You can view this report online at : https://www.hackerrank.com/x/tests/1330880/candidates/42301880/report

Full Name:	Ruchit Bhardwaj
Email:	ruchitbh@usc.edu
Test Name:	**CodePath SE103: Unit 7 Assessment - Summer 2022**
Taken On:	19 Jul 2022 17:35:29 PDT
Time Taken:	40 min 45 sec/ 90 min
Work Experience:	3 years
Invited by:	CodePath
Skills Score:	Problem Solving (Basic) 128/175 Problem Solving (Intermediate) 0/75
Tags Score:	Algorithms 50/50 Arrays 3/125 Easy 53/100 Implementation 125/125 Interviewer Guidelines 3/50 Medium 75/150 Prefix Sum 0/75 Problem Solving 75/75 Sorting 0/75 Strings 125/125

68.7%

268/390

scored in **CodePath SE103: Unit 7 Assessment - Summer 2022** in 40 min 45 sec on 19 Jul 2022 17:35:29 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Challenge: Alien Language > Coding	6 min 44 sec	140/ 140	✓
Q2	Zig-Zag Array > Coding	9 min 59 sec	3/ 50	✓
Q3	Efficient Workers > Coding	15 min 31 sec	0/ 75	✗
Q4	Circular Printer > Coding	2 min 32 sec	50/ 50	✓
Q5	Ancestral Names > Coding	5 min 40 sec	75/ 75	✓

QUESTION 1

Correct Answer

Challenge: Alien Language > Coding

Score 140

QUESTION DESCRIPTION

Given a sorted dictionary (array of words) of an alien language, find order of characters in the language.

Examples:

```
Input:  words[] = {"baa", "abcd", "abca", "cab", "cad"}
Output: Order of characters is 'b', 'd', 'a', 'c'
Note that words are sorted and in the given language "baa"
comes before "abcd", therefore 'b' is before 'a' in output.
Similarly we can find other orders.

Input:  words[] = {"caa", "aaa", "aab"}
Output: Order of characters is 'c', 'a', 'b'
```

If you're trying to understand how the test cases / inputs work, you can analyze the code outside of the function you're trying to implement to see how the input string is parsed to create the graph.

CANDIDATE ANSWER

Language used: **Java 8**

```
/*
This function finds and returns the order
of characers from a sorted array of words.
alpha is number of possible alphabets
starting from 'a'. For simplicity, this
function is written in a way that only
first 'alpha' characters can be
in words array. For example if alpha
is 7, then words[] should contain words
having only 'a', 'b','c' 'd', 'e', 'f', 'g'

Graph class
 Graph(numVertices)
 addEdge(startVertex, endVertex)
 topologicalSort()
 */

 public static ArrayList<Character> getOrder(String[] words, int alpha) {
     int n = words.length;
    // Create a graph with 'alpha' edges
    Graph graph = new Graph(alpha);

    for (int i = 0; i < n - 1; i++) {
        // Take the current two words and find the first
        // mismatching character
        String word1 = words[i];
        String word2 = words[i + 1];
        for (int j = 0; j < Math.min(word1.length(),
                                     word2.length());
             j++) {
            // If we find a mismatching character, then
            // add an edge from character of word1 to
            // that of word2
            if (word1.charAt(j) != word2.charAt(j)) {
                graph.addEdge(word1.charAt(j) - 'a',
                              word2.charAt(j) - 'a');
                break;
            }
        }
```

```
        }

        // Print topological sort of the above created graph
        return graph.topologicalSort();
    }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0928 sec	24.6 KB
Testcase 1	Easy	Hidden case	Success	10	0.1205 sec	25 KB
Testcase 2	Easy	Hidden case	Success	10	0.1322 sec	24.8 KB
Testcase 3	Easy	Hidden case	Success	10	0.0892 sec	24.9 KB
Testcase 4	Easy	Hidden case	Success	10	0.0993 sec	24.9 KB
Testcase 5	Easy	Hidden case	Success	10	0.0981 sec	25.1 KB
Testcase 7	Easy	Hidden case	Success	10	0.1083 sec	24.6 KB
Testcase 8	Easy	Hidden case	Success	10	0.091 sec	24.8 KB
Testcase 9	Easy	Hidden case	Success	10	0.1046 sec	24.9 KB
Testcase 10	Easy	Hidden case	Success	10	0.1005 sec	25 KB
Testcase 11	Easy	Hidden case	Success	10	0.1012 sec	24.5 KB
Testcase 12	Easy	Hidden case	Success	10	0.1059 sec	24.7 KB
Testcase 13	Easy	Hidden case	Success	10	0.0919 sec	24.7 KB
Testcase 14	Easy	Hidden case	Success	10	0.1064 sec	24.8 KB

No Comments

QUESTION 2

Correct Answer

Score 3

Zig-Zag Array › Coding

Easy | Arrays | Interviewer Guidelines

QUESTION DESCRIPTION

Given an array of integers, change it in such a way that it follows a zig-zag pattern. A zig-zag array is one where for each integer, its adjacent integers are both greater than or less than itself. In other words, using L to mean a lower value and H to mean higher, the array follows either the pattern [L,H,L,H...] or [H,L,H,L...]. To make the array a zig-zag array, you can replace any element with any other integer (positive, negative, or zero). What is the minimum number of replacements required to accomplish this?

Example

arr = [1, 2, 3, 4, 5]

```
Original: [1, 2, 3, 4, 5]
LHLHL:    [1, 2, -, 4, -]
HLHLH:    [+, 2, 3, -, 5]
```

To achieve an array starting with a low value, both the 3 and the 5 need to be reduced to any value less than 2 and 4 respectively.

To achieve an array starting with a high value, the 1 needs to be increased (any value > 2) and the 4 needs to be decreased (any value < 3)

In this case, creating either form of zig-zag array takes a minimum of 2 replacements, the final answer.

Function Description

Complete the function *minOperations* in the editor below.

minOperations has the following parameter:

int *arr[n]:* an array of integers

Returns

int: the minimum number of operations required to turn *arr* into a zig-zag array

Constraints

- $1 \le n \le 10^5$
- $1 \le arr[i] \le 10^9$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *arr*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer, *arr[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
8     →   n = 8
2     →   arr = [2, 1, 2, 3, 4, 5, 2, 9]
1
2
3
4
5
2
9
```

Sample Output

```
2
```

Explanation

```
Original: [2, 1, 2, 3, 4, 5, 2, 9]
           L  H  L  H  L  H  L  H
LHLHLHLH: [2, +, 2, 3, -, 5, 2, 9]

           H  L  H  L  H  L  H  H
HLHLHLHL: [2, 1, 2, -, 4, -, 2, -]
```

For the LHLH... pattern, replace the second value (1) with a number greater than 2 and the fifth value (4) with a number less than 3.

For the HLHL... pattern, replace the fourth value (3) with a number less than 2, and the sixth value (5) and the eighth value (9) with a number less than 2.

The LHLH... pattern only requires two replacements.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
6    →    arr[] size n = 6
1    →    arr = [1, 2, 4, 4, 5, 6]
2
4
4
```

```
5
6
```

Sample Output

```
2
```

Explanation

```
Original: [1, 2, 4, 4, 5, 6]
           L  H  L  H  L  H
LHLHLHLH: [1, 2, -, 4, -, 6]

           H  L  H  L  H  L
HLHLHLHL: [+, 2, 4, -, 5, -]
```

Starting with a low value takes 2 replacements, while starting with a high value takes 3. Return 2.

INTERVIEWER GUIDELINES

▼ Hint 1

Do each case (even indexed is greater, odd indexed is greater) separately. Take the minimum of the two as the answer.

▼ Hint 2

In say the even case, you should decrease each even-indexed element until it is lower than its immediate neighbors. Similarly for the odd case.

▼ Solution

Concepts Covered: Basic Programming Skills, Loops, Arrays, Problem Solving. The problem tests the candidate's ability to use loops and arrays. It requires the candidate to come up with an algorithm to find the minimum number of integers to modify in an array such that the resulting array is a zig-zag array in a constrained time and space complexity.

Optimal Solution:

Do each case (even indexed is greater, odd indexed is greater) separately. Take the minimum of the two as the answer. In the even case, you should decrease each even-indexed element until it is lower than its immediate neighbors. Similarly for the odd cases.

To make a number smaller than both its neighbors it's always optimal to change that number to a very small number.

Similarly, to make a number greater than both its neighbors its always optimal to change that number to a very large number so that other indices do not get affected by that modification.

Time Complexity: O(n)

```
from copy import copy

# the maximum value in the given array is 1e9
# set this to greater than limit so replacement
# will be higher/lower in all cases
maxval = 1e9+1

def minOperations(arr):
    ans1 = 0
    ans2 = 0

    arr1 = copy(arr) # starts with high value
    arr2 = copy(arr) # starts with low value

    n = len(arr)

    for i in range(1, n):
        # even indices
        if i % 2 == 0:
            # value must be lower
            if arr1[i] >= arr1[i-1]:
```

```
            if arr1[i] >= arr1[i-1]:
                arr1[i] = -maxval
                ans1 += 1
            # value must be higher
            if arr2[i] <= arr2[i-1]:
                arr2[i] = maxval
                ans2 += 1
        # odd indices
        elif (i % 2) == 1:
            # value must be higher
            if arr1[i] <= arr1[i-1]:
                arr1[i] = maxval
                ans1 += 1
            # value must be lower
            if arr2[i] >= arr2[i-1]:
                arr2[i] = -maxval
                ans2 += 1

    return min(ans1, ans2)
```

Brute Force Approach: For each position consider all the numbers that can be placed at that position such that it is either rather than both its immediate neighbors or smaller than both its neighbors. Time complexity: O(mx ^ n), where mx = maximum element in the array.

Error Handling:

1. The minimum number of changes for both the odd and even cases must be counted and the minimum must be returned.
2. The value of the array must be copied in two arrays to be changed accordingly rather than making the changes in the same array.
3. It's advisable to change the value of the index to -1e9 or smaller when its required to be smaller than both its neighbors. Similarly, for a value to be larger, it must be changed to 1e9 or greater.

▼ Complexity Analysis

Time Complexity - O(n).

The algorithm require linear time operations.

Space Complexity - O(n)

For the two cases, we need to make copies of the original array to make the modification of array index value easier.

▼ Follow up Question

Given an array of integers, what is the minimum number of moves required to make it zigzag?
A *move* consists of choosing any element and decreasing it by 1.

Solution: There are two possible ways the array can satisfy the zigzag requirement: 1: start with a zig (increasing) 2. start with zag (decreasing). There is a loop that goes through the whole list of numbers, in each step we keep track of the number of subtractions needed to satisfy the requirements for both possible ways mentioned before. We also keep track of the last modified number.

Psuedo Code -

```
def movesToMakeZigzag(nums):

        zig, zag = 0, 0
        prev_zig, prev_zag = nums[0], nums[0]

        for i in range(1, len(nums)):
            if i % 2 == 0:
                zig += max(0, prev_zig - nums[i] + 1)
                prev_zig = nums[i]
                zag += max(0, nums[i] - prev_zag + 1)
                prev_zag = nums[i] - max(0, nums[i] - prev_zag + 1)
            else:
                zag += max(0, prev_zag - nums[i] + 1)
                prev_zag = nums[i]
                zig += max(0, nums[i] - prev_zig + 1)
```

```
                    prev_zig = nums[i] - max(0, nums[i] - prev_zig + 1)

        return min(zig, zag)
```

CANDIDATE ANSWER

Language used: **Java 8**

```
class Result {

    /*
     * Complete the 'minOperations' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY arr as parameter.
     */

    public static int minOperations(List<Integer> A) {
        int res[] = new int[2],  n = A.size(), left, right;
        for (int i = 0; i < n; ++i) {
            left = i > 0 ? A.get(i - 1) : 1001;
            right = i + 1 < n ? A.get(i + 1) : 1001;
            res[i % 2] += Math.max(0, A.get(i) - Math.min(left, right) + 1);
        }
        return Math.min(res[0], res[1]);

    }

}


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Wrong Answer	0	0.1492 sec	29.6 KB
TestCase 1	Easy	Sample case	Wrong Answer	0	0.1325 sec	29.8 KB
TestCase 2	Easy	Sample case	Success	1	0.1435 sec	29.9 KB
TestCase 3	Easy	Sample case	Success	2	0.1421 sec	29.8 KB
TestCase 4	Easy	Hidden case	Wrong Answer	0	0.1366 sec	29.8 KB
TestCase 5	Easy	Sample case	Wrong Answer	0	0.1598 sec	30.8 KB
TestCase 6	Easy	Hidden case	Wrong Answer	0	0.1492 sec	30.6 KB
TestCase 7	Easy	Hidden case	Wrong Answer	0	0.1653 sec	30.9 KB
TestCase 8	Easy	Hidden case	Wrong Answer	0	0.1557 sec	31 KB
TestCase 9	Easy	Hidden case	Wrong Answer	0	0.1457 sec	30.8 KB
TestCase 10	Easy	Hidden case	Wrong Answer	0	0.3312 sec	56.1 KB
TestCase 11	Easy	Hidden case	Wrong Answer	0	0.3482 sec	54.9 KB
TestCase 12	Easy	Hidden case	Wrong Answer	0	0.3334 sec	55.2 KB
TestCase 13	Easy	Hidden case	Wrong Answer	0	0.302 sec	56.2 KB
TestCase 14	Easy	Hidden case	Wrong Answer	0	0.3977 sec	55.5 KB

No Comments

QUESTION 3

Wrong Answer

Score 0

Efficient Workers › Coding

Sorting | Prefix Sum | Medium | Arrays

QUESTION DESCRIPTION

A group of workers gathered to complete a task. Each worker has an *efficiency* rating. They will be grouped in pairs so an even number of workers are required. The cost of a pair is the absolute difference of the efficiencies assigned to the workers. The cost of the task is the sum of the costs of all pairs formed. There are an odd number of workers to choose from, so one worker will not be paired. Select the worker to exclude so the task's cost is minimized.

Given *n* workers and *efficiency* for each worker, find a configuration of the workers such that the cost of the task is the minimum possible. Return the minimum cost as the answer.

Example

efficiency = [4, 2, 8, 1, 9]
Using 1-based indexing, if worker 1 is excluded and the indices of the pairs are (2, 4) and (3, 5), the cost of the task is |2 - 1| + |8 - 9| = 2.

This is the minimum possible cost so return 2.

.

Function Description

Complete the function *findMinCost* in the editor below.

findMinCost has the following parameter(s):

int efficiency[n]: the efficiency of each worker

Returns

int: the minimum possible cost

Constraints

- $3 \le n < 10^5$
- $1 \le efficiency[i] \le 10^9$
- *n is odd*

▼ Input Format for Custom Testing

The first line contains an integer *n*, the size of the array *efficiency*.
Each of the next n lines contains an integer *efficiency[i]*.

▼ Sample Case 0

Sample Input 0

```
STDIN       FUNCTION
-----       --------
5       →   n = 5
4       →   efficiency = [4, 1, 2, 16, 8]
1
2
16
8
```

Sample Output 0

```
5
```

Explanation

Exclude worker 4 and make the pairs (2, 3) and (1, 5). The cost of the task is |1 - 2| + |4 - 8| = 5.

▼ Sample Case 1

Sample Input 1

```
STDIN        FUNCTION
-----        --------
7      →     n = 7
2      →     efficiency=  = [2, 13, 12, 9, 6, 3, 2]
16
12
9
6
3
2
```

Sample Output 1

```
4
```

Explanation

Exclude worker 4 and make the pairs (1, 7), (2, 3), and (5, 6). The cost is |2 - 2| + |13 - 12| + |6 - 3| = 4.

INTERVIEWER GUIDELINES

▼ Solution

Skills: Prefix Sums, Sorting

Optimal Solution:

This problem can be solved using prefix sums. We are given n elements, where n is odd. So, we can represent *n = 2 * p + 1*. We basically have to find *p* pairs such that the sum of absolute differences is minimum. let's first sort our initial array (as it doesn't affect anything in the problem). We can observe for a given array of even length it's always optimal to match (first worker with the second worker) (third worker with the fourth worker) and so on after sorting the initial array. In this problem, we are given an array of odd lengths and we have to remove one worker so that cost will be minimized, How to decide which worker should not be chosen - Let us build a brute force solution. We choose each element as "leaving worker", then for the remaining elements, create a new list and sort it. Then, add the value (*arr[i] - arr[i -1]*) for all even position i (1 - indexed). We can optimize this approach by prefix and suffix sums. We will create the prefix and suffix sum in the following way:

pref[i] = pref[i - 2] + A[i] - A[i - 1]

suff[i] = suff[i + 2] + A[i + 1] - A[i]

In this way, *pref[i]* denotes the sum of all adjacent pairs up to *i*, similarly, *suff[i]* denotes the sum of all adjacent pairs for the suffix up to *i*. using this we are reducing the time used for calculating the cost for each "leaving worker" by preprocessing the sum. Now, for each odd positioned *i*, if we choose this element as the "leaving worker", the answer would be the sum of *pref[i -1] + suff[i + 1].* The answer would be the minimum value for all odd positioned *i*. one can prove that it's always optimal to choose odd-indexed "leaving worker" rather than even-indexed.

```
def findMinCost(arr):
    n = len(arr)
    arr.sort()
    diff = []
    for i in range(0, n - 1):
        diff.append(arr[i + 1] - arr[i])
    suffix_odd = 0
    prefix_even = 0
    for i in range(0, n - 1):
        if i % 2 == 1:
            suffix_odd += diff[i]
    ans = suffix odd
```

```
    for i in range(0, n - 1):
        if i % 2 == 1:
            suffix_odd -= diff[i]
        else:
            prefix_even += diff[i]

        ans = min(ans, prefix_even + suffix_odd)

    return ans
```

▼ Complexity Analysis

Time Complexity - *O(n*logn)*
Since we are sorting our initial array.

Space Complexity - *O(n)*
Since we are building prefix and suffix sum.

CANDIDATE ANSWER

Language used: **Java 8**

```
class Result {

    /*
     * Complete the 'findMinCost' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY efficiency as parameter.
     */

    public static int findMinCost(List<Integer> efficiency) {
        int n = efficiency.size();
        int i;
        int c = 0;
        float s=0;
        for(i=0;i<n;i++)
        {
        s=s+efficiency.get(i);
        }
        s=s+3;
        c=(int)s/3;
        return c;

    }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	⊗ Wrong Answer	0	0.1475 sec	30 KB
Testcase 1	Easy	Sample case	⊗ Wrong Answer	0	0.1368 sec	29.8 KB
Testcase 2	Easy	Sample case	⊗ Wrong Answer	0	0.1457 sec	29.6 KB

Testcase 3	Easy	Hidden case	Wrong Answer	0	0.1444 sec	29.9 KB
Testcase 4	Easy	Hidden case	Wrong Answer	0	0.1263 sec	29.6 KB
Testcase 5	Easy	Hidden case	Wrong Answer	0	0.1584 sec	29.7 KB
Testcase 6	Easy	Hidden case	Wrong Answer	0	0.1424 sec	30 KB
Testcase 7	Medium	Hidden case	Wrong Answer	0	0.1969 sec	30.4 KB
Testcase 8	Medium	Hidden case	Wrong Answer	0	0.1692 sec	31.6 KB
Testcase 9	Medium	Hidden case	Wrong Answer	0	0.15 sec	30.5 KB
Testcase 10	Medium	Hidden case	Wrong Answer	0	0.1551 sec	30.8 KB
Testcase 11	Hard	Hidden case	Wrong Answer	0	0.2919 sec	56.1 KB
Testcase 12	Hard	Hidden case	Wrong Answer	0	0.3365 sec	56.1 KB
Testcase 13	Hard	Hidden case	Wrong Answer	0	0.31 sec	52 KB
Testcase 14	Hard	Hidden case	Wrong Answer	0	0.2685 sec	55.8 KB

No Comments

QUESTION 4

Correct Answer

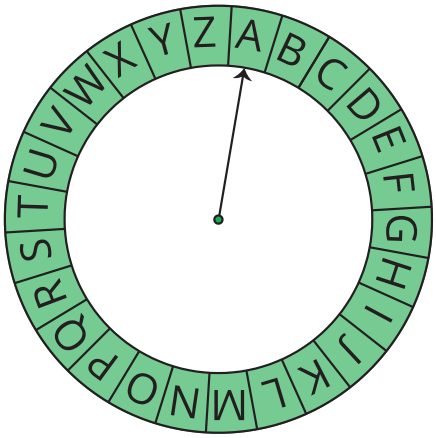
Score 50

Circular Printer › Coding

Easy | Strings | Implementation | Algorithms

QUESTION DESCRIPTION

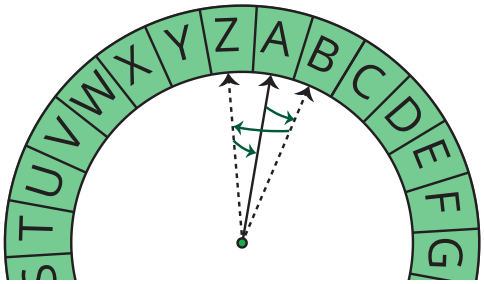
A company has invented a new type of printing technology—a circular printer that looks like this:

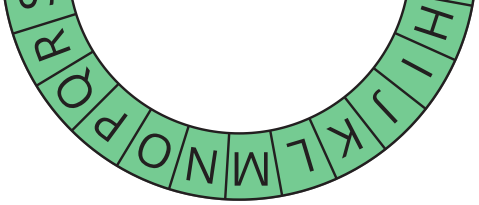


It is a circular printer wheel with the letters A through Z in sequence. It wraps so A and Z are adjacent. The printer has a pointer that is initially at 'A'. Moving from any character to any adjacent character takes 1 second. It can move in either direction. Given a string of letters, what is the minimum time needed to print the string? (Note: Assume that printing does not take any time. Only consider the time it takes for the pointer to move.)

Example

s = "BZA"





Total time to print “BZA” = 1+ 2 + 1 = 4 seconds

First, move the pointer from 'A' to 'B' (1 second), then from 'B' to 'Z' (2 seconds), and finally from 'Z' to 'A' (1 second). So the minimum time needed to print "BZA" is 4 seconds.

Function Description

Complete the function *getTime* in the editor below.

getTime has the following parameter:

string *s:* the string of characters that need to be printed

Returns:

int: the minimum number of seconds needed to print *s*

Constraints

- $1 \leq$ length of $s \leq 10^5$

▼ Input Format For Custom Testing

The first line contains a string, *s*, the string to be printed.

▼ Sample Case 0

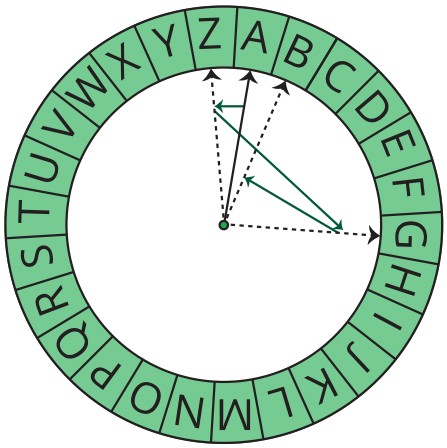
Sample Input For Custom Testing

```
STDIN       Function
-----       --------
AZGB   =>    s = "AZGB"
```

Sample Output

```
13
```

Explanation



Total time to print “AZGB” = 0 + 1 + 7 + 5 = 13 seconds

Initially, the pointer is at 'A', so there is no need to move it for the first character. First, move the pointer from 'A' to 'Z' (1 second), then from 'Z' to 'G' (7 seconds), and finally from 'G' to 'B' (5 seconds).

Therefore, the total time is 1 + 7 + 5 = 13.

▼ Sample Case 1

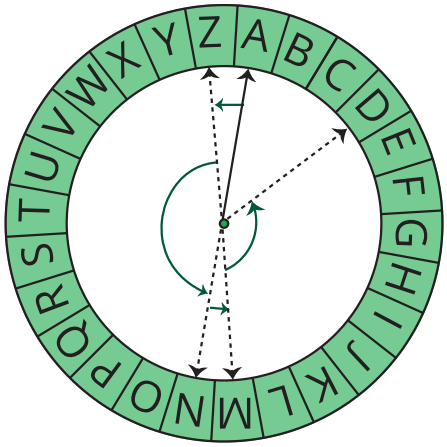
Sample Input For Custom Testing

```
ZNMD
```

Sample Output

```
23
```

Explanation



Total time to print "ZNMD" = 1 + 12 + 1 + 9 = 23 seconds

First, move the pointer from 'A' to 'Z' (1 second), then from 'Z' to 'N' (12 seconds), then from 'N' to 'M' (1 second), and finally from 'M' to 'D' (9 seconds). The total time is 1 + 12 + 1 + 9 = 23.

INTERVIEWER GUIDELINES

▼ Hint 1

For each move from one character to another, consider both its clockwise and counter clockwise moves. Move the shorter distance.

▼ Hint 2

For each adjacent character, sum the minimum distances to get the answer.

▼ Solution

Concepts Covered: Basic Programming Skills, Loops, Strings, Problem Solving. The problem tests the candidate's ability to use loops and strings. It requires the candidate to come up with an algorithm to find the minimum time to print a word in a circular printer in a constrained time and space complexity.

Optimal Solution:

For every move of the printer, sum up the minimum time of the two directions, i.e. clockwise and counterclockwise. The resulting sum is our answer.

```
def getTime(s):

    ans = 0

    # always start at 'A'
    cur_char = ord('A')

    for i in range(len(s)):

        # get distance between characters
        diff = abs(ord(s[i])- cur char)
```

```
        # see if it's closer to left or right
        ans += min(diff, 26 - diff)

        # and point to next character
        cur_char = ord(s[i])

    return ans
```

Error Handling:

1. For each character, both the clockwise and counterclockwise distance must be considered.
2. To find the other distance between a pair of letters, subtract from 26 rather than 25.

▼ Complexity Analysis

Time Complexity - O(n).

We iterate for all the letters in the word.

Space Complexity - O(1) - No extra space is required.

▼ Follow up Question

Let's suppose for the same problem, we can skip at most one letter while printing. Then what is the minimum time required to print the word?

Solution: We maintain two arrays, pf[ ] and sf[ ]. pf[i] denotes the minimum time required to print the string s[0, i] and sf[i] denotes the minimum time required to type the string s[i, n - 1]. So if we remove character s[i], we need to consider only character s[i - 1] and s[i + 1] as they now are adjacent and pf[i - 1] and sf[i + 1].

Find the minimum sum for all such indices i.

Psuedo Code -

```
def getTime(s):
    # Write your code here
    n = len(s)
    pf = [0] * n
    sf = [0] * n

    for i in range(1, n):
        pf[i] = min(26 - abs(ord(s[i]) - ord(s[i - 1])), abs(ord(s[i]) -
ord(s[i - 1])))
        pf[i] += pf[i - 1]

    for i in range(n - 2, -1, -1):
        sf[i] =  min(26 - abs(ord(s[i]) - ord(s[i + 1])), abs(ord(s[i]) -
ord(s[i + 1])))
        sf[i] += sf[i + 1]

    ans = min(pf[n - 2], sf[1]) # remove the first or last letter
    for i in range(1, n - 1):
        x = min(26 - abs(ord(s[i + 1]) - ord(s[i - 1])), abs(ord(s[i +
1]) - ord(s[i - 1])))
        ans = min(ans, x + sf[i + 1] + pf[i - 1])

    return ans
```

CANDIDATE ANSWER

Language used: **Java 8**

```java
class Result {

    /*
     * Complete the 'getTime' function below.
     *
     * The function is expected to return a LONG_INTEGER.
     * The function accepts STRING s as parameter.
     */

    public static long getTime(String s) {
        char prev = 'A';
        long val = 0;
        for (int i = 0; i < s.length(); i++) {
            char curr = s.charAt(i);
            if (prev < curr) {
                String s3 = String.valueOf(prev);
                String s4 = String.valueOf(curr);
                int len = s4.compareTo(s3);
                if (len <= 13)
                    val = val + len;
                else
                    val = val + 26-len;
                prev = curr;
            } else if (prev > curr) {
                String s3 = String.valueOf(prev);
                String s4 = String.valueOf(curr);
                int len = s3.compareTo(s4);
                if (len <= 13)
                    val = val + len;
                else
                    val = val + 26-len;
                prev = curr;
            }
        }
        return val;

    }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	1	0.0752 sec	23.4 KB
TestCase 1	Easy	Sample case	Success	1	0.0968 sec	23.2 KB
TestCase 2	Easy	Sample case	Success	1	0.0869 sec	23.4 KB
TestCase 3	Easy	Sample case	Success	2	0.0841 sec	23.1 KB
TestCase 4	Easy	Hidden case	Success	2	0.0877 sec	23.4 KB
TestCase 5	Easy	Sample case	Success	3	0.0803 sec	23.6 KB
TestCase 6	Easy	Hidden case	Success	3	0.0781 sec	23.5 KB
TestCase 7	Easy	Hidden case	Success	3	0.0848 sec	23.5 KB
TestCase 8	Easy	Hidden case	Success	3	0.0761 sec	23.6 KB
TestCase 9	Easy	Hidden case	Success	3	0.0872 sec	23.5 KB
TestCase 10	Easy	Hidden case	Success	5	0.1054 sec	34.6 KB
TestCase 11	Easy	Hidden case	Success	5	0.1018 sec	34.6 KB
TestCase 12	Easy	Hidden case	Success	6	0.1068 sec	34.4 KB

TestCase 13	Easy	Hidden case	Success	6	0.1493 sec	34.7 KB
TestCase 14	Easy	Hidden case	Success	6	0.1167 sec	34.5 KB

No Comments

QUESTION 5

Correct Answer

Score 75

Ancestral Names › Coding

Medium | Strings | Implementation | Problem Solving

QUESTION DESCRIPTION

Given a list of strings comprised of a name and a Roman numeral, sort the list first by name, then by the decimal value of the Roman numeral.

In Roman numerals, a value is not repeated more than three times. At that point, a smaller value precedes a larger value to indicate subtraction. For example, the letter I represents the number *1*, and V represents *5*. Reason through the formation of *1* to *10* below, and see how it is applied in the following lines.

- *I*, *II*, *III*, *IV*, *V*, *VI*, *VII*, *VIII*, *IX*, and *X* represent *1* through *10*.
- *XX*, *XXX*, *XL*, and *L* are *20, 30, 40*, and *50*.
- For any other two-digit number *< 50*, concatenate the Roman numeral(s) that represent its multiples of ten with the Roman numeral(s) for its values *< 10*. For example, *43* is *40 + 3* = '*XL*' + '*III*' = '*XLIII*'

Example

names = ['Steven XL', 'Steven XVI', 'David IX', 'Mary XV', 'Mary XIII', 'Mary XX']

The result with Roman numerals is the expected return value. Written in decimal and sorted, they are *['David 9', 'Mary 13', 'Mary 15', 'Mary 20', 'Steven 16', 'Steven 40']*. The return array is *['David IX', 'Mary XIII', 'Mary XV', 'Mary XX', 'Steven XVI', 'Steven XL']*.

Function Description

Complete the function *sortRoman* in the editor below.

sortRoman has the following parameter:

names[n]: an array of strings comprised of names and roman numerals

Returns:

string[n]: an array of strings sorted first by given name, then by ordinal

Constraints

- *1 ≤ n ≤ 50*
- Each *names[i]* is a single string composed of *2* space-separated values: *givenName* and *romanNumeral*.
- *romanNumeral* represents a number between *1* and *50*, inclusive.
- *1 ≤ |givenName| ≤ 20*
- Each *givenName* starts with an uppercase letter *ascii[A-Z]* which is followed by lowercase letters *ascii[a-z]*.
- There is a space between *givenName* and *romanNumeral*
- Each *names[i]* is distinct.

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *names*.

Each of the next *n* lines contains an element *names[i]*.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       -----
2   →  names[] size n = 2
Louis IX    →  names = ['Louis IX', 'Louis VIII']
Louis VIII
```

Sample Output

```
Louis VIII
Louis IX
```

Explanation

Sort first by *givenName* then, if *givenName* is not unique, by the value of the Roman numeral. In decimal, the list is sorted *['Louis 8', 'Louis 9']*.

▼ Sample Case 1

Sample Input

```
STDIN       Function
-----       -----
2   →  names[] size n = 2
Philippe I    →  names = ['Philippe I', 'Philip II']
Philip II
```

Sample Output

```
Philip II
Philippe I
```

CANDIDATE ANSWER

Language used: **Java 8**

```
class Result {

    /*
     * Complete the 'sortRoman' function below.
     *
     * The function is expected to return a STRING_ARRAY.
     * The function accepts STRING_ARRAY names as parameter.
     */

    public static List<String> sortRoman(List<String> names) {
        Collections.sort(names, (s1, s2) -> {
            //split the strings up into name,roman
            String[] arr1 = s1.split(" ");
            String[] arr2 = s2.split(" ");

            //grab the numerical values of the romans
            int val1 = romanToInt(arr1[1]);
            int val2 = romanToInt(arr2[1]);

            //if the names are equal, compare the numerals
            if (arr1[0].equals(arr2[0])) {
                //if first one is greater than, push it back
                if (val1 > val2) {
```

```
                if (val1 > val2) {
                    return 1;
                }
                //if first one is less than, stay same
                else {
                    return -1;
                }
            }
            else { //if not same, just compare the names
                return arr1[0].compareTo(arr2[0]);
            }
        });
        return names;
    }

    public static int romanToInt(String roman) {
        int total = 0;
        //create hashmap to store the roman numerals
        HashMap<Character, Integer> romans = new HashMap<>();
        romans.put('I', 1);
        romans.put('V', 5);
        romans.put('X', 10);
        romans.put('L', 50);
        romans.put('C', 100);
        romans.put('D', 500);
        romans.put('M', 1000);
        for (int j = 0; j < roman.length(); j++) {
            char c = roman.charAt(j); //grab first char
            //check to see if next roman is greater
            if (j + 1 < roman.length() && romans.get(c) <
romans.get(roman.charAt(j + 1))) {
                //if next roman is greater, you need to subtract
                int add = romans.get(roman.charAt(j + 1)) - romans.get(c);
                total += add;
                j++; //skip over next one since already calculated
            }
            //if less than, just add in order
            else {
                total += romans.get(c);
            }
        }
        return total;
    }

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	1	0.1446 sec	29.4 KB
Testcase 1	Easy	Sample case	Success	1	0.1286 sec	29.6 KB
Testcase 2	Easy	Sample case	Success	1	0.1356 sec	30 KB
Testcase 3	Easy	Sample case	Success	6	0.1443 sec	29.6 KB
Testcase 4	Easy	Sample case	Success	11	0.1902 sec	29.9 KB
Testcase 5	Easy	Hidden case	Success	11	0.1377 sec	30.3 KB
Testcase 6	Easy	Hidden case	Success	11	0.163 sec	29.8 KB
Testcase 7	Easy	Hidden case	Success	11	0.1455 sec	30 KB
Testcase 8	Easy	Hidden case	Success	11	0.1487 sec	30 KB
Testcase 9	Easy	Hidden case	Success	11	0.1357 sec	30.5 KB

No Comments

PDF generated at: 20 Jul 2022 01:17:58 UTC