You can view this report online at : https://www.hackerrank.com/x/tests/1121391/candidates/27774347/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Strings and Arrays Assessment 2021**
Taken On:	19 Jul 2021 17:05:36 PDT
Time Taken:	88 min 27 sec/ 90 min
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:	

49.1%

285/580

scored in **Strings and Arrays Assessment 2021** in 88 min 27 sec on 19 Jul 2021 17:05:36 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	Sorting Algorithm Modification > **Multiple Choice**	1 min 9 sec	5/ 5	✓
Q2	Sorting Algorithm Efficiency > **Multiple Choice**	46 sec	5/ 5	✓
Q3	Runtime Analysis > **Multiple Choice**	2 min 2 sec	0/ 5	✗
Q4	Add Intervals Output > **Multiple Choice**	1 min 16 sec	5/ 5	✓
Q5	Add Intervals Debugging > **Coding**	15 min 56 sec	100/ 100	!
Q6	Add Intervals Space Complexity > **Multiple Choice**	33 sec	5/ 5	✓

Q7	Add Intervals Time Complexity > Multiple Choice	16 sec	5/ 5	✓
Q8	3Sum Closest > Coding	7 min 40 sec	140/ 150	✓
Q9	Rotate Image > Coding	25 min 37 sec	10/ 150	✓
Q10	Minimum Window Substring > Coding	33 min 1 sec	10/ 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

Needs Review

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**

```
1  /**
2   * Definition for an interval.
3   * public class Interval {
4   *     int start;
5   *     int end;
6   *     Interval(int s, int e) { start = s; end = e; }
7   * }
8   */
9  public static ArrayList<Interval> addInterval(ArrayList<Interval>
```

```
intervals, Interval newInterval) {
        ArrayList<Interval> result = new ArrayList<Interval>();
        int i = 0;
        while (i < intervals.size() && intervals.get(i).end <
newInterval.start) {
            result.add(intervals.get(i));
            i++;
        }
        while (i < intervals.size() && intervals.get(i).start <=
newInterval.end) {
            newInterval.start = Math.min(newInterval.start,
intervals.get(i).start);
            newInterval.end = Math.max(newInterval.end,
intervals.get(i).end);
            i++;
        }
        result.add(newInterval);
        while (i < intervals.size()) {
            result.add(intervals.get(i));
            i++;
        }

        return result;
        /*for (Interval interval : intervals) {
            if (interval.end < newInterval.start){
                result.add(interval);
                System.out.println("SAVED interval: " + interval.start + ", "
+ interval.end);
            } else {
                newInterval = new Interval(interval.start,
Math.max(newInterval.end, interval.end));
                System.out.println("NEW interval: " + newInterval.start + ",
" + newInterval.end);
            } else if (interval.start > newInterval.end) {
                result.add(newInterval);
            } else if (interval.end >= newInterval.start || interval.start <=
newInterval.end) {
                newInterval = new Interval(Math.min(interval.start,
newInterval.start), Math.max(newInterval.end, interval.end));
            }
        }

        result.add(newInterval);
        System.out.println("LAST interval: " + newInterval.start + ", " +
newInterval.end);*/
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.1192 sec	25 KB
Testcase 1	Easy	Hidden case	Success	10	0.1189 sec	25 KB
Testcase 2	Easy	Hidden case	Success	10	0.0884 sec	25.1 KB
Testcase 3	Easy	Hidden case	Success	10	0.0939 sec	25 KB
Testcase 4	Easy	Hidden case	Success	10	0.1152 sec	25.1 KB
Testcase 5	Easy	Hidden case	Success	10	0.103 sec	25.1 KB
Testcase 6	Easy	Hidden case	Success	10	0.0936 sec	25 KB
Testcase 7	Easy	Hidden case	Success	10	0.1034 sec	25.2 KB
Testcase 8	Easy	Hidden case	Success	10	0.1235 sec	25.3 KB

Testcase 9	Easy	Hidden case	Success	10	0.0884 sec	25.1 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

Correct Answer

Score 5

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 140

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 min = Integer.MAX_VALUE;
    for (int i = 0; i < nums.length-2; i++) {
        int left = i + 1;
        int right = nums.length-1;
        while (left < right) {
            int sum = nums[i] + nums[left] + nums[right];
            if (sum < target)
                left++;
            else if (sum > target)
                right--;
            else
                return sum;

            if (Math.abs(sum-target) < Math.abs(min-target))
                min = sum;
        }
    }
    return min;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.1293 sec	24.9 KB
Testcase 1	Easy	Hidden case	Success	10	0.1139 sec	25.1 KB
Testcase 2	Easy	Hidden case	Success	10	0.0948 sec	24.9 KB
Testcase 3	Easy	Hidden case	Success	10	0.1366 sec	25 KB
Testcase 4	Easy	Hidden case	Success	10	0.0939 sec	24.7 KB
Testcase 5	Easy	Hidden case	Success	10	0.0906 sec	25.1 KB
Testcase 6	Easy	Hidden case	Success	10	0.0953 sec	24.9 KB
Testcase 7	Easy	Hidden case	Success	10	0.1257 sec	25 KB
Testcase 8	Easy	Hidden case	Success	10	0.1017 sec	24.8 KB
Testcase 9	Easy	Hidden case	Success	10	0.0884 sec	24.9 KB
Testcase 10	Easy	Hidden case	Success	10	0.0959 sec	24.9 KB
Testcase 11	Easy	Hidden case	Success	10	0.1062 sec	24.9 KB
Testcase 12	Easy	Hidden case	Success	10	0.0904 sec	25.1 KB
Testcase 13	Easy	Hidden case	Wrong Answer	0	0.1346 sec	24.9 KB

Testcase 14	Easy	Hidden case	Success	10	0.1227 sec	25 KB

No Comments

QUESTION 9

Correct Answer

Score 10

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**

```
1    public static void rotate(int[][] matrix) {
2        int N = matrix.length;
3        int top = 0;
4        int right = N-1;
5        int bottom = N-1;
```

```
        int left = 0;
        /*while (true)*/ {
            int[] tmp1 = new int[N];
            for (int i = 0; i < N; i++) {
                tmp1[i] = matrix[i][right];
                matrix[i][right] = matrix[top][i];
            }
            for (int i = bottom; i >= 0; i--) {
                matrix[top][bottom-i] = matrix[i][left];
            }
            for (int i = right; i >= 0; i--)
                matrix[i][left] = matrix[bottom][i];
            for (int i = 0; i < bottom; i++) {
                matrix[bottom][bottom-i] = tmp1[i];
            }
        }
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Wrong Answer	0	0.1279 sec	25 KB
Testcase 1	Easy	Sample case	Wrong Answer	0	0.097 sec	25.1 KB
Testcase 2	Easy	Hidden case	Wrong Answer	0	0.0952 sec	25.1 KB
Testcase 3	Easy	Hidden case	Wrong Answer	0	0.0985 sec	24.9 KB
Testcase 4	Easy	Hidden case	Wrong Answer	0	0.0937 sec	24.9 KB
Testcase 5	Easy	Hidden case	Wrong Answer	0	0.0965 sec	24.9 KB
Testcase 6	Easy	Hidden case	Wrong Answer	0	0.0977 sec	25 KB
Testcase 7	Easy	Hidden case	Wrong Answer	0	0.1245 sec	24.8 KB
Testcase 8	Easy	Hidden case	Wrong Answer	0	0.103 sec	25 KB
Testcase 9	Easy	Hidden case	Success	10	0.0892 sec	24.9 KB
Testcase 10	Easy	Hidden case	Wrong Answer	0	0.127 sec	25 KB
Testcase 11	Easy	Hidden case	Wrong Answer	0	0.0956 sec	24.9 KB
Testcase 12	Easy	Hidden case	Wrong Answer	0	0.0967 sec	25 KB
Testcase 13	Easy	Hidden case	Wrong Answer	0	0.0865 sec	25 KB
Testcase 14	Easy	Hidden case	Wrong Answer	0	0.1273 sec	25 KB

No Comments

QUESTION 10

Correct Answer

Score 10

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) {
    Map<Character, Integer> map = new HashMap<>();
    for (char c : t.toCharArray()) {
        int count = map.getOrDefault(c, 0);
        map.put(c, count+1);
    }

    int start = -1;
    int end = -1;
    String result = s;
    int min = Integer.MAX_VALUE;
    for (int i = 0; i < s.length(); i++) {
        char c = s.charAt(i);
        if (map.containsKey(c)) {
            // ?????
            int count = map.get(c);
            if (count != 0) {
                map.put(c, count-1);
                if (start == -1)
                    start = i;
            } else {
                boolean isMapEmpty = false;
                int tmp = 0;
                for (int value : map.values()) {
                    if (value == 0)
                        tmp++;
                }
                if (tmp == map.size()) {
                    end = i;
                    if (end-start+1 < min) {
                        min = end-start+1;
                        result = s.substring(start, end+1);
                    }
                }
            }
        }
    }
    return result;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Wrong Answer	0	0.0959 sec	24.7 KB
Testcase 2	Easy	Hidden case	Success	10	0.1266 sec	24.9 KB
Testcase 3	Easy	Hidden case	Wrong Answer	0	0.0904 sec	24.8 KB
Testcase 4	Easy	Hidden case	Wrong Answer	0	0.0872 sec	24.9 KB

Testcase 5	Easy	Hidden case	Wrong Answer	0	0.1179 sec	24.9 KB
Testcase 6	Easy	Hidden case	Wrong Answer	0	0.0874 sec	25.1 KB
Testcase 7	Easy	Hidden case	Wrong Answer	0	0.0902 sec	25 KB
Testcase 8	Easy	Hidden case	Wrong Answer	0	0.103 sec	25 KB
Testcase 9	Easy	Hidden case	Wrong Answer	0	0.1268 sec	25.1 KB
Testcase 10	Easy	Hidden case	Wrong Answer	0	0.0977 sec	25.2 KB
Testcase 11	Easy	Hidden case	Wrong Answer	0	0.1044 sec	24.8 KB
Testcase 12	Easy	Hidden case	Wrong Answer	0	0.1066 sec	24.9 KB
Testcase 13	Easy	Hidden case	Wrong Answer	0	0.1287 sec	24.9 KB
Testcase 14	Easy	Hidden case	Wrong Answer	0	0.0966 sec	25 KB
Testcase 15	Easy	Hidden case	Wrong Answer	0	0.1021 sec	25 KB

No Comments

PDF generated at: 20 Jul 2021 01:36:05 UTC