newInterval.start), Math.max(newInterval.end, interval.end));
            }
        }

        result.add(newInterval);
        Collections.sort(result, (a, b) -> a.start - b.start);

        return result;
    }
    private static final Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) throws IOException {
        int numIntervals = Integer.parseInt(scanner.nextLine().trim());
        ArrayList<Interval> intervals = new ArrayList<Interval>();
        for (int i = 0; i < numIntervals - 1; i++) {
            String allNums = scanner.nextLine().trim();
            String[] splitNums = allNums.split("\\s+");
            Interval interval = new Interval(Integer.parseInt(splitNums[0]),
Integer.parseInt(splitNums[1]));
            intervals.add(interval);
        }

        String allNums = scanner.nextLine().trim();
        String[] splitNums = allNums.split("\\s+");
        Interval newInterval = new Interval(Integer.parseInt(splitNums[0]),
Integer.parseInt(splitNums[1]));

        ArrayList<Interval> newIntervals = addInterval(intervals,
newInterval);
        System.out.print("[");
        for (int i = 0; i < newIntervals.size() - 1; i++) {
                System.out.print("[" + newIntervals.get(i).start + ", " +
newIntervals.get(i).end + "], ");
            }
            System.out.print("[" + newIntervals.get(newIntervals.size() -
1).start + ", " + newIntervals.get(newIntervals.size() - 1).end + "]]");
        }
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.1407 sec	30.5 KB
Testcase 1	Easy	Hidden case	✓ Success	10	0.1558 sec	30.2 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1625 sec	30.2 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1537 sec	30.2 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.1652 sec	30.3 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1397 sec	30.2 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.1459 sec	29.7 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.1464 sec	30.3 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.1383 sec	30 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.1582 sec	30.2 KB

No Comments

QUESTION 6

✓ Correct Answer

Score 5

Add Intervals Space Complexity › Multiple Choice

QUESTION DESCRIPTION

Recall the solution to the Add Intervals debugging problem.

What is the space complexity of the solution?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- O(1)
- ✓ [● O(n)]
- O(n^2)
- O(n^3)

No Comments

QUESTION 7

✗ Wrong Answer

Score 0

Add Intervals Time Complexity › Multiple Choice

QUESTION DESCRIPTION

Recall the solution to the Add Intervals debugging problem.

What is the time complexity of the solution?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

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

No Comments

QUESTION 8

✓ Correct Answer

Score 150

3Sum Closest › Coding

QUESTION DESCRIPTION

Given an array `nums` of *n* integers and an integer `target`, find three integers in `nums` such that the sum is closest to `target`. Return the sum of the three integers. You may assume that each input would have exactly one solution.

Example:

```
Given array nums = [-1, 2, 1, -4], and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

CANDIDATE ANSWER

Language used: **Java 8**

```
    public static int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int result = nums[0] + nums[1] + nums[nums.length - 1];
        for (int i = 0; i < nums.length - 2; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            int start = i + 1, end = nums.length - 1;
            while (start < end) {
                int sum = nums[i] + nums[start] + nums[end];
                if (sum > target) {
                    end--;
                } else if (sum < target) {
                    start++;
                } else if (sum == target) {
                    return sum;
                }
                if (Math.abs(sum - target) < Math.abs(result - target)) {
                    result = sum;
                }
            }
        }
        return result;
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.1051 sec	24.7 KB
Testcase 1	Easy	Hidden case	✓ Success	10	0.1238 sec	24.7 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1024 sec	24.6 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.0951 sec	24.6 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.1022 sec	24.8 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0973 sec	24.6 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.0944 sec	25.1 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.1116 sec	24.8 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.0938 sec	24.8 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.0964 sec	24.9 KB
Testcase 10	Easy	Hidden case	✓ Success	10	0.1178 sec	24.8 KB
Testcase 11	Easy	Hidden case	✓ Success	10	0.0995 sec	24.9 KB
Testcase 12	Easy	Hidden case	✓ Success	10	0.1117 sec	24.7 KB
Testcase 13	Easy	Hidden case	✓ Success	10	0.1077 sec	24.9 KB
Testcase 14	Easy	Hidden case	✓ Success	10	0.1029 sec	24.8 KB

No Comments

QUESTION 9

✓ Correct Answer

Score 150

Rotate Image › Coding

QUESTION DESCRIPTION

You are given an *n* x *n* 2D matrix representing an image.
Rotate the image by 90 degrees (clockwise).

Note:
You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT**allocate another 2D matrix and do the rotation.

Example 1:

```
Given input matrix =
[
  [1,2,3],
  [4,5,6],
  [7,8,9]
],

rotate the input matrix in-place such that it becomes:
[
  [7,4,1],
    [8,5,2],
    [9,6,3]
  ]
```

Example 2:

```
Given input matrix =
[
  [ 5, 1, 9,11],
  [ 2, 4, 8,10],
  [13, 3, 6, 7],
  [15,14,12,16]
],

rotate the input matrix in-place such that it becomes:
[
  [15,13, 2, 5],
  [14, 3, 4, 1],
  [12, 6, 8, 9],
  [16, 7,10,11]
]
```

CANDIDATE ANSWER

Language used: **Java 8**

```java
    public static void rotate(int[][] matrix) {
        int n = matrix.length;
        int[][] ans = new int[n][n];

        int row = n - 1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                ans[i][j] = matrix[row--][i];
            }
            row = n - 1;
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = ans[i][j];
            }
        }
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0951 sec	24.7 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.098 sec	24.7 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.098 sec	24.8 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1092 sec	24.9 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.0995 sec	25 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1018 sec	24.9 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.0904 sec	24.8 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.0999 sec	24.5 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.0914 sec	24.7 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.1082 sec	25 KB

Testcase 10	Easy	Hidden case	✓ Success	10	0.1092 sec	24.7 KB
Testcase 11	Easy	Hidden case	✓ Success	10	0.1135 sec	24.8 KB
Testcase 12	Easy	Hidden case	✓ Success	10	0.1097 sec	24.8 KB
Testcase 13	Easy	Hidden case	✓ Success	10	0.1057 sec	24.8 KB
Testcase 14	Easy	Hidden case	✓ Success	10	0.0925 sec	24.8 KB

No Comments

QUESTION 10

✓ Correct Answer

Score 150

Minimum Window Substring › Coding

QUESTION DESCRIPTION

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

Example:

```
Input: S = "ADOBECODEBANC", T = "ABC"
Output: "BANC"
```

Note:

- If there is no such window in S that covers all characters in T, return the empty string `""`.
- If there is such window, you are guaranteed that there will always be only one unique minimum window in S.

CANDIDATE ANSWER

Language used: **Java 8**

```
    public static String minWindow(String s, String t) {
        int [] map = new int[128];
        for (char c : t.toCharArray()) {
            map[c]++;
        }
        int start = 0, end = 0, minStart = 0, minLen = Integer.MAX_VALUE,
counter = t.length();
        while (end < s.length()) {
            final char c1 = s.charAt(end);
            if (map[c1] > 0) counter--;
            map[c1]--;
            end++;
            while (counter == 0) {
                if (minLen > end - start) {
                    minLen = end - start;
                    minStart = start;
                }
                final char c2 = s.charAt(start);
                map[c2]++;
                if (map[c2] > 0) counter++;
                    start++;
            }
        }
        return minLen == Integer.MAX_VALUE ? "" : s.substring(minStart,
minStart + minLen);
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.1244 sec	24.7 KB
Testcase 2	Easy	Hidden case	✓ Success	10	0.1029 sec	24.7 KB
Testcase 3	Easy	Hidden case	✓ Success	10	0.1012 sec	24.5 KB
Testcase 4	Easy	Hidden case	✓ Success	10	0.0939 sec	24.8 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.1165 sec	24.7 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.098 sec	24.7 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.0932 sec	24.7 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.0975 sec	24.8 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.1057 sec	24.9 KB
Testcase 10	Easy	Hidden case	✓ Success	10	0.1045 sec	24.5 KB
Testcase 11	Easy	Hidden case	✓ Success	10	0.0932 sec	24.7 KB
Testcase 12	Easy	Hidden case	✓ Success	10	0.0933 sec	24.5 KB
Testcase 13	Easy	Hidden case	✓ Success	10	0.1049 sec	24.6 KB
Testcase 14	Easy	Hidden case	✓ Success	10	0.1191 sec	24.5 KB
Testcase 15	Easy	Hidden case	✓ Success	10	0.0984 sec	24.6 KB

No Comments